Broker-based Service-oriented Content Adaptation Framework

by

Mohd Farhan Md Fudzee
MSc. (IT), BSc. (IT), Dip. (Comp. Sc.)

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

Deakin University

September, 2011

DEAKIN UNIVERSITY

ACCESS TO THESIS – A

I certify that the thesis entitled

Broker-based Service-oriented Content Adaptation Framework

submitted for the degree of

Doctor of Philosophy

This thesis may be made available for consultation, loan and limited copying in accordance with the Copyright Act 1968.

'I certify that I am the student named below and that the information provided in the form is correct'

Full Name: Mohd Farhan Md Fudzee

Signature Redacted by Library

Signed : .....

Date : 24 September 2011

DEAKIN UNIVERSITY

CANDIDATE DECLARATION

I certify that the thesis entitled

Broker-based Service-oriented Content Adaptation Framework

submitted for the degree of

Doctor of Philosophy

is the result of my own work and that where reference is made to the work of others, due acknowledgment is given.

I also certify that any material in the thesis which has been accepted for a degree or diploma by any university or institution is identified in the text.

'I certify that I am the student named below and that the information provided in the form is correct'

Full Name: Mohd Farhan Md Fudzee

Signed : Signature Redacted by Library .....

Date : 24 September 2011

Abstract

Electronic documents are becoming increasingly rich in content and varied in format and structure. At the same time, user preferences vary towards the contents and their devices are getting increasingly varied in capabilities. This mismatch between rich contents and user preferences along with the end device capability presents a challenge in providing ubiquitous access to these contents. Content adaptation is primarily used to bridge the mismatch by providing users with contents that is tailored to the given contexts e.g., device capability, preferences, or network bandwidth. Existing content adaptation systems employing these approaches such as client-side, server-side or proxy-side adaptation, operate in isolation, often encounter limited adaptation functionality, get overload if too many concurrent users and open to single point of failure, thus limiting the scope and scale of their services. To move beyond these shortcomings, this thesis establishes the basis for developing content adaptation solutions that are efficient and scalable. It presents a framework to enable content adaptation to be consumed as Web services provided by third-party service providers, which is termed as "service-oriented content adaptation". Towards this perspective, this thesis addresses five key issues – *how to enable content adaptation as services* (service-oriented framework); *how to locate services in the network* (service discovery protocol); *how to select best possible services* (path determination); *how to provide quality assurance* (service level agreement (SLA) framework); and *how to negotiate quality of service* (QoS negotiation). Specifically, we have: (i) identified the key research challenges for service-oriented content adaptation, along with a systematic understanding of the content adaptation research spectrum, captured in a taxonomy of content adaptation systems; (ii) developed an architectural framework that provides the basis for enabling content adaptation as Web services, providing the facilities to serve clients' content adaptation requests through the client-side brokering; (iii) developed a service discovery protocol, by taking into account the searching space, searching time, match type of the services and physical location of the service providers; (iv) developed a mechanism to choose the best possible combination of services to serve a given content adaptation request, considering QoS levels offered; (v) developed an architectural framework that provides the basis for managing quality through the conceptualization of service level agreement; and (vi) introduced a strategy for QoS negotiation between multiple brokers and service providers, by taking into account the incoming requests and server utilization and, thus requiring the basis of determining serving priority and negotiating new QoS levels. The performance of the proposed solutions are compared with other competitive solutions and shown to be substantially better.

Acknowledgements

I would like to express my heartiest gratefulness to God for His divine blessings, which has made it possible to complete this thesis successfully and for giving me the opportunity to work under the supervision of Professor Jemal Abawajy, which always provides innovative inputs, encouragement and constructive comments on my research, as well as guidance for publication. Without his supervision, this thesis would not have been completed. He always shares his experience not only on research but also about life, parenting, work and networking, among many others. Also, I would like to thank Professor Mustafa Mat Deris as the co-author for some of the papers.

I am thankful to all the past and present members of the *NCSS* (Networked Computing and System Security) research group. In particular, I thank Soon, Harinda, Shivali, Hairulnizam, Izuan, Masitah, Isredza, Nicholas, Davood, Ammar, Ahad, Raja, Fehad, Lucas, Aiden and Izzatdin for their help and comments on my work. I also had great time with them outside research. My pray is for our success and I am looking forward for future collaboration.

Many colleagues and researchers helped during my candidature by providing advice and incisive comments, and I am grateful to them. I also extend my gratitude to the staff from the Deakin's School of IT and International Office, and Universiti Tun Hussein Onn Malaysia for their support during my candidature. I am thankful to the thesis examiners for their feedbacks to improve the thesis content, structure and readability.

I acknowledge Malaysian Federal Government, Malaysian Ministry of Higher Education, Universiti Tun Hussein Onn Malaysia and Deakin University for providing scholarships and travel supports to pursue doctoral studies and attend international conferences.

I wish to give heartful thanks to my parents (Rohana and Md Fudzee; Suhaidah and Khalid), siblings (Farrah, Faiz, Farhana, Mariam, Luqman, Anisah, Nabielah, Busyra, Fatimah, Hakimi, Zarif, Syahnaz and Iqbal), relatives and friends (here and back there in Malaysia) for their help, support, dedication and love at all times. Without them, it would not be possible to come through the various stages of this life.

Finally, I sincerely express my heartiest gratefulness to God for giving me such a wonderful wife (Ruqayyah Khalid) to live with. Her inspiration, warmth, support and patience in educating our children (Amirul Hakim, Ainul Mardhiah, Ahmad Iskandar and Arfa Saida), organizing our daily routine and enriching our life are truly appreciated.

Mohd Farhan Md Fudzee
Geelong, Australia
September 2011

Publications

2012:

- M-F. Md-Fudzee and J. Abawajy (2012). Management of Service level Agreement for Service-oriented Content Adaptation. In Internet and Distributed Computing Advancements: Theoretical Frameworks and Practical Solution, Jemal Abawajy, Mukaddim Pathan, Al-Sakib Khan, Mustafizur Rahman and Mustafa Deris (eds.). pp. 21-42. IGI-Global: USA.
- M-F. Md-Fudzee and J. Abawajy (2012). A Service Discovery Protocol for Content Adaptation Systems. IEEE Transaction on Computers (submitted).

2011:

- M-F. Md-Fudzee and J. Abawajy (2011). QoS-based Adaptation Service Selection Broker. Future Generation Computer Systems 27 (3), pp. 256-264. Elsevier BV: Netherlands.
- M-F. Md-Fudzee and J. Abawajy (2011). A Protocol for Discovering Content Adaptation Services. Lecture Notes on Computer Science 7017, pp. 235-244. Springer: Heidelberg.
- M-F. Md-Fudzee and J. Abawajy (2011). A Service Discovery Protocol for Content Adaptation Systems. TR C11/05. School of IT: Deakin University, Australia.
- M-F. Md-Fudzee and J. Abawajy (2011). A Strategy of Negotiating QoS for Content Adaptation Services. TR C11/07. School of IT: Deakin University, Australia.

2010:

- M-F. Md-Fudzee, J. Abawajy and M. Deris (2010). Multi-criteria Content Adaptation Service Selection Broker. Proceedings of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 721-726. IEEE Computer Society: New York.
- M-F. Md-Fudzee and J. Abawajy (2010). Request-driven Cross-media Content Adaptation Technique. In Developing Advanced Web Services through P2P Computing and Autonomous Agents: Trends and Innovation, Khaled Ragab, Tarek Helmy and Aboul-Ella Hassanien (eds.), pp. 91-113. IGI-Global: USA.
- M-F. Md-Fudzee, J. Abawajy and M. Deris (2010). Service Discovery for Service-oriented Content Adaptation. Proceedings of the 7th International ICST Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, pp. 1-2. Springer: Heidelberg.
- M-F. Md-Fudzee and J. Abawajy (2010). On the Design and Evaluation of Adaptation Service Discovery Approach. TR C10/04. School of IT: Deakin University, Australia.

2009:

- M-F. Md-Fudzee and J. Abawajy (2009). Modeling Request-driven Cross-media Content Adaptation using Service Oriented Architecture. TR C09/01. School of IT: Deakin University, Australia.
- M-F. Md-Fudzee and J. Abawajy (2009). On the Design and Evaluation of a Novel Path Determination Technique for Content Adaptation. TR C09/02. School of IT: Deakin University, Australia.

2008:

- M-F. Md-Fudzee and J. Abawajy (2008). Classification of Content Adaptation System. Proceedings of 10th International Conference on Information Integration and Web-based Application and Services, pp. 426-429. ACM Press: New York.

Table of Contents

Introduction 1
1.1 Motivation and Scope 2
1.2 Research Significance 3
1.3 Research Problems 3
1.4 Research Objectives 4
1.5 Methodology 5
1.6 Research Contributions 5
1.7 Thesis Organization 6
Content Adaptation Systems 9
2.1 Introduction 9
2.2 Background 10
2.3 Existing Content Adaptation Systems 15
2.3.1 Representative Content Adaptation Systems 15
2.3.2 Existing Taxonomies 19
2.4 A Taxonomy of Content Adaptation Systems 20
2.4.1 Design Themes 20
2.4.2 Content Adaptation Strategies 22
2.4.3 Content Adaptation Components 29
2.4.4 Extended Service Oriented Architecture (SOA) 39
2.5 Mapping Taxonomy to Existing Systems 40
2.6 Positioning the Thesis 44
2.7 Summary 45
SOCA Framework 47
3.1 Introduction 47
3.2 Service-oriented Content Adaptation (SOCA) 49
3.2.1 Architecture 49
3.2.2 Interaction Protocol 51
3.2.3 Request Management Components 53
3.2.4 Content Adaptation as Web Services 65
3.3 Analysis of Framework 68

3.4 Summary 70
Service Discovery Protocol 71
4.1 Introduction 71
4.2 Models 72
4.2.1 System Model 72
4.2.2 Content Adaptation Request 73
4.2.3 Matching Request and Service 74
4.2.4 Problem Statement 76
4.2.5 Performance Metrics 76
4.3 Service Discovery Protocol 78
4.3.1 Architecture 78
4.3.2 Analysis of Protocol 85
4.4 Performance Evaluation 88
4.5 Results and Discussion 90
4.6 Summary 96
Path Determination 99
5.1 Introduction 99
5.1.1 Problem Statement 100
5.2 Adaptation Path Determination Policy (APDC) 101
5.2.1 Constructing Adaptation Path 102
5.2.2 Calculate Aggregate Score 104
5.2.3 Service Selection 107
5.2.4 Analysis of Algorithm 108
5.3 Performance Evaluation 109
5.4 Results and Discussion 112
5.4.1 Single Optimal Path Generation 112
5.4.2 APDC's Execution Analysis 115
5.5 Summary 116
Service Level Agreement 119
6.1 Introduction 119
6.2 Requirement for SLA 120
6.2.1 Quality Views 121
6.2.2 Motivational Example 121

6.2.3 Problem Statement 123
6.3 SLA Framework 124
6.3.1 Architectural Framework 124
6.4 Negotiation Strategy 135
6.4.1 Proposed Strategy 138
6.4.2 Analysis of Strategy 144
6.5 Performance Evaluation 150
6.6 Results and Discussion 152
6.6.1 SLA Settlement Rate 152
6.6.2 Rejection Rate 154
6.6.3 Potential SLA Violation 154
6.8 Summary 156
Conclusion and Future Directions 157
7.1 Conclusion 157
7.2 Future Directions 160
7.2.1 Solutions in Multimedia Content Distribution 160
7.2.2 Service Outsourcing 161
7.2.3 Web Design Requirements 161
7.2.4 Non-technical Issues 162
7.2.5 Scalability of ADTE 162
7.2.6 SLA Monitoring and Enforcement 162
7.2.7 Energy-Aware Routing 163
7.2.8 Failure Detection and Recovery 163
7.2.9 Trust Issues 164
References 165

List of Figures

Figure 1.1: Non technical view of content adaptation issue…..……....2

Figure 2.1: Typical content adaptation layers……11
Figure 2.2: Content adaptation abstract model……...13
Figure 2.3: Content adaptation design themes taxonomy……...20
Figure 2.4: Content adaptation strategies taxonomy……...23
Figure 2.5: Classification of content types……25
Figure 2.6: Classification of content structure models……...27
Figure 2.7: Content structure illustration……28
Figure 2.8: Content adaptation contexts taxonomy……....29
Figure 2.9: Contexts monitoring taxonomy……32
Figure 2.10: ADTE taxonomy…….……...32
Figure 2.11: Service discovery taxonomy……....33
Figure 2.12: Path determination taxonomy……36
Figure 2.13: [13]'s score computation illustration...…….……37
Figure 2.14: Extended SOA [161]……40

Figure 3.1: Service-oriented content adaptation framework……..49
Figure 3.2: Content adaptation sequence diagram……52
Figure 3.3: Required tasks precedence graph……56
Figure 3.4: Service discovery component.....…...57
Figure 3.5: Possible adaptation paths……59
Figure 3.6: (a) positive monotonic QoS (b) negative monotonic QoS……60
Figure 3.7: Example of paths' score tree……61
Figure 3.8: (a) OCM (b) SCM……62
Figure 3.9: (a) Combining SCM with OCM (b) Combining OCM with SCM……...62
Figure 3.10: SLA management framework for service-oriented content adaptation…..64
Figure 3.11: Local proxies-Web Services interaction model……..66
Figure 3.12: Port binding operation……67
Figure 3.13: Simultaneous service's proxies' interaction for content passing……...67
Figure 3.14: An example of a translation adaptation service's definition using WSDL.68

Figure 4.1: Service discovery architecture for service-oriented content adaptation…...78
Figure 4.2: The proposed service discovery protocol……..80
Figure 4.3: Service discovery algorithm……..81
Figure 4.4: Update $\bar{S}$ algorithm……...83
Figure 4.5: Algorithm discoverability towards task variations……...91
Figure 4.6: Algorithm discoverability towards service provider variations……....92
Figure 4.7: Algorithm discoverability towards QoS variations……...93
Figure 4.8: Algorithm discoverability towards client variations……94
Figure 4.9: Algorithm discoverability towards registry variations……..95

Figure 5.1: Adaptation path determination scenario……101
Figure 5.2: APDC algorithm……102
Figure 5.3: Paths' score tree generation……104
Figure 5.4: Paths' score tree with aggregate score……106
Figure 5.5: Content adaptation control information.....107
Figure 5.6: Single Optimal Path Generation towards QoS……...113
Figure 5.7: Single Optimal Path Generation towards Tasks……114
Figure 5.8: Single Optimal Path Generation towards Service Providers……..115

Figure 6.1: Execution against varied sizes for desktop and mobile devices……123
Figure 6.2: SLA framework for service-oriented content adaptation……..125
Figure 6.3: Negotiation in creation phase……..128
Figure 6.4: Monitoring phase……132
Figure 6.5: Enforcement phase……..133
Figure 6.6: Negotiation context……136
Figure 6.7: Service provider modelled as a M/G/1 queue……138
Figure 6.8: SLA evaluator algorithm……142
Figure 6.9: Advertised *E(W)* versus new *E(W)* to be offered based on current load…143
Figure 6.10: SLA settlement rate versus number of requests (server load = 0.6)…….152
Figure 6.11: SLA settlement rate versus server load variations (λ = 60)……..153
Figure 6.12: Request rejection versus server load variations (λ = 60)……..154
Figure 6.13: Potential SLA violation versus server load variations (λ = 60)……155

List of Tables

Table 2.1: List of HTTP request headers……30
Table 2.2: Mapping representative centralized systems with the taxonomy……...41
Table 2.3: Mapping representative decentralized systems to the taxonomy……..…….43

Table 3.1: Example of service registry……51
Table 3.2: Examples of criteria categorization……60

Table 4.1: Example of content adaptation types……74
Table 4.2: List of notations……..76
Table 4.3: Example of result returned by the proposed protocol……84

Table 5.1: List of notations……101
Table 5.2: Examples of QoS categorization……..105
Table 5.3: List of workloads……..110
Table 5.4: List of QoS used……...112
Table 5.5: APDC average service selection improvement ratio……116

Table 6.1: Example of common QoS for a service……126
Table 6.2: Example of adaptation functions including objective QoS and SLA……..126
Table 6.3: Example of an SLA tags property……129
Table 6.4: List of commonly used notation……...130
Table 6.5: Example of non-compliance attributes……134
Table 6.6: List of commonly used notation……...139
Table 6.7: Characteristics of different priority models……..141
Table 6.8: Simulation settings……....151

Chapter 1

Introduction

The rapid development of digital media technologies has enabled the emergence of novel media content types for various domains including e-Commerce, e-Education, and e-entertainment. As a result, there is a phenomenal growth in consumable electronic information on subjects such as entertainment, security, education, and technical documentation targeted to diverse users in the form of content and services.

While online documents are becoming increasingly rich in content and varied in format and style, the original content is normally developed for a specific platform and is naturally made-up of media objects of different types with complicated structure and layout [1]. For instance, most of existing Web content is originally designed for desktop displays. At the same time, client devices are getting increasingly varied in their capabilities (e.g., processing power, input and output facilities). Therefore, direct content delivery to handheld devices without layout adjustment often leads to disorganization of information [2]. Moreover, as depicted in figure 1.1, not every handheld device can play all media types. For example, a non-multimedia mobile phone cannot play continuous video clips, while only H.264, MPEG-4 and M-JPEG formats are currently supported for iPhone video playback. As such, some widely employed video formats such as MKV and FLV will require format conversion or additional player before they can be played on iPhone.

Although content providers are under constant pressure to make content available in a variety of formats and for a variety of purposes [3], the mismatch between rich contents and the end devices capability coupled with specific users preferences

continues to present a challenge in providing seamless and ubiquitous device-independent access to the online electronic contents to interested users. It becomes apparent that a mechanism for dynamically transforming the original content to suite the end device and user's preference as appropriate is required.

Figure 1.1: Non technical view of content adaptation issue.

1.1 Motivation and Scope

Content adaptation has emerged as a potential mechanism to address some of the problems arising from the content-device mismatch [2, 4, 5, 6, 7, 8, 9, 10, 11]. Although many content adaptation approaches have been proposed, most of them tend to be fully or partially centralized. Problems with centralized adaptation scheme such as scalability and single-point failure are well known [12]. In order to address these problems, the idea of establishing content adaptation as a service that allows the use of a large number of adaptation mechanisms located in many places in the network has recently been advocated [11, 13, 14, 15]. Thus, the in-depth exploration of service-oriented architecture for content adaptation together with the enabling mechanisms is required to provide flexible and scalable content adaptation services. Unfortunately, analysis of the previous research efforts [5, 14, 16, 18] in this context reveals that there has been only a

few rudimentary frameworks exist. Also, in order to provide an efficient service-oriented content adaptation system, enabling mechanisms such as service discovery, path determination and service quality assurance are essential. However, these mechanisms have not been fully explored. The reason for this lack of progress is due to the complexity of the technological problems that need to be addressed in the practical context.

1.2 Research Significance

The content adaptation challenge is how to make the original contents readily available on a wide range of access devices for interested clients. One way to address this problem is by creating and maintaining different format of the original content suitable to the targeted access devices. However, keeping multiple copies of the original content will lead to tremendous overhead and places unwieldy burden on to the content authors.

Thus, what is required is a content adaptation system with the appropriate logic to analyse the content and all aspects of the adaptation contexts and formulate the content adaptation strategy accordingly. There are many content adaptation approaches that generate any content version from one single original version [3, 6, 7, 8, 9, 10, 16]. A request may require multiple content adaptation tasks that can lead to the requirement of multiple adaptation strategies including cross-media adaptation (e.g., media conversion, translation, summarization, and integration). None of existing standalone content adaptation systems is able to completely serve this request. Moreover, building one system that capable of providing various adaptation strategies is inefficient and costly.

On the other hand, there are many service providers offering a variety of content adaptation that can be loosely coupled. Therefore, the solution for these services is to cooperate with each other to completely serve the request that they cannot attain individually. A platform that enables such interconnection and interoperation is required. Thus, a greater scale as well as service quality can be achieved.

1.3 Research Problems

This thesis tackles the research challenges in relation to the development of scalable and efficient content adaptation solutions by enabling coordination and cooperation between

multiple service providers. While service-oriented content adaptation is appealing, the challenges in adopting it include architecting a system that analyses the required content adaptation tasks and distributes these tasks to the potential service providers. In particular, we identify and investigate the following five research issues:

- *How to enable content adaptation as services.* The platform that allows content adaptation to be performed as services by external service providers. This should include the essential mechanisms to manage client requests and service provider advertisements.
- *How to locate services in the network*. The protocol used to locate potential content adaptation service from the network. Such a protocol must take into account searching space, searching time, matching category of services and physical location of the service providers.
- *How to select best possible services.* The decision making mechanism used for choosing the best possible services to serve a request, given that multiple services can potentially perform a particular task.
- *How to provide quality assurance*. The framework used to manage service agreement between service providers and clients. It should formally specify the creation, monitoring and enforcement of such an agreement.
- *How to negotiate quality of service*. A mechanism to negotiate QoS before the agreement being settled. Service provider should ensure that the QoS they advertised is deliverable to avoid potential violation.

1.4 Research Objectives

To achieve the research aim, three main research objectives are identified and need to be fulfilled:

1. To develop the taxonomy of content adaptation systems and to determine the issue pertaining to existing content adaptation systems that have not been fully explored.
2. To design a conceptual framework for the service-oriented content adaptation based on the identified components and functions required for a complete content adaptation system.

3. To design, develop and analyse the enabling mechanisms, i.e. service discovery, path determination and service level agreement, in relation to service-oriented content adaptation.

1.5 Methodology

The proposed work will be carried out based on the experimental computer science method [17]. This method examines the research work to demonstrate two important concepts: proof-of-concept and proof-of-performance.

To demonstrate the proof-of-concept, some important steps were performed. First, the research area within content adaptation is critically reviewed to provide the overview that leads to the formulation of valid problem statements. From this review, the research work is justified. Then, the proposed conceptual framework of the service-oriented content adaptation architecture is designed and analytically analysed.

Proof-of-performance is demonstrated by conducting the implementation for the service discovery protocol, path determination and QoS negotiation using simulations. In those simulations, various parameters and workloads were used to examine and demonstrate the viability of the proposed solutions compared to the similar competitive solutions. Also, analytical analysis of some proposed algorithms is performed to evaluate the correctness.

1.6 Research Contributions

We detail the thesis contributions as the following:

1. *Content adaptation taxonomy*. This thesis presents a taxonomy of content adaptation systems. It investigates related concepts, describes the design themes and identifies implementation components required. The presented taxonomy is mapped to representative content adaptation systems to demonstrate its applicability. Also, the mapping assists to perform a gap analysis in this research field.
2. *Broker-based service-oriented content adaptation framework*. The thesis introduces an architectural model for service-oriented content adaptation. It describes the essential components, interaction sequences, and related protocols

for enabling content adaptation as services. An analytical analysis is conducted to demonstrate the framework applicability.

3. *Service discovery protocol*. The thesis investigates and presents the protocol to locate available content adaptation services from the network. Along with the derivation of the discoverability performance metric, extensive simulations have been conducted to study the performance of the service discovery protocol in this regards. The proposed protocol is able to quickly terminate the search when specified conditions are achieved. Also, the analytical analysis proved the completeness and the accuracy of the protocol.
4. *Path determination mechanism*. The thesis presents the mechanism to determine the best possible services based on the single objective assignment function. The proposed mechanism is evaluated through simulations in term of service selection execution. The mechanism is demonstrated to meet its objective i.e., appropriate service QoS value assignment.
5. *SLA framework*. This thesis introduces a framework for managing service level agreement in relation to content adaptation. It describes the interrelated phases and the essential mechanisms. Then within the framework, a QoS negotiation strategy is presented. The proposed negotiation strategy is evaluated through simulations and is shown to increase SLA settlement, and reduce request rejection and potential SLA violation as well.

To summarize, the work presented in this thesis is in line with the current trends that enable multitude content adaptation services without having to build a dedicated infrastructure [5, 14]. Therefore, it is our thesis to present service-oriented content adaptation solutions that are scalable and efficient.

1.7 Thesis Organization

The chapters of this thesis are derived from various papers published during the PhD candidature. The remainder of the thesis is organized as the following:

- *Chapter 2: Content Adaptation Systems.* This chapter provides an in-depth analysis and overview of existing content adaptation systems, presented within a comprehensive taxonomy.

- *Chapter 3: Service-oriented Content Adaptation*. This chapter presents an architecture to enable content adaptation to be consumed as services. It describes the key components to realize service-oriented content adaptation.
- *Chapter 4: Service Discovery Protocol*. This chapter presents a service discovery protocol in relation to service-oriented content adaptation. The simulation results are discussed as well.
- *Chapter 5: Path Determination*. This chapter presents a path determination mechanism in relation to service-oriented content adaptation and the related simulation results.
- *Chapter 6: Service Level Agreement*. This chapter presents a framework for managing service level agreement and QoS negotiation strategy in relation to service-oriented content adaptation. The simulation results of the negotiation strategy are discussed as well.
- *Chapter 7: Conclusion and Future Directions*. The concluding chapter provides a summary of contributions and a future research challenges.

Chapter 2

Content Adaptation Systems

The ever-increasing amount of electronic information coupled with proliferation of diverse and heterogeneous devices, data sources, user preferences and networks has significantly increased the demand of content adaptation. This makes content adaptation as a thriving research field. There are many projects focused on the content adaptation being introduced constantly. This chapter provides an in-depth analysis of current content adaptation technologies, organized as a comprehensive taxonomy. The taxonomy provides a basis for categorizing related solutions and being mapped to a few representative systems to demonstrate its applicability. Then, a "gap analysis" is performed from the presented literature and used to position the thesis.

2.1 Introduction

Today, computing is no longer limited to a specific location using desktops devices, but can be done on laptop computers and information appliances (e.g., PDAs, smart phones, etc.) from anywhere at any time. This new computing platform is known as pervasive or ubiquitous computing and has recently attracted a lot of attention. However, the characteristics of this paradigm shift (including device heterogeneity, limited device capability, and user's high mobility) bring about new challenges in the delivery of information, content and services in these environments. This makes the ability to adapt information, content and services to a diversity of computing devices a key to pervasive computing.

Specifically, devices, standards and software develop rapidly, but still often independently of each other [15]. This creates problems in terms of content suitability. Also, in pervasive environment, user and system-level applications must execute subject to a variety of resource constraints that generally can be ignored in modern desktop environments. Moreover, Web applications are designed with desktop platform in mind that usually contains rich media content and authored in a single version. In order to increase the usability of mobile Internet services, content adaptation is required. Also, the emergence of these requirements (e.g., device heterogeneity, user preferences, rich content) demands efficient content adaptation architecture. Designing such an architecture that will meet these requirements is challenging due to several issues: (a) supporting scalability, (b) meeting computational constraints, and (c) enhancing adapted content quality.

In this chapter, we present the literature of the content adaptation field. The research field of content adaptation have been growing rapidly during the past ten years and this has resulted in a plethora of new concepts, models and systems. An abstract architecture for a content adaptation system that succinctly captures the essential components and functions of a content adaptation system is presented. The significance of the different components and functions of the model are also discussed. A taxonomy that classifies the approaches that form the design space and implementation requirements of content adaptation systems is presented. The applicability of the taxonomy is demonstrated by mapping representative existing systems. Also, this taxonomy is used to perform "gap analysis" by revealing some of the areas that are yet to be fully explored that can lead to creative solutions.

2.2 Background

In this section, we present a generic content adaptation architecture that outlines the different components of content adaptation system architecture. The architecture is important to provide a central knowledge regarding the architecture (i.e., components, functions) choices made by existing content adaptation systems. To keep the model compact, only the core functions of the content adaptation systems are included. The essential definitions for content adaptation are also presented.

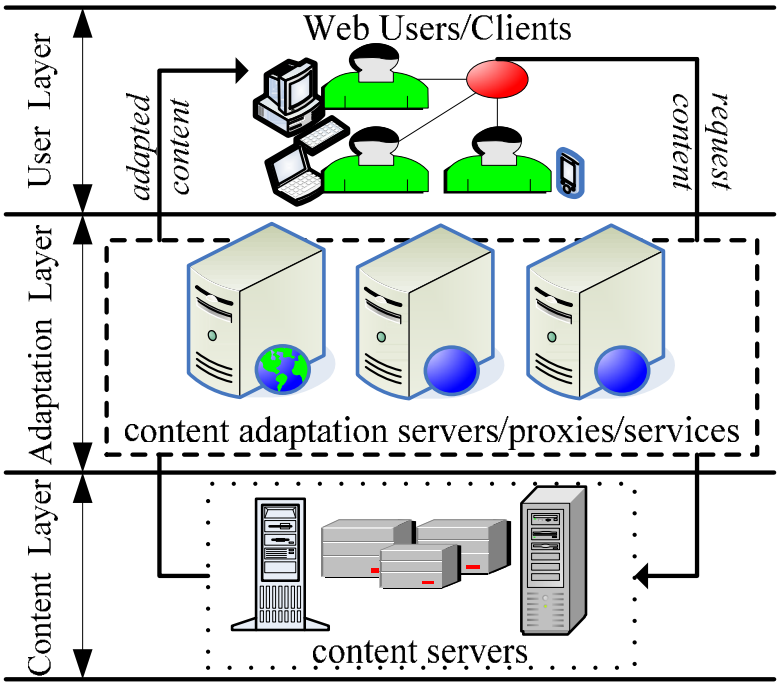

Figure 2.1: Typical content adaptation framework.

Figure 2.1 shows the outline of content adaptation extracted from [2, 6, 13, 19, 20]. Generally, a typical content adaptation contains three layers: user, adaptation and content layers. At the user layer, user/client requests for the Web content from content servers via different devices. The content servers are grouped at the content layer and located in many places across the network. At adaptation layer, original Web content is tailored to meet the contexts (e.g., device's constraints, preferences) of each targeted user determined by adaptation decision. This tailoring process can be performed by the adaptation mechanism(s) at a single or several different locations (e.g., content servers, proxies). Finally, the adapted version of the Web content is delivered to the users.

In content adaptation, several essential terms are defined as the following:

Content adaptation is a term that defines the tailoring, aligning or customizing content into a required version [2, 6]. It is performed to tailor with the adaptation contexts.

Context is the circumstances surrounding an entity or event [21]. This includes any information that can characterize an entity's situation or state.

Context is motivated by this key question: “to adapt the content to what”. It could be a device, network, user/client or combination of them.

Client is a Web user that consumed content adaptation services to get the required content version [12]. Clients use these services directly or through a service broker.

A system must exist with the appropriate logic to analyse the content with all aspects of the contexts and formulate the content adaptation strategy that will deliver a version required by the client. Clients can benefit from the expansion of cross-media adaptation strategies (e.g., media conversion, translation, summarization and integration) provided by third-party service providers. This opportunity has attracted both academic and business communities (e.g., Web services) [15].

In general, a content adaptation system is made of several core components. Some additional components are required for decentralized/distributed architecture. Figure 2.2 shows an abstract model of a content adaptation system. This model is developed by considering existing systems surveyed. The model contains four components and divided into two major blocks: common components and distributed components. Each component has specific function. The abstract model complies with the work presented in [22, 23, 24].

The common block contains two key components: contexts gathering and adaptation decision-taking engine (ADTE). Contexts gathering function is to collect necessary data/information (e.g., network profile, device profile, user preferences) including the content metadata from the particular entity to be considered for adaptation and mapping them into the semantic representation. The content metadata (e.g., Web structure, page dimension, number of objects, links) is collected from the content server. This metadata is produced in a process called content parser [24]. Network profile (e.g., UMTS/GPRS/GSM Data) is gathered on-demand as it is hard to determine the user network environment in advance and can be gathered using a particular network monitoring tool. Client profile (i.e., device profile and user preferences) could be fetched from independent client registry. This registry can be maintained at client profile server and is updated periodically. For instance, client profile can be represented according to the composite capabilities/preference profile (CC/PP) specification

introduced by World Wide Web Consortium (W3C). This profile can be detected through Bluetooth or ZigBee configuration, if activated. Another important function in the context gathering component is contexts (including resources) monitoring [25]. In fact, to ensure content adaptation can be carried out accordingly, context monitoring needs to be measured accurately and efficiently.

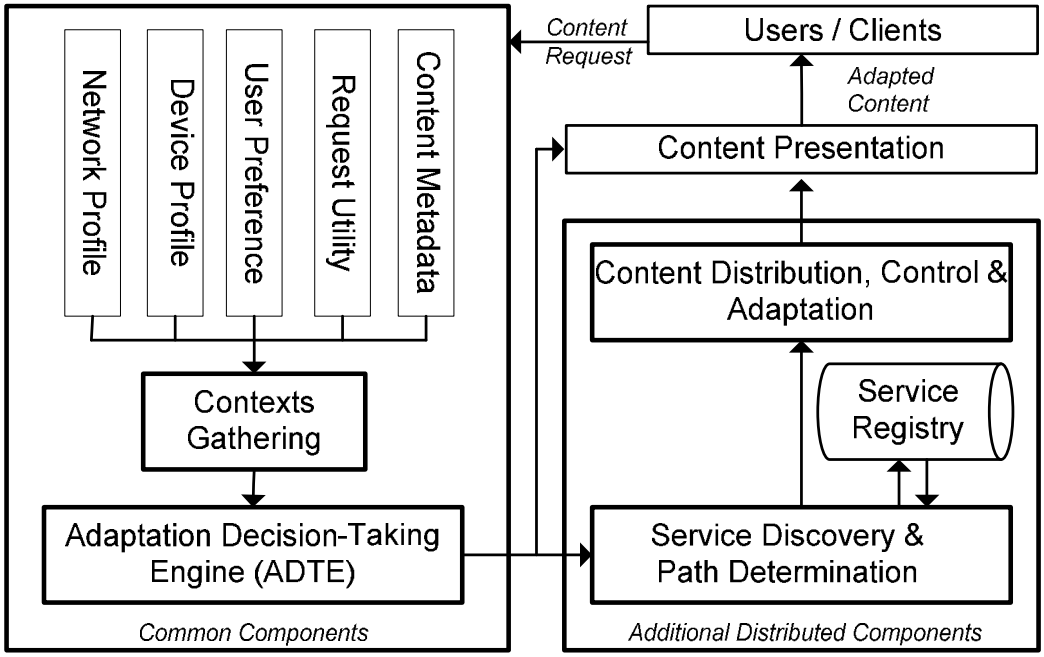

Figure 2.2: Content adaptation abstract model.

These contexts are sent to ADTE for processing. ADTE analyses these contexts with the content metadata to produce adaptation decision for obtaining the required content version. This decision determines the content adaptation tasks/strategies. Example of the decision from ADTE is adaptation information (e.g., media modality value, media fidelity value, number of column, etc). For centralized content adaptation systems, this decision is use by the local adaptation engine to adapt the content accordingly. That is, the ADTE and the adaptation engine is combined as one component and located at the same location. Meanwhile, for distributed content adaptation, the adaptation tasks are distributed to several adaptation proxies across the Internet [13, 14, 15]. Currently, there are three kinds of ADTE model: probability-based model, rule-based model and optimization-based model.

The second block contains two key components for distributed architecture: path determination that includes service discovery and content distribution management. The primary objective of the path determination component is to decide who should perform the adaptation tasks. Establishing content adaptation as a service allows the use of a large number of adaptation mechanisms located in many places in the network thus, a task can be performed by multiple services. To benefit form these services, clients must be able to locate them in the network. This makes service discovery an important component. An efficient service discovery mechanism is essential for the success of the distributed content adaptation systems [26]. Also, selecting appropriate services among the many located services is necessary to increase the overall performance of the system [13].

Content distribution management is required to manage the distribution of content, adaptation tasks (including control information) between clients, local proxy/broker and adaptation service proxies. Via this component, adaptation tasks together with the content segments are distributed to several services to be adapted and a proper control mechanism is imposed to ensure each segment is adapted accordingly. Efficient and secure content distribution between cooperative intermediaries has been discussed in [27]. A content distribution mechanism differs for a content delivery system such as [28] in which the former requires content to be modified by the service proxies along the path. On the other hand, content delivery only deal to provide client with the original content requested from the origin content server through replicated servers across the Internet. A fault tolerance mechanism can assist content distribution to recover failed service(s) [1].

Content distribution can be managed using two approaches: centralized (star-based topology) and decentralized (mesh-based topology). In centralized approach, the service providers must communicate through the local proxy or broker to get and/or to deliver content, while decentralized approach allows direct communication between service providers. As a result, centralized approach suffers additional overhead while decentralized approach requires efficient distribution monitoring. Studies in [29, 1] prove that the decentralized approach performs substantially better than centralized in distributed content adaptation.

2.3 Existing Content Adaptation Systems

In this section, we will provide some descriptions of the existing content adaptation system ranging from the late 90's to the recent years. The example systems surveyed are not exhaustive, but comprehensive enough to cover many of the classes in our taxonomy.

2.3.1 Representative Content Adaptation Systems

The surveyed systems include both centralized and decentralized such as InfoPyramid, Power Browser, PDCAS, VTP, XAdaptor, PACER, ADAPT^2, DCAF, CAF, SCAP, CAIN and PUMA. In the following, we provide some description of each system.

InfoPyramid - InfoPyramid [2] is a centralized proxy-based browsing adaptation system that adapts multimedia Web documents to optimally match the targeted device. It is structured into two components: a multimodal content representation and a customizer that selects the best content representation to tailor device context based on the optimization strategy. In the first component, the content items on a Web page are transcoded into multiple resolution and modality version, before actually analysing the targeted device – static adaptation is implemented. The proxy is responsible for the context monitoring. Appearance, size and format are the supported adaptation strategies.

Power Browser - Power Browser [30] is a centralized proxy-based browsing adaptation system that adapts text display to suit mobile devices. The adaptation contexts are the device and user preference, and being monitored by the proxy. It breaks the Web page into text units that can be easily displayed, hidden or summarized. Each text unit is represented by a keyword. For mobile screen, Power Browser displays this keyword list rather than the whole text units. The full text unit will be displayed if clicked by the user. Four main components are form processor, keyword extractor, sentence ranking and summary generator. Device context is fetched from the profile database. Navigational, appearance and encapsulation are the supported adaptation strategies.

PDCAS - PDCAS [6, 31] enables documents adaptation based on five quality domains: color, downloading time, scaling, modality and segment. These domains correspond to the user input collected in pre-processing phase. This information, together with

contexts gathered in real time, is used in score node selection algorithm to produce desired content. The ADTE adopts optimization strategy. Device, network and user preferences are the contexts monitored by the proxy. It is a dynamic media adaptation system that adopted proxy-based architecture. Appearance, size and format are the supported adaptation strategies.

VTP - Versatile transcoding proxy (VTP) [9] is a centralized proxy-based browsing adaptation system that can accept and execute transcoding preference script provided by the client to transform the corresponding content accordingly. That is, it can deal with multimedia content as long as the transcoding preference is supplied. In addition, a specific transcoding scheme is used to maintain cache objects and perform cache replacement. The adaptation contexts are device and user preferences, which are monitored by the proxy or a service agent. Weighted transcoding graph is used to dynamically select the suitable version. Appearance, size and format are the supported adaptation strategies.

XAdaptor - XAdaptor [8] is an extensible proxy-based browsing adaptation system that classifies page objects into structure, content and pointers objects. The key idea is to adapt based on structure object HTML table. Rule-based strategy is adopted for the ADTE and to provide extensibility. Device and user preferences are the contexts monitored by the proxy. Appearance, size and format are the supported adaptation strategies.

PACER - PACER [32, 33] adapts online educational resources to suit the targeted user with different learning style: personalization. In addition, PACER takes into account not just the interests, but also the current knowledge and goals of their users. This is a dynamic server-based system that can provide adaptive navigation support for browsing-based access to open corpus resources and support information access through adaptive information visualization. The context is monitored by the proxy. The ADTE applies a rule-based strategy. Appearance and navigation are the supported adaptation strategies.

ADAPT^2 - A similar content adaptation system to PACER is ADAPT^2. The exceptional is the architecture design. It is aimed at providing personalization and adaptation services for developers of otherwise not personalized content [35]. The system's components (e.g., user modelling server, ontology server, value-added service and content server) are designed using distributed architecture. Appearance and navigation are the supported adaptation strategies.

DCAF - Distributed content adaptation framework (DCAF) presented in [1, 5] is a service-oriented media adaptation. It enables adaptation tasks to be performed by a third party. In this system, content adaptation is performed in several steps. The ADTE is not specifically discussed. The path determination is performed using a greedy single objective assignment function. The path associated with highest score is selected as the optimal path. Device, network, user preferences are the contexts monitored by the local proxy. A continuity of this system is ConAMi [35]. It is customized version to support content adaptation for Mobile Ad hoc NETwork (MANET). Size, format, encapsulation, media conversion and translation are the supported adaptation strategies.

CAF - Co-browsing adaptation framework (CAF) presented in [36] is a partially static proxy-based browsing adaptation system. It implements rule-based strategy to tailor content with the device context. The original content is adapted into co-browsing version in order to support co-browsing activity between devices with different capabilities. It is partially static adaptation because the default co-browsing content is adapted before the device context is monitored by the proxy, but not during the authoring time. Appearance, size and format are the supported adaptation strategies.

SCAP - SCAP is a centralized proxy-based solution [16]. Its primary objective is to provide mobile users with adaptive content without direct user input and to provide value-added content. Value-added content is achieved by creating composite content, i.e., best possible presentation of content to the user's device together with additional content as a result of capitalizing request's information. For instance, if a page containing a movie sound track is requested, SCAP will suggest and cross-sell other related products such as video clips and movie trailers. SCAP captures each device

capability (i.e., screen, display and supported media format) from available device capability server. Its ADTE renders a content version suitable to be presented at requesting device by analysing the content metadata to match the device capability. It integrates value added content by performing any of these three methods: content-to-content correlation, attribute-based and collaborative filtering. This value-added content will be presented tailored to each requesting device as well.

CAIN - In this system [37, 38], the content adaptation manager provides meta-data driven content adaptation. It allows multiple content adaptation tools (termed as CATs) to be added in the system. The primary operation of CAIN is to increase user's experience in browsing content by analysing user's context (i.e., terminal capabilities and network characteristic) with the content. CAIN collects the descriptions of both content and user's context. Enabling the addition of new CATs enables a wide range of content adaptation strategies such as transcoding, summation and transmoding. Content and user's context are described using MPEG-7 MDS/MPEG-21 BSD and MPEG-21 DIA, respectively. These inputs are sent to the decision module to decide which CAT is required to provide the best adapted content. CAIN is a centralized proxy based model. The ADTE treated content adaptation as content satisfaction problem by searching for CAT that matched more constraints and the best option. It also provides an optimization method to select the best CAT if more than one available by specifically compare each constraint.

PUMA - It is a service-oriented system that distributes content adaptation to several services along the network before the final adapted content reach the user [14, 39]. It is made of four components: workflow preparation, validation, instantiation and monitoring. Workflow outlines the adaptation steps in sequences; validation component validate the interoperability of the services; instantiation component invoked the services; and monitoring component monitoring the service execution and replaces failing service(s). PUMA's decision engine gets the request with the content preferences and technical constraints through MPEG-21 DIA. It uses Pareto Preference Graph to choose the optimal adaptation (the first option is the best possible option) in a manner similar to constraints satisfaction. PUMA demonstrated that content adaptation under

real-time constraint is possible in a service-oriented way. It has the capability to recover from failures of individual services [40]. PUMA's service discovery and service selection is based on functional aspect and cost offered [41].

In the next subsection, we review some existing taxonomies pertaining to adaptation or content adaptation.

2.3.2 Existing Taxonomies

Content adaptation research area emerged from the idea of bridging the mismatch between requested resources (i.e., content) and the requesting device. Early researches exploited some approaches in solving general adaptation issues. Taxonomies focusing on issues related to adaptation in general are discussed [25, 42]. In [42], adaptation techniques are classified into user-centred, system-centred, and mixed adaptivity. On contrary, [25] classified adaptation techniques into laissez-faire, application-transparent and application-aware. However, both [25] and [42] emphasize on adapting resources, not specifically content per say.

Several taxonomies with specific aspects of content adaptation have been discussed in the literature. An early classification of adaptive hypermedia systems is presented in [22]. However, it only covers basic adaptation strategies that deal with layout rearrangement. Moreover, it does not consider different adaptation contexts such as device and network heterogeneity.

A taxonomy that is specifically dedicated to mobile application is discussed in [43]. Another taxonomy focusing on locality of content adaptation is presented in [5]. The taxonomy breaks content adaptation locality into two groups: centralized and decentralized. Also, a taxonomy that solely focuses on content delivery network (CDN) is presented in [28]. A recent taxonomy in [44] explores QoS issues in customizing content.

As research on content adaptation is quite extensive and the landscape is changing fast, new research issues have been raised. However, the established taxonomies no longer include many of the recent new concepts and developments. Unlike these taxonomies, we present a taxonomy that focuses on design themes and the implementation details of the recent content adaptation systems. This taxonomy is presented in the next section.

2.4 A Taxonomy of Content Adaptation Systems

In this section, the taxonomy of content adaptation is presented. The taxonomy classifies content adaptation by characterizing different components. The intent of the different components and functions is to differentiate content adaptation implementations.

2.4.1 Design Themes

The design objectives for content adaptation motivate the architecture of the content adaptation application. It is important for the application designer to identify the underlying components of content adaptation architecture. We classify the design objectives into two themes: (a) enhancing user's browsing experience and (b) enhancing scalability and media adaptability. Using these themes, content adaptation systems are placed into two categories as shown in figure 2.3.

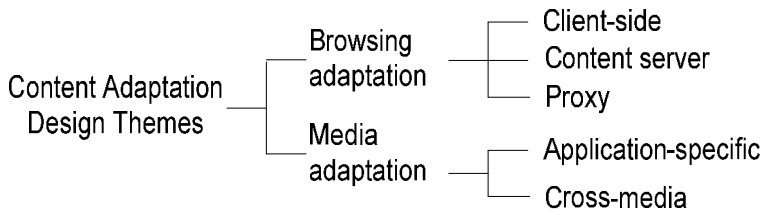

Figure 2.3: Content adaptation design themes taxonomy.

Before the design themes are elaborated, we discuss two different ways on how different content versions can be prepared. One way is by creating and maintaining different format of the original content suitable for the targeted access devices. This approach is used in InfoPyramid [2]. In this case, content is formatted differently for displays that have different capabilities, and is also delivered differently for devices that have a different connectivity [3]. Although the pre-adapted version (i.e., static adaptation) is simple to implement, it suffers from a number of serious drawbacks. To create a pre-adapted content version, a human designer can be involved to hand-tailor a version for some specific rendering requirement [3]. Keeping multiple copies of the original content will lead to tremendous overhead and places unwieldy burden on to the content authors. Moreover, any changes in the content may require changes on every version of the contents, which renders this approach error-prone. In addition, new

device may require a new format. Clearly, this is neither practical nor feasible for providers of large volumes of content.

An alternative content adaptation approach is to automatically generate any content version from one single original version such that the content is adapted to the device and the user preferences (i.e., dynamic adaptation). This requires a content adaptation system with the appropriate logic to analyse the content and all aspects of the delivery context and formulate the content adaptation strategy that will deliver the required content version. Dynamic adaptation provides suitable adapted version to each device or client and no multiple versions is created at the authoring time. [3, 6, 7, 8, 9, 10, 16] are some examples of content adaptation systems performing dynamic adaptation.

2.4.1.1 Browsing Adaptation

The browsing adaptation (also known as general purpose) category denotes systems that focus on adapting content to enhance user's browsing experience. It concerns with tailoring Web properties (e.g., layout, table, text column) and objects properties (e.g., size, format) to the diversity and heterogeneity of users devices. Most of the earlier systems such as InfoPyramid [2], Power Browser [30], and Odyssey [45] belong to this category. These systems can further be subdivided into client-side and server-side adaptation approaches.

In client-side approach, the client itself (e.g., netbook, PDA, smart phone) needs to perform the adaptation, and then send the adapted Web to the user's display. For example, after downloading the Web page requested by the user, the adaptation is performed to suit device's capabilities at the client, immediately before the adapted page viewed by the user. [36] is one of the example that used this approach. The main advantage of this approach is that the device capabilities can be determined directly. However, some requirements (at client side) of the adaptation (such as processing, encoder) may not be available and insufficient.

In server-side approach, adaptation is performed at the origin server, where the original Web page resides. For example, while the user requests to browse a particular Web content, the server automatically collects the related contexts (device's profile, user preferences, and network constraints) and adapts accordingly. [45, 46] are among the pioneer systems designed with this approach in mind. Server-side approach

performs very well for relatively small number of users, but will suffer overload if too many simultaneous requests.

Alternatively, content adaptation can be performed by a middleware i.e., proxy-based approach. [2, 8, 9] are some browsing adaptation systems that implemented this approach. Content adaptation and contexts monitoring is managed by external server called proxy. Some of these browsing category systems [2, 30, 45] implement fixed and hard-coded algorithms to easily and securely control the adaptation process, however leads to the difficulty to adapt changes especially when the new browsing requirements is introduced. More recent projects such as VTP [9] and Xadaptor [8] used scripts and agents to facilitate the server with extensibility and flexibility capabilities.

2.4.1.2 Media Adaptation

The media adaptation category is for systems that provide specific media adaptation. This category is further subdivided as application-specific and cross-media. Application-specific proxy is designed to handle a specific media adaptation. For instance, an image type adaptation proxy only caters image adaptation. In application-specific, adaptation is managed by varying fidelity; the qualities or formats of the specific media. For example, an image may have different colour scheme, format or size. The adaptation engine computes for the best version to suit the contexts. One example system is Portable Document Format Content Adaptation System-PDCAS [6, 31]. It tailors portable document format into suitable version (e.g., WBMP, WML or PDF, with 2, 16 or 256 colours) based on user preferences, device profiles and network environment.

On the other hand, cross-media is performed by transforming one media type into another [47]. For instance, the video data can be transformed into a series of images. Also, we can convert text into audio file to assist users to read text message or important email while driving.

2.4.2 Content Adaptation Strategies

Both browsing adaptation and media adaptation have to perform a particular strategy(s) to provide the user with the required content version. Specifically, content adaptation strategies are important to classify the action required to adapt the content according to the contexts. This organization differentiates the different action towards tailoring the

content to the targeted contexts. It answers the question "how to adapt the content". As shown in figure 2.4, content can be adapted using several strategies: size adaptation, appearance adaptation, format adaptation, encapsulation adaptation, summarization, translation, media conversation and navigational adaptation.

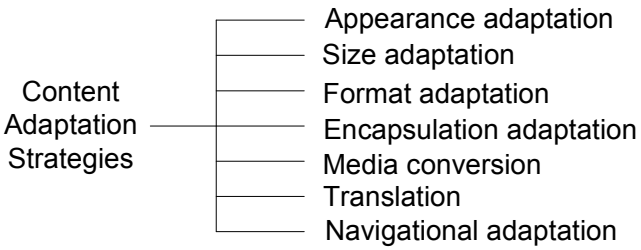

Figure 2.4: Content adaptation strategies taxonomy.

Most of the new Web sites designed with a fixed width and have a centred column where the main text resides. To read overall text, user needs to scroll horizontally and vertically. One way to deal with this is through the appearance adaptation. A real life example of appearance adaptation is the Opera's Small Screen Rendering™ technology [48]. Basically, in this strategy, only the Web layout is adjusted while preserving the content and functionality. In [10], a multi column Web is altered into a single column while preserving the content. Both examples eliminate horizontal scrolling.

Web authors often include objects such as images and video to attract users. Those objects' sizes are relatively normal for desktop viewing. But for mobile display, this is intolerable. This requires size adaptation and can be done by resizing or scaling the object's dimension. [49] proposed an attention model to adapt the image according to the particular display's size.

A media can be represented by different format. For instance, an image can be displayed using different colour scheme (black & white, 2-bits) and different format (e.g., jpeg, bitmap). Format adaptation is performed by changing incompatible content into a more suitable format in the same media. It is widely use in the area of mobile application, due to the devices diversity. For instance, considers Multimedia Messaging Service (MMS) communication over the network. Today embedded phone's camera usually provides user with high quality image. During delivery, this higher quality image should be transcoded to a lower resolution image with fewer colours in order to

better fit the targeted device. In [6, 31], the most suitable document format is generated to suit the targeted device. For example, a HTML document will be transformed to a compact format version (cHTML) that more suitable for small screen.

When a user looks for shorter version of the original content, summarization adaptation can be used. The key idea is to extract the most important aspect of the content that enough to convey the overall content. This practice is similar to executive summary in an annual report, which allows reader to grasp the gist. Through this strategy, [46] proposed Unit of Information concept to represent the Web content structure. It compromises of a set of segments and media objects to be presented together. When summarization is required, the most important content set is selected. In [30], an approach that breaks each web page into text units that can easily be displayed, hidden, visible or summarized, is presented, while [50] proposed a block-based content decomposition structure, where a HTML page is factorized into blocks with assigned scoring value. Content block with the highest score will be displayed. For a Web with a long text, instead of changing the column layout, the text itself also can be adapted by summarizing it into a shorter version.

Media conversion is concerned with adapting one content media from one type into another. It is more complicated if compared to other strategies. It can be done by converting or transforming the media type. For instance, if a client device cannot support video content, it can be transformed into a series of images. This enables some information to be conveyed to the user. Media-Convert [51] is an online service that enables audio visual media file to be converted, separated and integrated.

When dealing with language barrier, translation adaptation can be implemented. For instance, a user could request for a specific English audio file. However, only the Spanish version is currently available. Using translation, the Spanish audio can be converted into English audio. WebServiceX.Net [52] is an example of Web Services that provide language translation service.

Navigational adaptation strategy is used to guide user to access content based on their knowledge and interest. That is content is tailored to provide the user with content keyword(s) rather than the whole content segment. When the keyword is clicked, it will navigate the user to the desired content. [30, 32, 33, 34] are the examples of systems implemented this strategy.

Most of these strategies (e.g., size, format, appearance, encapsulation adaptation) perform well with well-formatted Web pages, however may not work properly with unstructured Web design.

To demonstrate the relation between content adaptation and content structure in content adaptation systems, we present the content types and content structure taxonomies.

2.4.2.1 Content Types

Content type's taxonomy presents the categories of the content. This taxonomy is important for the system designer to understand and consider what types of content could and potentially be adapted. As depicted in figure 2.5, content can be divided into four types: media content, presentation content, application data and application functionality (code). Media content can further be grouped into two: text and audio visual. Presentation content includes stylesheet, markup languages and emerging technologies. Content adaptation systems such as [2, 6, 8, 9, 30] deal with these two types.

Application data can be divided into two: application specific and XML formatted. Meanwhile, application functionality can be divided into three: standard software platform, software as content and device independent software.

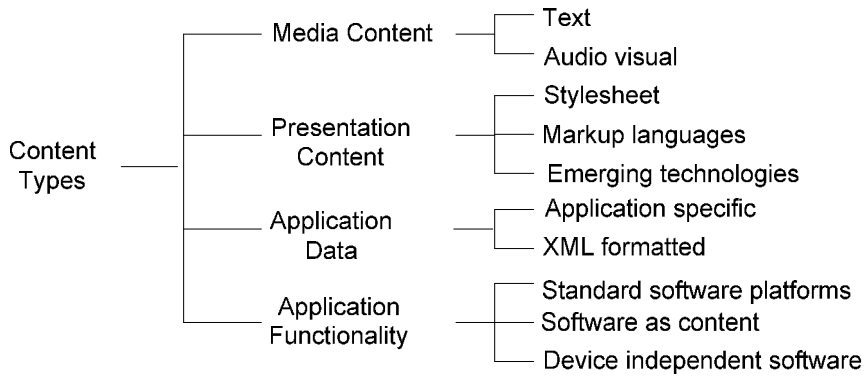

Figure 2.5: Classification of content types.

In media content type, text is usually associated with typography. Typography presents the text in several styles, font types and sizes. The term audio visual refers to

work with both sound and visual components. Multimedia content adaptation systems such as [53, 7] deal with this content type.

In presentation content type, stylesheet refers to a presentation of structured Web documents that control the visual layout (for colour, fonts etc.) using stylistic rules. CSS (Cascading StyleSheet) is an example of a widespread use stylesheet language. A markup language is an artificial language using a set of annotations to the text that gives instructions regarding the structure. It contains the Web page's semantic content and structure. HTML (HyperText Markup Language) and XHTML are some examples of markup languages. Both stylesheet and markup language complement to each other, as Web page is designed in a form of separation content and its presentation. Most applications targeted to both desktop and mobile devices need to consider this content type.

Application data can be either in application specific or XML (Extensible Markup Language) formatted. Application specific data is declared and used in a particular application and not extensible. In contrast, XML formatted is a general purpose specification for creating custom languages. As such, it allows developers to define their own elements. It is a fee-free open standard, and can help information systems share structured data, particularly via the Internet.

While in application functionality type, standard software platform refers to a set of hardware architecture and application framework that allows software to run. It contains complete suit of APIs (Application Programming Interface) and usually is dependent to the underlying operating system (such as Visual Basic) and only a few are non dependent. J2ME (Java 2 platform micro edition) is the example of independent platform that adaptable for mobile devices [54]. Finally, software is device independent when its function is universal on different types of device. One of the examples is Apache Cocoon and can be found at [55].

2.4.2.2 Content Structure

Content structure taxonomy is required to classify content into different layers. By having these layers, a possible transition of a media form can be demonstrated. In addition, this content structure model is used to specify feasible fidelity classification for each object. This structure is the central to design ADTE. That is, decision for choosing the suitable version is reflected from this structure. This taxonomy complies

with MPEG-7 MDS/MPEG-21 BSD descriptions. The content structure models can be divided into two: 4 layers and 3 layers models, as depicted in figure 2.6.

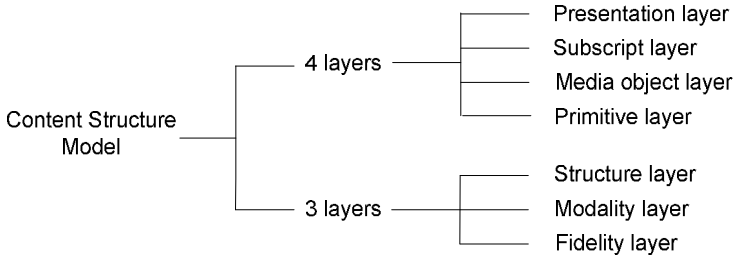

Figure 2.6: Classification of content structure models.

A four layers model that consists of presentation, subscript, media object, primitive layers is proposed in [56]. The presentation layer describes the content of the presentation in general (e.g., title, author). The subscript layer describes the presentation logic, which can be based either on the semantic or physical partitioning of the presentation (e.g., script, hypertext section) [56]. The media object layer describes the smallest visual block entities (e.g., text, video, and image). The encoded or format of the entities is defines in primitive layer. It usually defines the encoding type, colour map, and other properties in the same entity.

A three layers model is consists of structure, modality and fidelity layer that basically based on Web logical design is discussed in [10, 46]. Generally, at structure layer, page is broken into several media objects. Then, at modality layer, each possible media type for a particular object is recognized. Examples of media types are text, image, audio, video, column and animation. At fidelity layer, every related presentation format for each media type is presented. Fidelity refers to different possible forms of a particular media.

Both the four layers and the three layers models contain two layers in common: fidelity and primitive at the lowest layer; and, modality and media object located above the previous layer. The upper layers in four layers model (i.e., presentation and subscript layers) can be mapped into structure layer in three levels model. Both models should work for any Web content/page. However, three levels model is easier to be implemented due to the less number of layers.

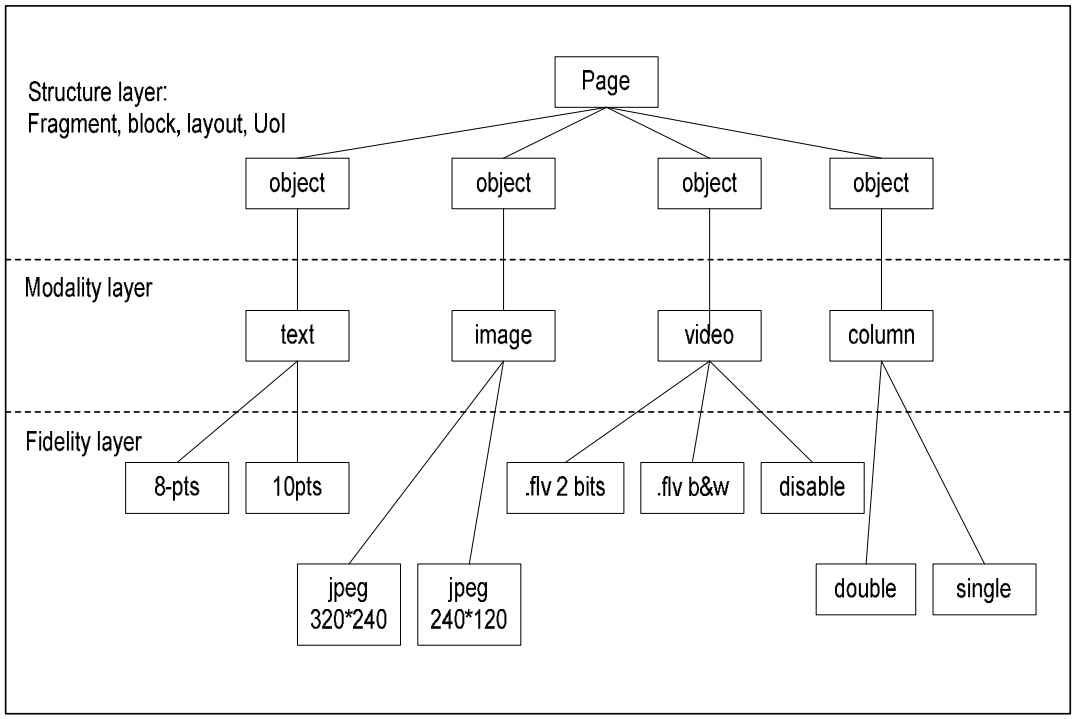

Figure 2.7: Content structure illustration.

Figure 2.7 depicted the sample of content structure mapping using three layers model. For instance, a Web page is designed to include several objects. Each object is represented by a media such as text, image video or column. Then every possible format for each media is described into detailed. For instance, text presentation media type can be provided in two sizes, 8-pts and 10-pts. An image may have different size, colour scheme or frame size. A video media can be displayed, hidden, scaled to a different size, compressed or transcoded to another format. In case of column, we might have a single or double. If we deal only with specific content, the structure layer can be ignored. This is a basic structure and can be extended to meet the specific design objective. However, for cross-media adaptation, an extended version that can illustrate the possible transition between different media types is required. From figure 2.7, ADTE could enforce a smaller font size for a smaller screen device. Also, if the text layout is hard to be read through double column, single column reading could be suggested.

2.4.3 Content Adaptation Components

In this section, we classify content adaptation by characterizing different components relevant to the content adaptation systems. The intent of having several different components is to differentiate content adaptation implementations. Content adaptation components can be classified to adaptation contexts, contexts monitoring, ADTE, path determination and adaptation strategies.

2.4.3.1 Adaptation Contexts

Context plays an important role in describing the nature of a particular entity. Context is the circumstances surrounding an entity or event that includes any relevant information that can characterize an entity's situation or state [21]. In a content adaptation system, any relevant object to content request such as device, person or place could be an entity, and information regarding the entity, i.e., device (e.g., device capability); person (e.g., user preference, situation); and place (e.g., location accessibility, network) can be the adaptation contexts. Fig. 2.8 depicts the content adaptation contexts. It can be divided into five: network, device, accessibility, personalization, and utility/facility.

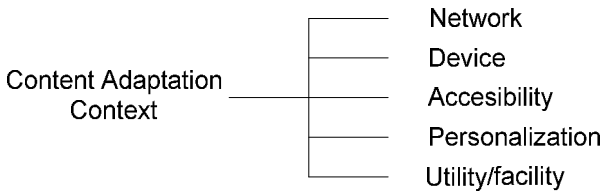

Figure 2.8: Content adaptation contexts taxonomy.

For the network context, bandwidth and round trip time (RTT) for some channels (such as Code Division Multiple Access (CDMA), General Packet Radio Service (GPRS) and Cellular Digital Packet Data) is essential to be monitored. In the case when user is moving between different environment, network context is acquired to be monitored timely. There are two adaptation strategies to deal with network context: send content sequentially based on priority, and varying the content fidelity (i.e., when passing through different network environment, the content adaptation is performed by switching the fidelity of adapted content version to keep downloading time within the allowed tolerance). For instance, [53] explores on buffer driven adaptive video streaming method that send the content layer by layer.

For device context, some of this information (refer table 2.1) is available in the Web request's standard HTTP protocol header. Servers can use this header to try guessing the client device's nature (such as the Web browser and acceptable representations). To express specific data format for delivery context, composite capabilities/preference profile (CC/PP) is used. User agent profile (UAProf) is another approach for this context. Bluetooth or ZigBee profile is also useful in passing context during the user and the content application interaction. Mobile Information Device Profile (MIDP) is developed to support Connected Limited Device Configuration (CLDC) for Java 2 platform. CLDC enables dynamic content delivery of Java applications to mobile devices. The WWW consortium is also planning for further work on protocols and standards to support transparent context exchange. [43, 20, 30] are some projects dealing with device contexts. These approaches reduce the amount of information required to be sent through a limited-bandwidth.

Table 2.1: List of HTTP request headers.

Header	**Description**
Accept	Content-types that are acceptable
Accept-Charset	Content-characters that are acceptable
Accept-Encoding	Acceptable encoding
Accept-Language	Acceptable language for response
Content-Type	The content type of the body of the request

Personalization (such as user preference, user's habit and interest, user background) could be acquired through different ways. First, users explicitly need to express all information relevant to their preferences. It can be done at the user's end once, and be used repeatedly for different purposes. PDCAS is an example of using user centric approach by taking into account the user's quality of services (QoS). In [6], the user preferences need to be pre-processed explicitly before inferred with device capabilities and network constraint in the later stage. In [57] Web content is adapted based on both user-perceived-QoS and user interest related to the web content. It models user-perceived performance features via different QoS metrics that further suggests adaptation suggestions. [58], a continuity of [57], considers the quality of experience

(QoE) concept in order to ensure that the users have a positive experience using their systems and that they are happy to re-use them. This is done by using a mechanism to take into account multiple factors affecting QoE in relation to the delivery of Web content in variable environment. Personalization in learning and management system is another important project pioneered by hypermedia research group [22]. The key idea is to adapt content according to education personalization context which consider the user's goal, knowledge and background. The adapted content is tailored to the user's learning path [23]. In [59], the desired content will be selected based on user shared behavior, personal characteristic and lifetime.

Accessibility context concerns with user's accessibility towards the content. It depends on the content request's situation. For instance, browsing video affects a user's attention when one is in a meeting. Sometimes, situation also may affected by rules. Driving prohibits the driver from reading text via the mobile device. But listening to an audio file from that device is legal. This makes accessibility as an important context for content adaptation. Study in [10, 46] emphasize on enhancing user accessibility through content adaptation by using fixed rule-based engine and unit of information selection, respectively.

Client's utility/facility context is initiated to provide content adaptation based on user's need rather than satisfying constraint (e.g., device limitation, network limitation, and situation). This enables content adaptation to be treated as a service. It also provides opportunities to business providers to provide the adaptation services. The competition to provide a better service will enhance quality of services and drive content adaptation into a finer level. In [7], the spatial and temporal aspect of the audio video content types is tailored to meet specific requirement of each user, thus increasing the user's utility. This requirement is collected during user-application interaction.

2.4.3.1.1 Contexts Monitoring

The responsibility of context monitoring affects the architectural deployment. The organization describes how the monitoring involved in content adaptation systems. As depicted in figure 2.9, context monitoring can be divided into three: client's system, content application/server and third party.

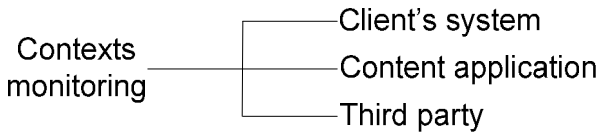

Figure 2.9: Contexts monitoring taxonomy.

In the former, monitoring is performed entirely by the client's system and suitable for client-side adaptation system. The second put entire monitoring responsibility to the origin server where the content application resides, while the latter delegates it to the third party. Third party can be a specific media adaptation proxy or a broker assigned by the client. Examples of client's system, content application and third party contexts monitoring are [36]; [2, 30, 45]; and [6, 8, 9], respectively.

2.4.3.2 Adaptation Decision Taking Engine

The approaches of ADTE affect the searching for the suitable adaptation strategies to suit the adaptation contexts. The organization describes how the engine operated in generating the decision. There are three kinds of ADTE model implemented by content adaptation systems: probability-based model, rule-based model and optimization-based model, as depicted in figure 2.10.

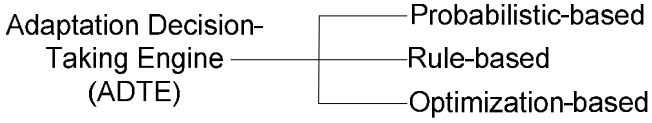

Figure 2.10: ADTE taxonomy

Probabilistic-based model utilizes probability by calculating the probability of each content version and takes the version with maximum value to achieve adaptation decision. This model is popular among browsing adaptation category systems [2, 45]. However, this model is not suitable if the required version requires cross-media adaptation which is out of the probability scope.

Rule-based model applies cause-effect structure to solve problem of adaptation decision. This model is commonly adopted in existing systems [8, 15, 24] as it produces

an exact decision stated by author and suitable for both application-specific and cross-media. One key challenge of this method is how to discover rules especially when new subject is introduced. An effective solution to this is to use scripts and agents to facilitate new rules as implemented in [8].

Optimization-based model observes ADTE as an optimal problem with constraints and derives solution using a tree scheme. The key idea is to plan the proper target function and the resolving algorithm. Adaptation contexts are referred as the target function. Some existing systems [31, 49] implemented this model only deal with specific-media fidelity layer, i.e., format adaptation. A recent project that adopts optimization model for both modality and fidelity layers is [60]. This enables cross-media adaptation decision to be supported. One key challenge with this model is to combine the target function to produce finer adaptation decision.

2.4.3.3 Service Discovery

To the best of our knowledge, there is no service discovery mechanism specifically tailored to service-oriented content adaptation has been presented. The service discovery taxonomy depicted in figure 2.11 is developed based on the solution proposed for similar Internet service systems such as grid and Web service.

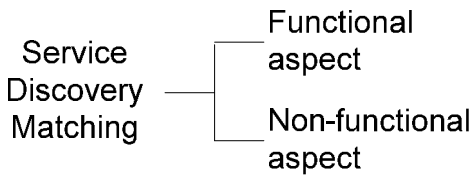

Figure 2.11: Service discovery taxonomy.

Service discovery protocol is paramount to any service-oriented system. Commercial service providers are under constant pressure to ensure their services can be easily located and invoked by clients. The service discovery requirements (i.e., service description, storage of service description, message communication and searching methods) differ from one application scenario to another (e,.g., wired, wireless, MANET, vehicular networks). Many existing service discovery solutions are originally based on UDDI reference model. UDDI specification (version 2 and 3) defines a resource directory framework and represents the standard for implementing

the business services registries that enterprises are deploying today and facilitates the service publication processes [61]. It allows for indexing, searching and retrieving services through their descriptions. On top of that, UDDI also includes security API. The security API provides authentication capability to secure service's publication and inquiry. Also, UDDI supports registries replication and enables synchronized addition, deletion and update of service advertisements between replicated registries. In this way, single point of failure of registries is avoided.

In practice, service discovery can be divided into two phases. In the first phase, a service is discovered using its functional aspect i.e., what the service does [62]. The second phase is used to select services that matched client QoS requirements. This QoS is referred as non-functional aspect of the service. Thus, current discovery solutions can be divided into two dominant approaches: function-based and non-functional-based. Function-based approaches utilize the service's description (i.e., function's name, input and output parameters, preconditions, and effects) to match user query for services. One of the well known function-based approaches is keyword matching. In wired networks where services are stationary, there are many industry standard discovery protocols such as Sun's Jini, IBM's Salutation, and IETF's Service Location protocol (SLP) that rely on keyword matching, alone [63].

Jini is a popular Java-based service discovery system. Its discovery process consists of two parts: lookup service node detection and searching or publishing. Lookup service node acts as a service registry and enables keyword-based searching. Jini is flexible as it can be deployed in any type of network as long as Java Virtual Machine is supported. The lookup service node however, is open to single point of failure due to its centralized architecture. Similarly, SLP is also a well known centralized service discovery system. A service advertisement in SLP includes its service type, unified resource locator (URL) and attributes. Thus, the query contains either the searched service type or attributes specifications or a combination of both while the reply contains the URL of the desired service [63]. On the other hand, Salutation also can be used with any network. Its architecture is flexible (can be P2P or centralized) and the service registry is installed on every salutation manager. Searching based on keyword is initiated at the salutation manager using specified protocol.

For wireless networks, Bluetooth and ZigBee are some of the industry standard service discovery protocols [63]. Bluetooth allows multiple devices to cooperate with each other and share resources. It is designed to function in the resource-constrained environment. Bluetooth however, does not enable complex queries in order to save device resources. A service discovery approach for vehicular network is discussed in [15]. It bases matching according on weighted keywords. However, keyword-based approach may return a huge list containing inappropriate services that may not satisfy the requester's intended requirements. To solve this, ontology technology is used.

Ontology organizes service profile according to its semantic [64, 65]. It is formed based on the domains of interest. Specifically, it contains a set of standard terms to describe service classes. For instance, DAML-S provides a mechanism to record semantic information or description within a UDDI registry. It also shows how the UDDI registries can be modified to use the semantic information provided by DAML-S [10]. In practice, this can be done using *tModel* element in UDDI data model. Then, *businessService* element uses this specific *tModel* (i.e., DAML-S) to index the value it stores from the DAML-S service profile it intends to represent. A reasoning process is then performed by exploiting the information included in the service description to best match the ontology. However, ontology-based approach may suffer from performance problems (e.g., inappropriate matching) due to the use of immature ontology reasoners [66].

On the other hand, non-function-based approaches incorporate the service's quality of service (QoS) attributes together with the service's descriptions (i.e., function-based). QoS is a set of service attributes that encompass performance characteristic such as cost, reliability, availability, rating, and reputation. Efforts such as OWL-Q are trying to make QoS description more flexible to describe and present the formal description of a service [67].

Existing methods such as DAML-QoS [62] and [68] perform matchmaking between the client and the service's QoS, thus requiring both QoSs must be known a priori. DAML-QoS has the capability to measure the real-time service's QoS thus ensuring up to date information. Authors in [68] use mixed integer programming to match QoS between the client and the service. Also, they propose a constraint relaxation method to enable partial matching services as potential candidate to serve clients.

However, discovering all services from an Internet-scale list that specifically matched the client's QoS requirements is time consuming. Moreover, acquiring external resources (such in [62]) to regularly measure each service QoS are tedious and expensive especially to cope with ever-growing services.

2.4.3.4 Adaptation Path Determination

As depicted in figure 2.12, adaptation path determination is generally composed of at least two inter-related steps: (a) adaptation path construction; and (b) mechanism of choosing the optimal path. To address the adaptation path construction, a directed acyclic graph (DAG) is discussed in [11, 13]. The transformation prescript graph for DAG is organized in serial manner and bounded by the media format. An extension of DAG; horizontal score tree is introduced in [15] as an alternative.

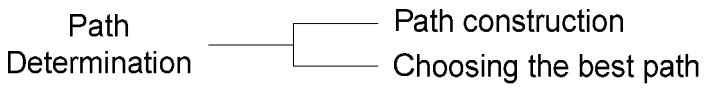

Figure 2.12: Path determination taxonomy.

To address the problem of choosing the optimal adaptation path, only one approach has been presented so far. This approach is a single objective and greedy assignment. To select the best possible path, weighted score is assigned to each path. The best path is the one with the highest score. This approach assigns single-relation score to every service's quality of services (QoS), which is accumulated to generate aggregate score for each adaptation path [13]. Adaptation service's QoS is used as the selection criteria. Single-relation score classifies every QoS to have same relation towards computing score (i.e., negative relation-high QoS value computes a low score). Although this approach is sound and relatively simple, it suffers from a number of shortcomings. For example, availability represents the existence of a particular service at any given time. Accumulating availability value into a node score will lead to a wrong conclusion about the availability of the service. Furthermore, its score computation policy misleads the optimal path determination [12].

Figure 2.13 illustrates an example of computing score applied in [13]. Let's say a content adaptation task can be served by service provider 1 or service provider 2. Let us

further assume that a suitable service provider for the task is selected based on time and rating quality of services such that time (service 1=1.0s, service 2=1.1s) and rating (service 1=4.2, service 2= 4.0). Based on equation (5.7), SPDC computes aggregate score for both service providers to be the same implying that the task could be performed equally by both service providers.

	QoS value		*QoS score*		Aggregate score
	Time	Rating	Time	Rating	
$service_1$	1.0 s	4.2	1	0	1
$service_2$	1.1 s	4.0	0	1	1

Figure 2.13: [13]'s score computation illustration.

Another possible solution to compute the paths' score is using fuzzy logic. Using fuzzy approach, QoS is normalized using the fuzzy membership function [109]. It takes each QoS value as the input and produces the normalized cumulative score for each path as the output. The fuzzy approach is useful if the membership function is known a priori, otherwise it complicates the score computation process. Also, an average client may find it is difficult to define or develop the fuzzy membership function for different QoS types.

2.4.3.5 Service Level Agreement

To the best of our knowledge, there is no specific work discussing service level agreement tailored to service-oriented content adaptation. Here, we highlight some of the main research issues on SLA management from similar application e.g., Web service, content delivery network, cloud services, grid services. The issues are divided into four as the followings:

Definition and performance QoS: It deals with standard for service level (QoS) performance parameters in Internet services: what they are and how their value are derived or computed for the SLAs. However, some SLAs are made of non-standardized QoS metrics and attributes, especially when the QoS specifications are provided by different providers. This is due to a different perception of the same concept and

different type of system reading for the same metric (i.e., different associated units [minutes versus seconds] thus implying different associated value [1 versus 60] respectively) [70]. As a result, it arouses the problem of inferring equivalence on two QoS metrics. Moreover, unlike other Internet services, it is important to have a standard to specify the accuracy of adapted content object in service-oriented content adaptation platform.

Negotiation and QoS adaptation: In practice, existing service providers advertise 'one for all' QoS offer for their published services. The offered QoS are accepted directly without further negotiation, which render this approach (i.e., fixed SLA) violation-prone [71]. Due to the heterogeneity of adaptation requirements (i.e., client devices and preference, network bandwidth and variation of amount of content requested), a mechanism for negotiating QoS levels is of important requirement. Negotiation enables QoS being negotiated for specific adaptation requirements before agreement between clients and providers is finalized. Also, in the service-oriented content adaptation scheme, there are multiple brokers negotiating on behalf of multiple clients. As such, it is possible, at any certain time instant; many brokers are negotiating with the same service provider and require service at the same time. This necessitates a strategy for one-to-many negotiation. Selected service providers however, may not been able to meet the demand by all brokers. One way to deal with this is by rejecting some requests, however, this action may increase clients' frustration and decrease service providers' reputation [69]. Alternatively, the service provider can offer QoS adaptation to the broker. If the new offer is accepted by the broker, the request can be served.

Measurement and monitoring: This issue deals with how to accurately measure the QoS being delivered to the client. Each service QoS levels should able to be measured accordingly [66]. Client should receive exact adapted content version. This necessitates accurate measurement tool for each adaptation function. A standard and cost effective monitoring apparatus should serve as the basis for effective SLA management. This issue includes the placement of monitoring apparatus.

Compliance management: This issue deals with how to manage and control QoS levels delivered to comply with negotiated SLAs. A non-compliance of SLA is not necessarily a violation. It can be caused by a conflict as well. As such, a mechanism to determine the type and the corresponding action of an SLA non-compliance case

requires urgent attention. This includes reporting mechanism to deal with clients who demand real-time reporting of SLA compliance [72]. The report is used to confirm that clients are receiving QoS levels and the adapted content version they were promised.

Noticeably, QoS is the key factor that relates the agreement within an SLA between the client and the service provider. There are several views of quality that can be implemented in managing SLA for service-oriented content adaptation platform. *Quality as functionality* is measured by considering the amount of functionality that a service can offer to its clients. One service is considered better than others in one of these two cases: it provides a function (or additional function) that is not provided by other, or/and secondly it provides a better value for the same function across providers. *Quality as conformance* is a view of comparing the actual QoS delivery with the promise. A good service means that it delivers no less than the stated promise. *Quality as reputation* depends on clients expectation and experience from the service. It is built collectively over the time of the service existence from clients' feedback [69]. A service with good reputation means that it consistently provided specific functionality with specific performance over the time. Most of existing Internet services (e.g., [72, 73]) based their QoS monitoring using *quality as conformance* view.

2.4.4 Extended Service Oriented Architecture (SOA)

In this subsection, we present the extended service-oriented architecture (xSOA) based on the framework presented in [161] and [162]. Basic components of a traditional SOA such as service registration, discovery and load balancing of service requests however, should be extended to support capabilities such as service orchestration, provisioning and service management. Figure 2.14 presents the xSOA [161] that consists of three main layers: description and basic operations; composition; and management. The bottom layer is identical to the basic SOA components i.e., publication, discovery, selection and binding. This includes the description of the service's capability, interface, behaviour and QoS. The middle layer defines the services' composition activity such as coordination, conformance, monitoring and semantics. In this layer, the networking of services established the composite services. The upper layer manages the operation of the composite services thus providing assurance and support based on the established market's rating, certification and/or SLAs agreed by trading parties. We take into account these layers to ensure the practicality of the proposed solutions.

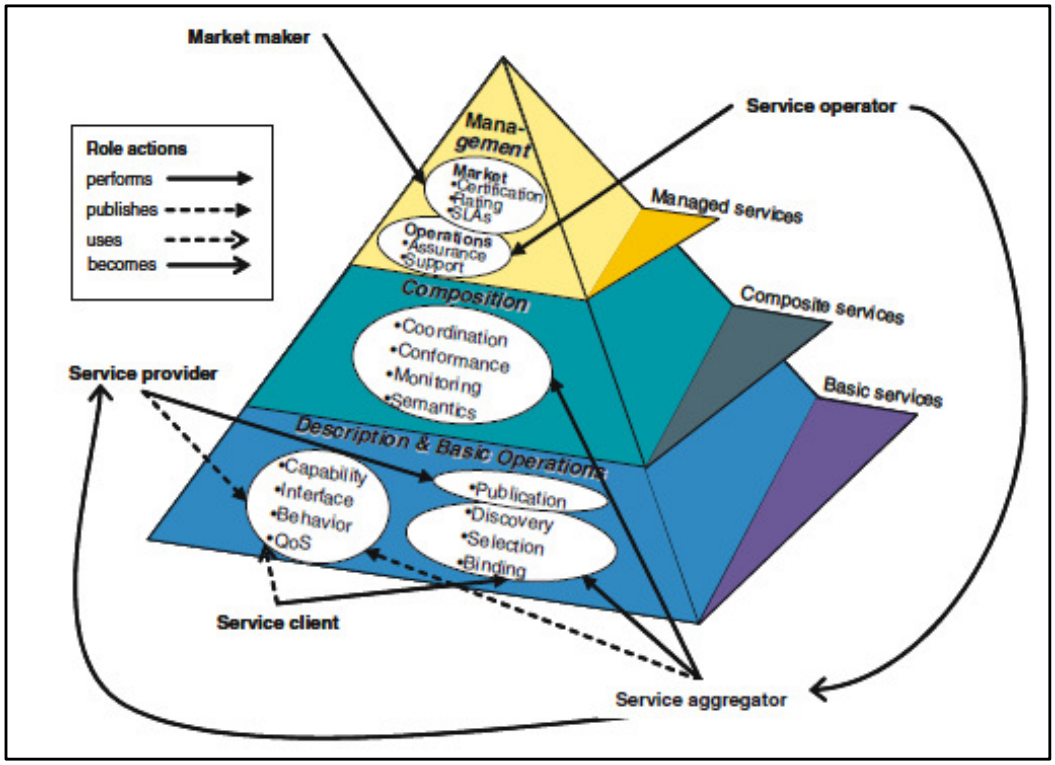

Figure 2.14: Extended SOA [161].

2.5 Mapping Taxonomy to Existing Systems

In this section, the existing systems are mapped into the content adaptation system taxonomy. Table 2.2 and 2.3 present the mapping between the centralized systems and decentralized systems with the taxonomy, respectively.

Table 2.2: Mapping representative centralized systems with the taxonomy.

Systems	Design theme	Implementation Components	Adaptation Strategy
Info Pyramid	A proxy-based browsing adaptation system that performs static content adaptation (pre-adapted multiple versions) for mobile and desktop displays	Adaptation context: device (monitor by the proxy) ADTE: optimization-based, find pre-adapted version that match the targeting device	Support appearance, size and format adaptation
Power Browser	A server-based browsing adaptation system that performs dynamic adaptation (on the fly) for mobile device	Adaptation context: device and user preference (monitor by the proxy) ADTE: optimization-based, find text units that suitable to be displayed at the targeted mobile	Support text appearance, encapsulation and navigational adaptation
PDCAS	A proxy-based media adaptation system that performs dynamic adaptation for mobile and desktop displays	Adaptation context: device, user preference and network (monitor by the proxy) ADTE: optimization-based, find the quantization steps of media that suitable to be displayed at the targeted display	Support appearance, size and format adaptation
VTP	A proxy-based browsing and (possibly media) adaptation system that performs dynamic adaptation for mobile and desktop displays	Adaptation context: device and user preference (monitor by the proxy or agent) ADTE: optimization-based, using weighted transcoding graph	Support appearance, size, encapsulation, translation and format adaptation
XAdaptor	An extensible proxy-based browsing adaptation system that performs dynamic adaptation for mobile and desktop displays	Adaptation context: device and user preference (monitor by the proxy) ADTE: rule-based, find transcoding instruction that matched the rule of the required contexts	Support appearance, size, encapsulation, translation and format adaptation

PACER	A server-based browsing (education content) adaptation system that performs dynamic adaptation for user personalization	Adaptation context: User personalization (monitor by the proxy and specify by the user) ADTE: rule-based, find content version that matched learning style	Support appearance and navigational adaptation
CAF	A proxy-based browsing adaptation system that performs partially static content adaptation for mobile device	Adaptation context: device (monitor by the proxy) ADTE: rule-based, find content version that matched the personal computer and the collaborated mobile	Support appearance, size and format adaptation
Service-based	A proxy-based browsing adaptation system that also provide dynamic value-added content for mobile device	Adaptation context: device and content's keyword being request (monitor by the proxy) ADTE: optimization-based, find content version that matched the targeted device	Support appearance, size, encapsulation, translation and format adaptation
CAIN	A proxy-based browsing and media adaptation system that performs dynamic adaptation for mobile and desktop displays.	Adaptation context: device and network (monitor by the proxy) ADTE: optimization-based, find adaptation tool that will provides best possible content version to the device	Support media, appearance, size, encapsulation, translation and format adaptation

Table 2.3: Mapping representative decentralized systems to the taxonomy.

Systems	Design theme	Implementation Components	Adaptation Strategy
ADAPT^2	A media (education content) adaptation system that performs dynamic content adaptation for users personalization	Adaptation context: personalization (monitor by the proxy and specify by the user) ADTE: rule-based, find content version that matched learning style rule. Discovery and Selection: N/A Monitoring: N/A	Support appearance, media and navigational adaptation
DCAF	A service-oriented browsing and media adaptation system that performs dynamic adaptation for mobile and desktop displays	Adaptation context: device, network and user preferences (monitor by the proxy and specify by the user) ADTE: N/A Discovery and Selection: Discovery through service registry, path determination using single objective function (QoS-based). Monitoring: recovery mechanism is provided	Support media, appearance, size, encapsulation, translation and format adaptation
PUMA	A service-oriented browsing and media adaptation system that performs dynamic adaptation for mobile and desktop displays	Adaptation context: device, network and user preferences (monitor by the proxy and specify by the user) ADTE: Pareto preference graph similar to rule-based Discovery and Selection: Discovery through service registry, path determination using single objective function (cost-based). Monitoring: recovery mechanism is provided	Support media, appearance, size, encapsulation, translation and format adaptation

2.6 Positioning the Thesis

In this section, we discuss some open challenges for content adaptation research area and then position the thesis. The proposed taxonomy and the covered existing systems establish the background for this thesis. Our work on broker-based service-oriented content adaptation is targeted to provide a scalable and flexible platform to content adaptation.

Our investigation on the current content adaptation systems reveals that many of them are using centralized architecture. Centralized content adaptation system is easy to manage; however, suffers some serious drawbacks such as single point failure, get overloaded during peak demand and unscaleable. Specifically, users cannot get the adapted page when the server or the proxy is down. In term of scalability, the adaptation strategy(s) supported at a particular server or proxy is limited.

On contrary, decentralized architecture enables designer to break the adaptation tasks and delegate it to different location. It is scalable and many content adaptation services are available across wide area network. Decentralized architecture however, requires more components (e.g., service discovery, path determination, distributed adaptation control), thus, elevates performance, cost and management issues. To date, only a few rudimentary frameworks exist to enable distributed content adaptation. These service-oriented systems such as DCAF and PUMA present some basic components thus; a more comprehensive framework is required.

The issues considered in the taxonomy provide us a guideline for developing a content adaptation system. We follow a service-oriented approach to develop an architecture for content adaptation (Chapter 3) by exploiting the emergence of Web services that can be used as content adaptation services. A flexible platform for passing and delivering the content (e.g., original, partially adapted or fully adapted) across the adaptation proxies is presented.

In this thesis, we have limited focus on the issues for service discovery, path determination and service level agreement for service-oriented content adaptation. We have developed a service discovery protocol (Chapter 4). Although there are many service discovery protocols exist, there is none to the best of our knowledge for service-oriented content adaptation. Moreover, existing protocols that serve similar purpose could not be directly adapted to service-oriented content adaptation as they tend to

perform extensive searching (i.e., search for all available published services) which makes it time consuming. Furthermore, matching client's QoS requirements with all services is unrealistic. Also, as content adaptation is served by a network of services, discovering closer providers is an obvious requirement. Therefore, what is required is a discovery algorithm that quickly terminate when specified search space is achieved.

Then, we present a path determination mechanism to select the best possible services for providing content adaptation services (Chapter 5). Even though a specific solution has been proposed by [5, 13] using a single objective assignment function, it suffers from a serious shortcoming. Its score computation policy misleads the optimal path determination. The proposed solution overcomes this shortcoming.

In chapter 6, we address the requirement for service level agreement in content adaptation context. Much research work has been done on service's quality definition and description for Internet and Web services [62, 66]. Efforts such as WSML [74], WSLA [76], DAML-QoS [75], OWL-Q [62] are trying to make QoS description more flexible to describe and present the formal description of a service. To address non-standardized QoS metrics and attributes, a mechanism for mapping SLAs is studied in [70]. However, to the best of our knowledge, there is no attention given on describing content adaptation performance QoS in service-oriented content adaptation.

Dealing with adaptation decision-taking engine, service orchestration, content distribution, fault tolerance, recovery and security issues do not fall within the scope of this thesis. Therefore, we refrain from developing solutions for them. In this chapter, we have provided necessary references and brief discussion on some of these issues.

2.7 Summary

In this chapter, we have presented a survey and taxonomy on existing content adaptation systems. After analysing the content adaptation research landscape, we have developed a taxonomy based on three issues: design themes, content adaptation strategies and implementation components. Hereby, we provided pointers to related work in each context. Also, we mapped the developed taxonomy to some representative centralized and decentralized systems. This mapping provides an in-depth analysis and complete understanding of the content adaptation and to validate the applicability of the proposed taxonomy.

The representative systems surveyed are mostly focused on dynamic browsing adaptation category. The only static browsing adaptation systems surveyed are [2, 36]. In term of adaptation strategies, most of them support size, appearance and format adaptation strategies. Only a few support cross-media strategies (e.g., encapsulation, translation, conversion). Most of the surveyed systems deal with media content and presentation content types, and adopted the three layers model to map the content structure. Also, an abstract model has been developed according to the essential requirements of content adaptation.

In the next chapter, we present a comprehensive architecture that enables brokers to manage content adaptation on behalf of the clients using available content adaptation services.

Chapter 3

SOCA Framework

In this chapter, we develop a comprehensive service-oriented content adaptation (SOCA) framework. This framework enables content adaptation to be performed as services. Establishing content adaptation as a service allows the use of a large number of adaptation mechanisms located in many places in the network. The proposed service-oriented content adaptation framework consists of essential enabling components such as adaptation decision-taking engine, service discovery, path determination and service level agreement.

3.1 Introduction

Over the past years, a considerable amount of researches use strategies such as content selection, transcoding or distillation, to perform content adaptation have been discussed [15]. These strategies can be performed at a particularly designated proxy (i.e., proxy-side adaptation) or at an origin server (i.e., server-side adaptation) or at the client device itself (i.e., client-side adaptation). Client-side approach suffers computation limitation; server-side approach leads to overload problem when experiencing flash crowds and open to single point of failure; while proxy-side approach is only workable if the adaptation context(s) and content metadata are known a priori [13, 19].

Most importantly, the common thread among all these approaches is that they perform well in browsing (e.g., adapting layout, text column) and single element content adaptation (e.g., converting format within a content media). However, cross media adaptation (e.g., converted, translated or integrated from one media into another)

requires different adaptation functions than the one used for single adaptation such as fidelity adaptation (i.e., convert two bits image into black and white image), modality adaptation (i.e., change one column text into two column), layout adaptation (i.e., change the orientation of the Web) and structure rearrangement (i.e., organize long text into read more option) [47, 77]. For instance, considered this scenario:

> *Suppose Mohammad went to visit his relative at Royal Women's hospital in Melbourne. Mohammed learns that his relative has been diagnosed with a heart complication. To get more information on the heart complication that his relative diagnosed with, Mohammed decided to browse, using his web-enabled mobile phone, the e-health server at the hospital. Confronted with medical jargons received from the e-health server and to make sense of it all, Mohammed decided to browses an e-learning server. Mohammed prefers graphic-based explanation of the heart condition in Spanish. However, the content on the e-learning server is a video and it is in English.*

To achieve the desired content form (i.e., summarized Spanish audio corresponded to the related image sequences), at least four adaptation tasks are required: (1) video type conversion to a series of images, (2) text translation, (3) summarization of the information in Spanish; and (4) text to speech conversion. None of the aforementioned approaches is capable of serving these content adaptation tasks. Hence, there is a need to perform and manage the entire tasks in distributed location where it can be possibly done. What is required is a flexible platform for dynamically adapting and delivering the content (e.g., original, partially adapted or fully adapted) across the distributed proxies and servers. The platform should be reliable and scalable.

The emergence of web services has made it possible to provide content adaptation to a new level; where the tasks are facilitated by service proxies [13]. As such, a variety of content adaptation services can be offered by multiple providers. Clients have more option of choosing which services to be consumed. On the other hand, some essential enabling mechanisms are required to ensure these services can be located, contacted, negotiated and cooperated with one another to serve the content adaptation requests.

3.2 Service-oriented Content Adaptation (SOCA)

3.2.1 Architecture

Figure 3.1 illustrates the high level concepts of the proposed service-oriented content adaptation system architecture. The system is based on service oriented architecture (SOA), which is a form of distributed system architecture that is typically characterized by logical view of actual entity; message oriented with service description; and platform neutral [78, 79]. A brokering approach is used to manage client requests.

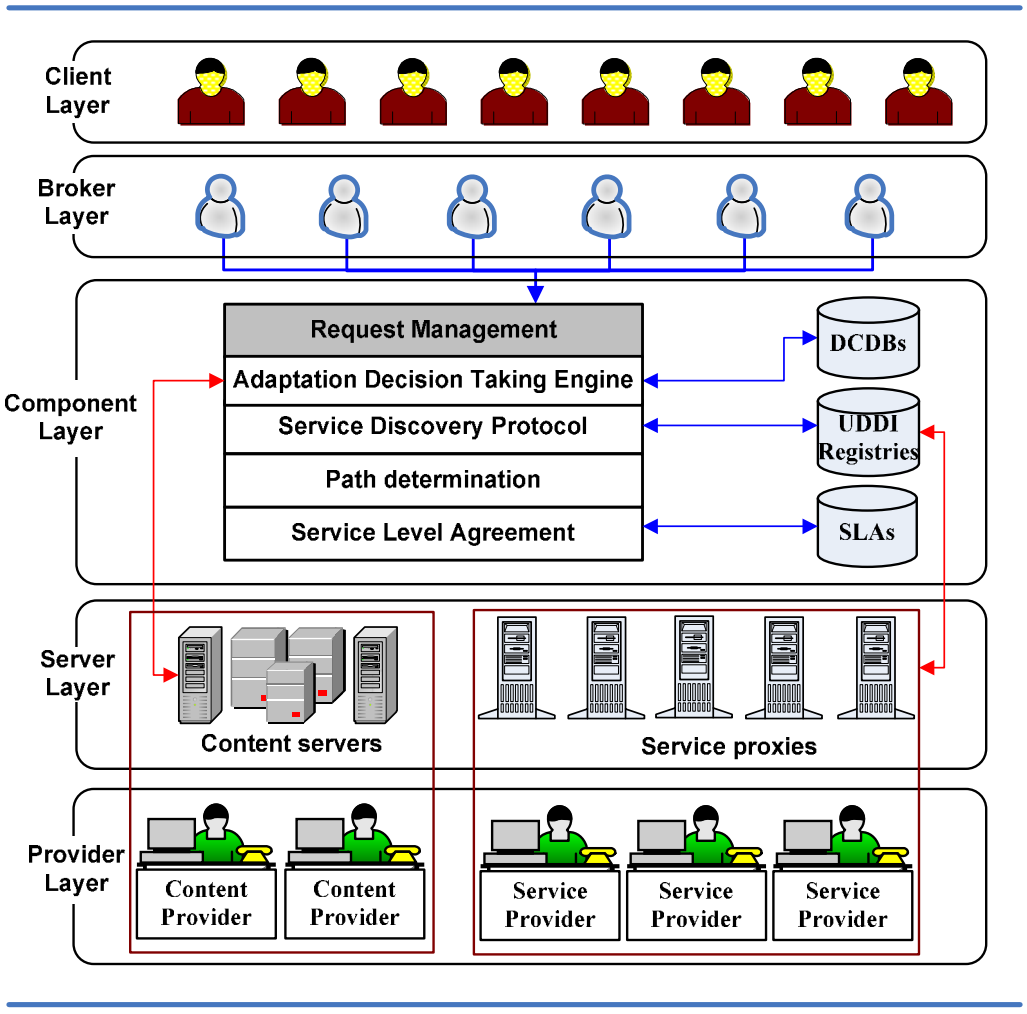

Figure 3.1: Service-oriented content adaptation framework.

The architecture is made of five inter-related layers: client, broker, component, server and provider. It consists of components that provide access to content servers, formulate user request to source format, manage and provide content description (meta-data). Clients made request to contents via heterogenous devices (e.g., multimedia desktop, PDA and mobile phones). The content providers maintain content servers that store the content and they are distributed across the Internet. Similarly, there are several service providers providing content adaptation services located in many places in the network. The broker, service discovery and path determination components of the system cooperate to locate and select the best possible service for the query to provide a content version required by the clients.

The client and broker layers deal with those aspects that describe how incoming client requests are handled by the broker. The broker uses the ADTE to analyse content adaptation requirements. The input to ADTE is the adaptation contexts (e.g., device profile, client preference). A device profile, stored in the device capability database (DCDB), contains all relevant and device-specific information to accurately render output pages for display at the respective client device. An incoming client request is received by the broker via the HTTP protocol [80]. The user-agent header from this request is extracted and looked up in the DCDB to identify the requesting device. With the device identified, all device capabilities are retrieved from the database as well. Client's device capability is represented according to the composite capabilities/ preference profile (CC/PP) specification introduced by World Wide Web Consortium (W3C). Devices can be detected through Bluetooth or ZigBee configuration, if activated. DCDBs can be placed in distributed location and should be synchronized timely. Also, a client may have specific preference towards the required content version. This preference can be explicitly supplied to the broker.

At the component layer, four inter-related components such as ADTE, service discovery, path determination and SLA are defined. These components are used by the broker to manage requests issue by the clients. The primary role of the ADTE is to analyse the content adaptation requirements and to produce the required tasks. Service discovery component lookups for potential services from accessible registries that is capable of performing these tasks. Path determination component select the best possible service for each task using QoS criteria. For each selected service, the broker

contact the provider using the service handle. If the provider agrees to serve the request according to the advertised QoS, SLA is settled; otherwise they may negotiate the QoS. SLA ensures client to get the required content version within the agreed QoS.

The server and provider layers have the platform of providing the content and content adaptation service. These services provide the client with a content version, which is adapted to the requesting device using content adaptation methods tailored to the requiting devices. Content server is the origin server where the Web content resides. It is provided by the content providers and distributed across the Internet. A service provider publishes and periodically updates their services at the UDDI registries. Examples of description that are maintained in the service registry include the service provider, adaptation function types with supported formats, available bandwidth, availability status, cost, adaptation time, handle and binding template (refer table 3.1). A new service provider must publish their services. A reputation bootstrapping for trust establishment among services can be used to enhance trustworthiness of the service proxies [81].

Table 3.1: Example of service registry.

Description	Service type	Input (format)	Output (format)	Location	Cost (cent)	Time (sec)	Availability status	Handle
Service 1	Translate ( )	word (doc)	Word (txt, rtf)	http://..	free	10	on	…
Service 2	Convert ( )	Video (avi)	Images (tif)	http://..	50	15	off	...
Service n	...	...	...	...	...	…	...	...

3.2.2 Interaction Protocol

We now describe how the system components described in figure 3.1 interact with each other to accomplish the content version required. Figure 3.2 shows the interaction sequence diagram. A client issues a request for the content to the corresponding content

server through the nearest broker. The broker then fetches the adaptation contexts (e.g., client preferences, device capability from the client profile database (DCDB), and the QoS requirements).

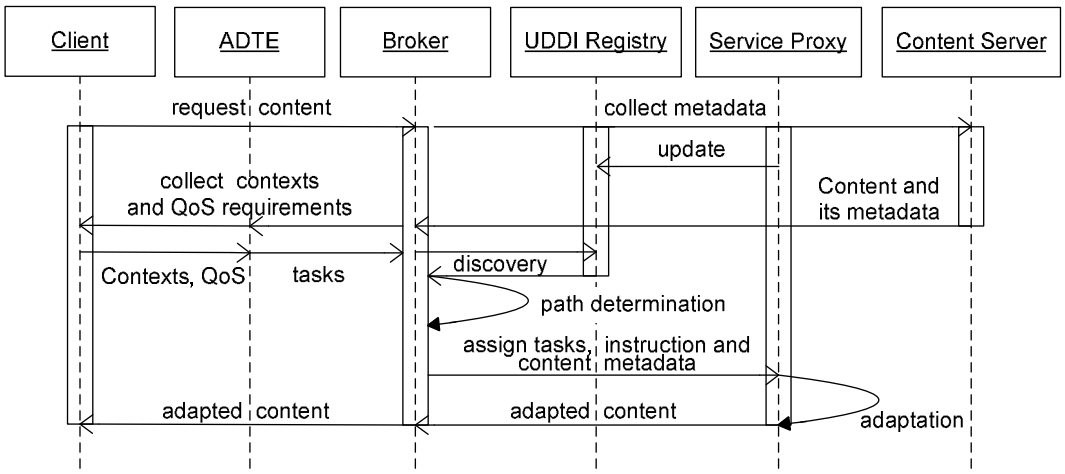

Figure 3.2: Content adaptation sequence diagram.

In response to the broker (on behalf of the client) request for the Web content, the content server sends back metadata for the requested content. At the same time, the local proxy gets the client profile from client host. It then matches the collected client profile with the client profile that exists in the DCDB. If there is no entry for the client, the collected client information is added to the DCDB. Using these adaptation contexts, the broker uses the ADTE to analyse the requirement for content adaptation. The ADTE constructs the corresponding semantic representation and maps the tasks accordingly. The output from the ADTE is the required tasks. Using these outputs, the broker uses the service discovery mechanism to locate potential services from the network that capable of performing the tasks and matched client QoS requirements. These services are published at the UDDI registries. The service provider periodically updates its proxy(s) description and QoS to the UDDI registry.

Then, the path determination mechanism computes the best possible adaptation path. This path consists of a set of services; each of which will be used to perform a particular task. The broker assigns tasks including the related adaptation instruction to the selected service proxies for adaptation and monitors the overall content adaptation

process, including the tasks distribution. SLA can be used to ensure content adaptation is performed promptly by service providers. Finally, the broker sends the required content version back to the client.

3.2.3 Request Management Components

The enabling components include adaptation decision taking engine (ADTE), service discovery, path determination and service level agreement (SLA).

3.2.3.1 ADTE

ADTE uses adaptation contexts as the input and produces the required tasks. These tasks provide the client with the content version required. ADTE is made of two essential processes: representation modelling and task mapping.

3.2.3.1.1 Semantic Representation Model

The information collected by the local proxy is mapped into the semantic representation. Semantic representation is used to convey the collected data into linguistic term in order to relate their form together. This representation is then used to compute the required tasks using rule-based technique. Generally, we can describe the entire context semantic using the equations 3.1 to 3.3. There are *n* adaptation factors (equation 3.1). Each adaptation factor, $factor_i \in adaptation_{factors}$, has *m* attributes (equation 3.2) and each attribute, $attrbute_j \in factor_i$, has *k* quantization steps (where each quantization can be represented in a certain value) (equation 3.3):

$$adaptation_{factors} = \{factor_1, factor_2, request_{context,} \dots factor_n\} \quad (3.1)$$

$$factor_i = \{attribute_1, attribute_2, attribute_{3,} \dots attribute_m\} \quad (3.2)$$

$$attribute_j = \{qstep_1, qstep_2, qstep_{3,} \dots qstep_k\} \quad (3.3)$$

The following is an example of adaptive context representation:

$$adaptation_{factors}$$
$$= \{user_{preferences}, user_{situation,} device_{profile,} network_{parameters}\}$$
$$request_{context} = \{preffered\_language, readability, visual\}$$
$$user_{preferences} = \{scrolling\}$$

$$user_{situation} = \{meeting, driving\}$$
$$network_{parameters} = \{bandwidth\}$$
$$device_{profile} = \{screensize, color, supportedformat\}$$
$$color_{properties} = \{blackwhite, 2bits, 16bits\}$$

For more detail regarding semantic representation for the constraints/resources based context, and media utility context, interested readers can refer to [6, 9, 8, 10, 24, 31] and [7, 82] work, respectively.

We assume that the Web authors provide the Web page and content with the metadata during authoring process. This will make unnecessary real time content decomposition and information loss. The authors will have some control over the final presentation. We use the universal content structure model [10] to represent the content metadata as follow:

$$Web\_content_{properties} = \{property_1, property_{2,} property_{3,} \dots property_n\}$$
$$property_n = \{object1_{metadata}, object2_{metadata,} \dots objectN_{metadata}\}$$
$$object1_{metadata} = \{qstep_1, qstep_{2,} qstep_{3,} \dots qstep_n\}$$

The following is an example of the web content metadata representation:

$$Web\_content_{properties} = \{content, navigation, presentation\}$$
$$content_n = \{video_{metadata}, audio_{metadata,} image_{metadata}, language_{metadata}\}$$
$$navigation_n = \{hypertext_{metadata}, link_{metadata,} hypermedia_{metadata}\}$$
$$presentation_n = \{modality_{metadata,} column_{metadata}\}$$
$$column_{metadata} = \{single, double, triple\}$$

Preference context is important to facilitate client need. For example, a person visiting France could state “I want to hear the news in English”. We can describe this context as follows:

$$preference_{context} = \{language_{media}, readability_{media}, browsing_{media}\}$$

where the conditions can be assumed as below:

$$condition\ 1: language_{audio} = \{english\}$$
$$condition\ 2: readability_{text} = \{summarize\}$$
$$condition\ 3: browsing_{text} = \{change\_audio\}$$

These contexts can be captured using mechanisms such as the query-based approaches (e.g., [83]) or more interactive approaches (e.g., [10]).

3.2.3.1.2 Mapping Representation to Tasks

We use rule-based technique for mapping requests to tasks. The rule takes a form as shown below:

Condition n: `If situation == x, then y;` (3.4)

where "condition" is the particular request to be fulfilled, "x" is the default media state/format to be analysed and "y" is the required tasks.

For example, suppose we have a user requesting for a sport news video highlight in English while driving using his PDA. In this case, we have to express rules for "driving situation", "audio_length", "video_length",and "audio_original". Based on equation 3.4, these rules can be stated as follows:

Condition 1:

```
If video_length > 1 minute
          keyframe extraction ( );
```

Condition 2:

```
If situation == driving
     convert video to audio ( );
```

Condition 3:

```
If highlight_audio == long
     summarize audio( );
```

Condition 4:

```
If audio_original =! audio preferred_language
     translate audio ( );
```

Then, the analyser gathers all the required tasks and lists it as below:

```
keyframe extraction(),
convert video to audio(),
summarize audio(),and
translate audio().
```

We can rearrange the sequence of the tasks (based on logical dependency) as in figure 3.3:

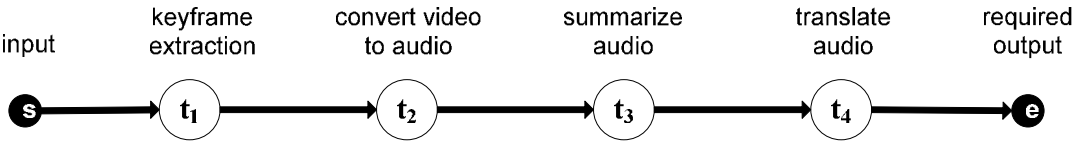

Figure 3.3: Required tasks precedence graph.

Next, the path determiner plans for the tasks allocation to the related service proxies by taking into account the information from service profile registry. The original content or partial adapted content will be passed within service proxies to be adapted.

To provide the client with required content version, available services are used. In contrast, an adaptation task can be performed by one or more services. As such, a complete adaptation request may consist of several tasks that require services from several service providers. There are many content adaptation services located across the network. To benefit from these services, the client must be able to locate and discover

them in the network. This requires a mechanism to the services. In the following subsection, we will elaborate on how this is achieved.

3.2.3.2 Service Discovery

Service discovery is a protocol to show step by step how services are discovered. As depicted in figure 3.4, content adaptations are provided as services and are geographically distributed across the wide area network. Each service is advertised at the service registry(s) by the service provider.

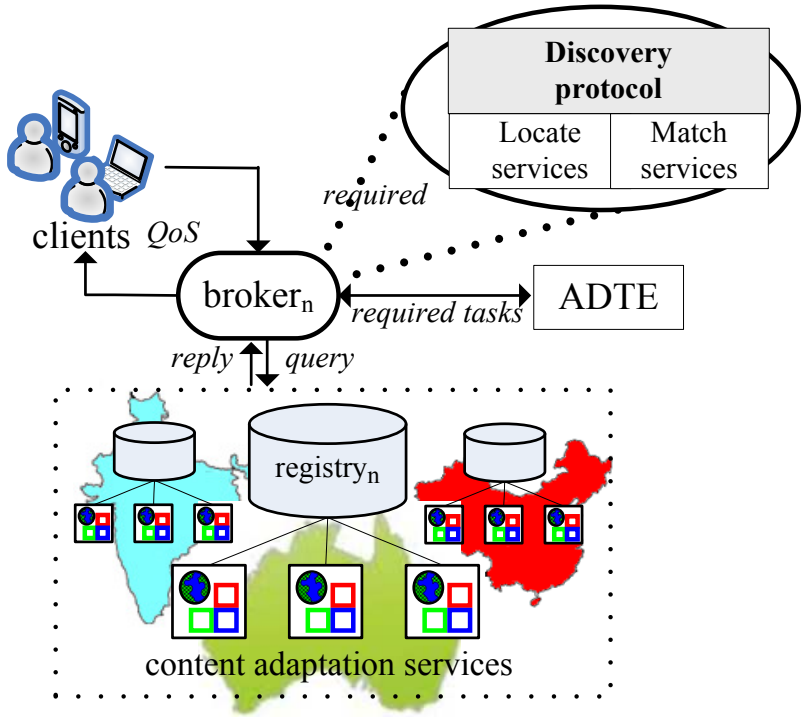

Figure 3.4: Generic service discovery component.

The service registry is part of the service discovery component and it is similar to UDDI. The registry is responsible for caching advertisement from available service with some other maintenance information including the semantic information (e.g., the type of data the service handles, QoS that the service provides) and allows for service look up. Service description is stored as a set of attribute-value pairs. XML-based representation is used for extensibility and openness. The service registry also maintains the availability status and will ensure that the services update their status periodically.

A broker, on behalf of a client, initiates lookup for the available services at the accessible registry using the inquiry API. Clients provide their QoS requirements to the broker so that potential services that are tailored to the client requirements can be located. Each client has different QoS requirements. Then, through inquiry API, the broker sends a runtime message via HTTP protocol to the registry to perform query. The inquiry allows the broker to locate and obtain details entered in the registry. At the registry, it responds to the query and returns the message containing set of services and required information, including the handle and binding template. Upon receiving the reply message from the registry, the broker matches up these services with the given requirements. For each task, the broker returns one or more shortlisted services. The service providers can be automatically invoked using their binding templates. If there is no service specifically matched up the client requirement, the broker has the ability to relax the constraint, i.e., client QoS. In this way, the partial matched service can be used to perform content adaptation.

In order to choose the best possible adaptation path, a mechanism to compute and assign score to each path is required. In the following section, we will elaborate on how this is achieved.

3.2.3.3 Path Determination

The path determination mechanism plans for the tasks allocation to the related services by taking into account the services' QoS. The original content or partial adapted content will be passed between service proxies to be adapted.

For the service proxy to adapt content on behalf of the user, it uses the offered services. In contrast, an adaptation task can be performed by one or more service proxies. There is many to many relationships between tasks and service proxies. As such, a complete adaptation request may consist of several tasks that require services from several adaptation service proxies. This requires a mechanism to schedule the tasks to the service proxies. In the following subsection, we will elaborate on how this is achieved.

3.2.3.3.1 Task Scheduling

Tasks can be performed serially, in parallel or both. The outputs of the path determination mechanism are the assignment of tasks to the related service proxies.

We have *n* tasks: $t = \{t_1, t_2, \ldots t_n\}$ and *m* service proxies: $S = \{s_1, s_2, \ldots s_m\}$. A task may depend onto another task $\{t_2 \rightarrow t_1\}$ or independent of one another $\{t_2 \leftrightarrow t_1\}$. A task $t_i \in T$ can be performed by multiple services.

$$T \rightarrow S = \{t_1 : s_1, s_2;\ t_2 : s_3, s_4;\ t_3 : s_5, s_6\} \tag{3.5}$$

The problem of determining the best path for the tasks $t = \{t_1, t_2, \ldots t_n\}$ by services $S = \{s_1, s_2, \ldots s_m\}$ can be generally viewed as decision making problem. We apply a horizontal path's tree (from left to right rather than up to down, in order to clearly show the possible connection between start and end node), where the desired path will be selected based on score node. We use the comparative method (e.g., compare the value at the i^{th} node with the maximum or minimum node value at the same level). The comparative method is simple and the score is fine-grained.

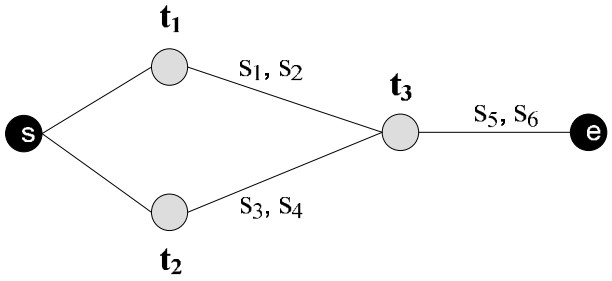

Figure 3.5: Possible adaptation paths.

To illustrate the proposed technique, suppose we have tasks $t = \{t_1, t_2, t_3\}$, where t_1 is conversion of video to animation, t_2 is translation of Spanish to English audio (of the video) and t_3 is media summarization of the animation. Tasks t_1 and t_2 are independent of each other, but both are the predecessors of t_3. Suppose the three tasks t_1, t_2 and t_3 can be performed by any of the following service providers: $\{s_1, s_2\}$, $\{s_3, s_4\}$ and $\{s_5, s_6\}$. The assignment of the tasks to the services can be represented as critical path analysis as shown in figure 3.5. We represent this as a decision problem to find the optimal score in a path tree.

Decision criteria are divided into two: positive monotonic and negative monotonic. Table 3.2 depicted some examples of criteria categorization. New criterion can be simply included according to the relevant category.

Table 3.2: Examples of criteria categorization.

Positive monotonic QoS	Negative monotonic QoS
Rating	Service cost
Reliability	Adaptation time
Trustworthy	Transport time

The value of a positive monotonic QoS is proportional to the score. The higher the value, the better is the score. On the other hand, the value of a negative monotonic QoS is inversely proportional to the score. The score is higher if the QoS value is lower. Figure 3.6 depicted the first order characteristics of the positive and negative monotonic QoS towards score.

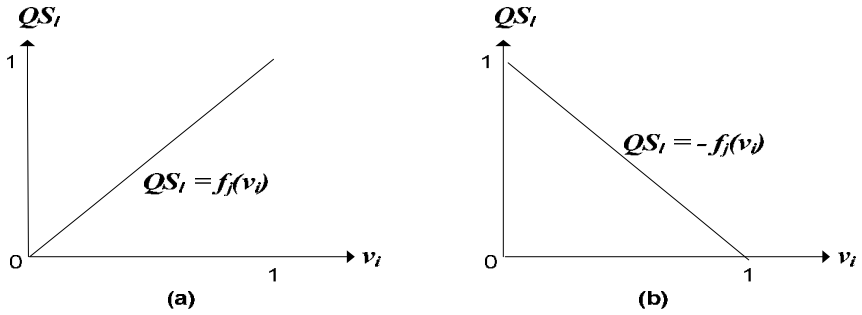

Figure 3.6: (a) positive monotonic QoS (b) negative monotonic QoS.

In Figure 3.7, each path (1 to 8) is associated with an aggregate score. The Aggregate Score *AgQS (P)*, for each path is based on the accumulated score of each residual service, where *k* is the number of tasks, is defined as equation 3.6. A score for each service is computes based on the QoS value.

$$AgQS\,(P) = \sum_{i=1}^{k} QS_i\,. \tag{3.6}$$

Highest aggregate score is represented as the best adaptation path.

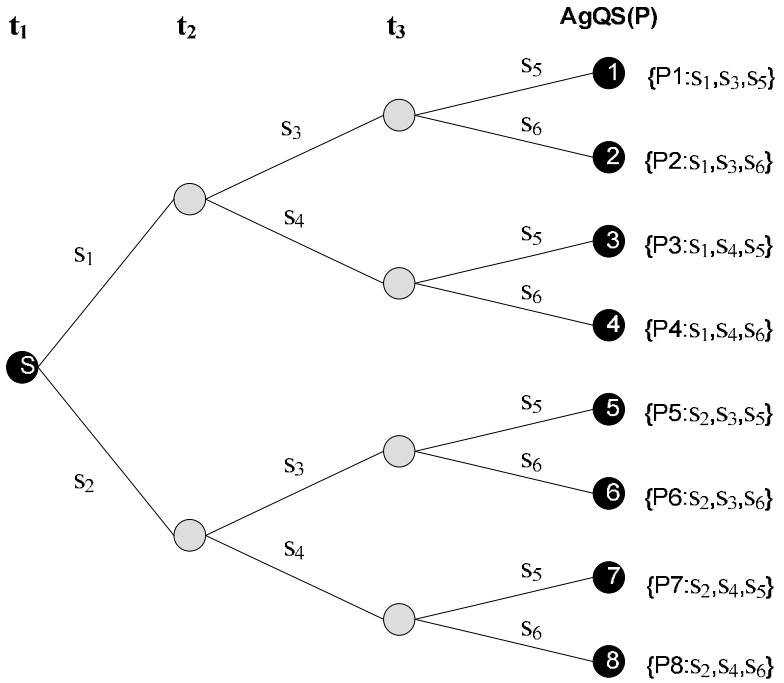

Figure 3.7: Example of paths' score tree.

After the best possible adaptation path is chosen, the original content or partial adapted content is passed within service proxies. This requires a mechanism to deliver the content (including the adaptation instruction) to the service proxies. In the following subsection, we will elaborate on how this is achieved.

3.2.3.3.2 Task Distribution

Task distribution (including content) is demonstrated using communication models, similar to [84, 85, 86]. These models are chosen because it is easy to be understood and clearly show the disjoint portions. Task distribution can be applied either using one-way communication model (OCM), or simultaneous communication model (SCM), or both. For instance, content (message) is treated as input output in a symmetric function *f (t)* given with two disjoint portions of the adaptation task input *t*, as in equation 3.7.

$$t = t_a . t_b \, . \tag{3.7}$$

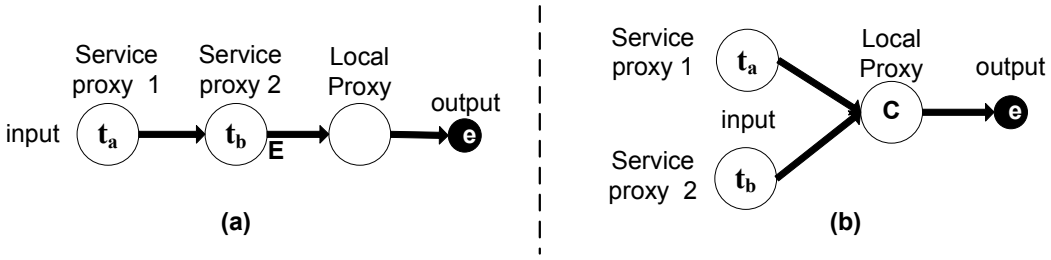

Figure 3.8: (a) OCM (b) SCM.

In OCM, t_a is the predecessor of t_b. Assume, t_a is translation of Spanish to English text and t_b is summarization of English text. Service proxy 1, performs portion t_a, and sends the output $D(t_a)$ to service proxy 2, who performs portion t_b based on t_a input (refer figure 3.8a). Service proxy 2 then combines both inputs (as in equation 3.8) and sends the output (fully adapted content) to the local proxy.

$$E(D(t_a), t_b) = f(t_a, t_b)\,. \tag{3.8}$$

SCM is applied in the case where both task is independent. Assume, t_a is conversion of video to images and t_b is conversion of Spanish to English audio. As shows in figure 3.8b, both service proxies 1 and 2 send an adapted input $D(t_a)$ and $D(t_b)$ respectively, simultaneously to local proxy who combines the end result, as in equation 3.9.

$$C\big(D(t_a), D(t_b)\big) = f(t_a, t_b)\,. \tag{3.9}$$

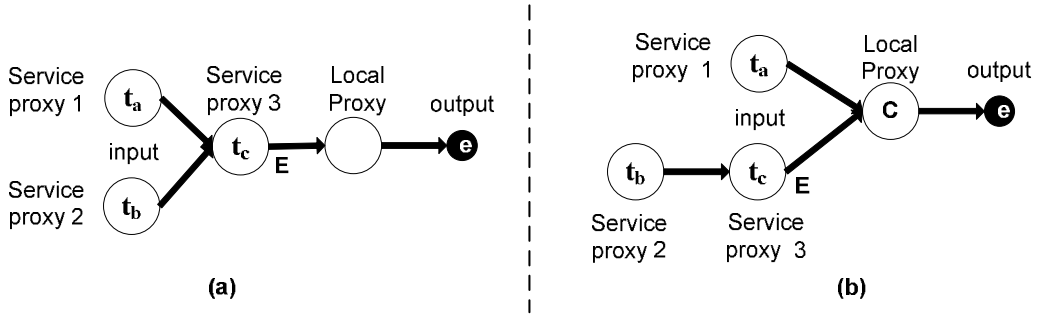

Figure 3.9: (a) Combining SCM tasks with OCM (b) Combining OCM tasks with SCM.

Now, let us consider a scenario with three disjoint portions. Assume, t_a is conversion of video to animation, t_b is translation of Spanish to English audio (of the video) and t_c is media summarization of the animation. Let say t_a and t_b are the predecessor of t_c. Service proxies 1 and 2, performs portion t_a, and t_b respectively and send the output $D(t_a)$ and $D(t_b)$ to service proxy 3. Then, service proxy 3, performs portion t_c based on $D(t_a)$ and $D(t_b)$ and sends the output to the local proxy (refer equation 3.10). Finally, the local proxy forwards the final output to the user (figure 3.9a).

$$E(C(D(t_a), D(t_b)), t_c) = f(f(t_a, t_b), t_c)\,. \tag{3.10}$$

Consider another scenario with three tasks. Assume, t_a is conversion of video to images, t_b is translation of Spanish to English text and t_c is summarization of English text. Let say t_a is an independent task, while t_b is the predecessor of t_c. So, service proxy 1, performs portion t_a and sends the output $D(t_a)$ to the local proxy. Alongside, service proxy 2, performs portion t_b and sends the output $D(t_b)$ to service proxy 3, with which performs t_c based on $D(t_b)$ input and sends the output to the local proxy (refer figure 3.9b). As a final point, the local proxy computes both inputs (as in equation 3.11) and sends the output (fully adapted content) to the user.

$$C\big(E(D(t_a), t_b), D(t_c)\big) = f(f(t_a, t_b), t_c)\,. \tag{3.11}$$

After a service has been selected, the broker communicates with provider to perform the request. The broker and the provider settle the SLA if both parties agree with the QoS. In the following subsection, we will elaborate how this is achieved.

3.2.3.4 Service Level Agreement

SLA is used to express commitments, expectations and restrictions in a transaction between two parties. Specifically, the main objectives of SLA in service-oriented content adaptation are (a) to facilitate two-way communication between negotiating parties that includes understanding of need, priorities, and specifications, (b) to protect

against expectation creep that includes the identification and negotiation of service levels, (c) to have mutually agreed standard, and (d) to gauge service effectiveness that includes the basis for performing assessment. In our context, it provides clients with the platform to specify their QoS requirements and ensures the QoS offered by the selected service providers are delivered accordingly. Figure 3.10 depicted the SLA management framework.

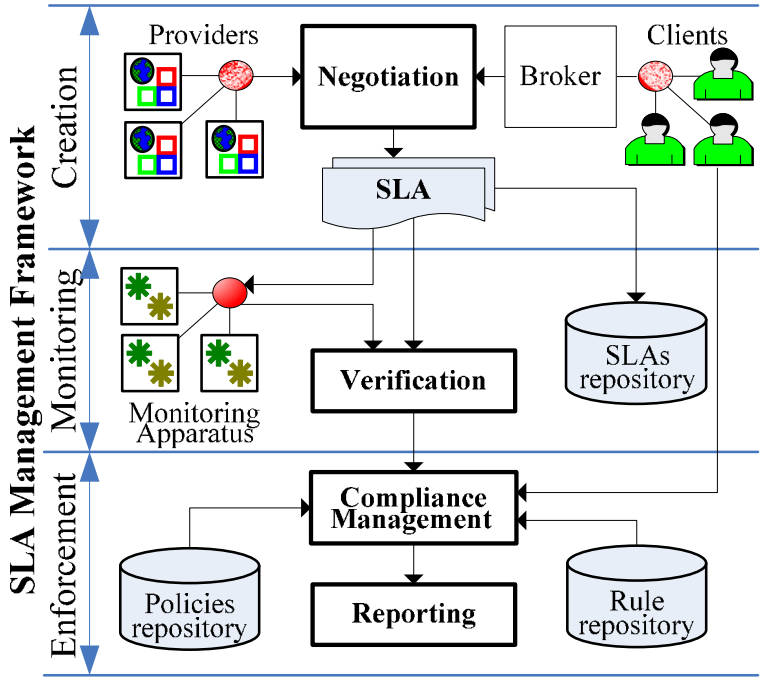

Figure 3.10: SLA management framework for service-oriented content adaptation.

The SLA management framework consists of three inter-related phases: creation, monitoring and enforcement. Brokers and providers have the mechanism to establish SLA. A broker, on behalf of the client, negotiates SLA with service providers. These newly created SLA clearly express the required QoS to be maintained till the end of services execution, the required content object level to be delivered, the penalties in case of failure to provide the offered QoS and the resolution actions in case of a conflict.

After the successful creation of SLAs, providers are tasked to perform adaptation. These services executions are monitored using specified monitoring apparatus to ensure

offered QoS and required content adaptation are obeyed. The monitored QoS levels are compared with SLA created. If the SLA for the service is being delivered accordingly, the SLA is in compliance; otherwise the SLA is in non-compliance form. When a non-compliance case detected, the enforcement phase is invoked.

In the enforcement phase, a specific making mechanism is used to decide whether a non-compliance case is a direct violation, a conflict or a result from bypassing conditions. Based on the determined result, it enforces the necessary action and to provide a real-time compliance reporting to clients. In case of any violation or any conflict, the enforcement mechanism is activated to penalize the provider or to resolve the conflict, respectively. The broker sends a real time report to the client for each service consumed including the compliance status.

In order to pass and adapt the content between service proxies, a mechanism to connect and manage the proxies (both the broker and service) is required. In the following section, we will elaborate on how this is achieved.

3.2.4 Content Adaptation as Web Services

The broker settles the SLA with the selected service providers and bonds them together. These service proxies are responsible to complete their assigned adaptation task(s). To steer the task distribution, each proxy is provided with the address of previous and next destination proxy. Based on the determined destination, the current proxy is required to pass their partially adapted content to the consecutive assigned proxy. This passing procedure is continued until a complete version is produced and sent back to the broker before reaching the client. In this way, the broker can ensure that the adapted version is in compliant with the version required by the client. Now, the basic requirement of utilizing content adaptation as the Web services is elaborated.

3.2.4.1 Requirements

Web services is an application accessible through a uniform resource locator (URL), that is accessed by clients using extensible markup language (XML) based protocols such as simple object access protocol (SOAP) [87]. It is designed to support machine to machine interaction over the hypertext transfer protocol (HTTP) used on the Internet. Web services are defined using Web Services definition language (WSDL) file. WDSL

provides the XML grammar for describing network services as collections of communication endpoints capable of exchanging messages [88].

As depicted in figure 3.11, the broker performs interaction with service proxies via SOAP. First, the broker lookups for suitable services from service registry. Service proxies register and update their state at the service registry using WSDL. Service registry maintains the available services using the standard specification (i.e., universal description discovery and integration (UDDI)) and responds to the broker's query. Then, through SOAP, the broker calls the related services to perform content adaptation tasks. While using these services, the service proxy's resources should be locked to avoid possible interruption and corruption by using any relevant locking mechanism [89].

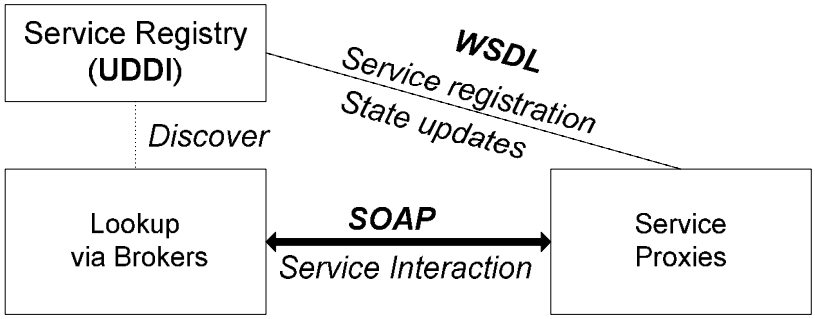

Figure 3.11: Local proxies-Web Services interaction model.

3.2.4.2 Related Protocols

SOAP: It is a protocol for messages exchange using HTTP/HTTPS or even (parallel) PHHTP [90]. It consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls (RPC) and responses [87].

UDDI: It is a XML-based registry for service providers to list their services and discover each other on the Internet. It is designed to be interrogated by SOAP messages and to provide access to WSDL documents describing the protocol bindings and message formats required to interact with the Web Services listed in its directory. Detailed information about UDDI can be found at [91].

Figure 3.12 depicts the typical port binding using socket number via the TCP layer between two machines. Socket number is a unique port number with machine IP address. To establish binding operation, the sending machine sends request together with the socket number, while the receiving machine acknowledges the binding and replies with the related socket number. Together, they need to maintain a port table consisting the active port numbers in a process called binding. Both machines will have reversed entries for each session between them, with which is referred as binding [92]. Note that the socket numbers of both machines is still unique as both IP addresses are not the same.

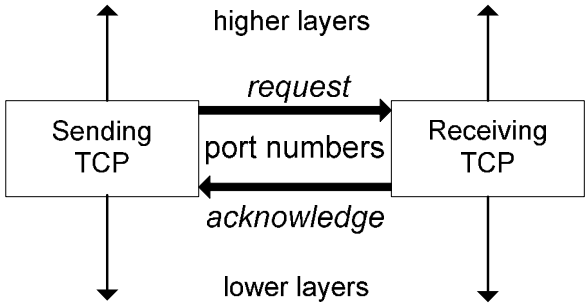

Figure 3.12: Port binding operation.

Figure 3.13 depicts the simultaneous service proxies' interaction with the local proxy for content passing or delivery. For example, the local proxy sends adaptation tasks to the service proxies through agreed destination port. Then, after performing the assigned tasks, both service proxies 1 and 2 send output to the same destination socket (same destination IP address and port) in a process called multiplexing.

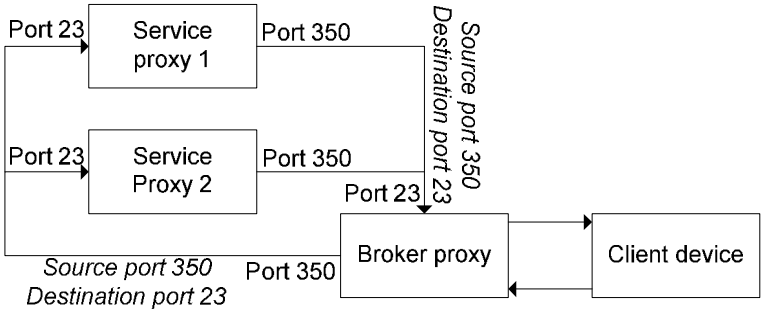

Figure 3.13: Simultaneous service's proxies' interaction for content passing.

Figure 3.14 shows the example of defining a translation adaptation service using WSDL. The document is provided in an XML format defining the ports and messages. Together with reusable binding, a port is associates with a network address. A message comprises of an abstract description of exchanged data [92]. Port types represent the collections of supported operation (in our work, it is called adaptation services). Altogether, WSDL describes the Web service public interface.

```
<binding name="TranslationBinding"
        type="tns:Translation">
    <soap:binding style="rpc"
              transport="http"/>
      <operation name="Translate">
        <soap:operation soapAction="translate"/>
          <input>
            <soap:body use="encoded"
                 Namespace="Translate"
             encodingStyle="soap encoding"/>
         </input>
      </operation>
</binding>
<service name="TranslationService">
  <port name="Translation"
     Binding="tns:TranslationBinding">
      <soap:address location="http://Translation"/>
      <state>
        <DescriptionName="……">
        </Description>
      </state>
  </port>
</service>
……
<nextDestination>
  <soap:targetAddressLocation="…"/>
</nextDestination> …… passing partially adapted content
```

Figure 3.14: An example of a translation adaptation service's definition using WSDL.

3.3 Analysis of Framework

In this section, we justify the strength of our proposed design in terms of reliability, scalability, extensibility, simplicity and portability.

In the application architecture perspective, reliability is viewed as the degree to which the architecture is susceptible to failure at the system level in the presence of partial failures within components, connectors, or data [93]. As such, service-oriented

approach improves reliability by avoiding single points of failure (that occur in centralized system), enabling replication between similar service proxies, allowing monitoring by the broker or third party monitoring apparatus, or even reducing the scope of failure to a recoverable action.

In term of score-based mechanism, our work differs from previous studies, where we consider the comparative approach to compute score, results in fairer score, rather than the baseline approach. In addition, we concentrate on more reliable model (here, we refer the reliability as the ability to reduce the chances to produce identical paths' ranking). Unlike [1, 13], in which this issue is neglected, we assume that having a reliable model is essential to increase the path determination performance (for instance, if we have 2 identical score for top two path options, then we need to have another decision making condition/rules, with which increases the decision complexity).

Next, we discuss the scalability of the architecture. We refer scalability as the ability of content adaptation architecture to support large numbers of components including interaction among them, within an active configuration [93]. As such, by simplifying components; by controlling interactions; and by distributing/decentralizing services across many components, scalability can be improved. In our design, these factors are influenced by determining available adaptation services and its state; the extent of adaptation tasks distribution and monitoring interactions; allowing new updates on service registries; and even a good balance on performing adaptation tasks between service proxies. This guarantees content adaptation on a best effort basis.

In addition to scalability, extensibility is also important. Extensibility is defined as the ability to add functionality to a system [94]. In our design, extensibility is induced by allowing new services to be listed in the service registry dynamically, without restructuring the architecture. New adaptation tasks or data types can be facilitated as long as the new Web Services is introduced. Furthermore, our generic rule-based for context mapping can be easily extended with new rules.

In our design, simplicity of the architecture is also considered. We refer simplicity as simplifying the functionality of each component in the architectural design. We induce simplicity by applying principle of separation of concerns to the allocation of functionality within components [93]. As such, each main component is separated into an independent module (e.g., ADTE, service discovery, path

determination). Moreover, each service provider can externally facilitate a particular adaptation task, thus enabling delegation of tasks. Web Services provide specific services, offer better adaptation services and becoming intelligent [81, 95].

Finally, we discuss the strength of the architecture in term of portability. Software is portable if it can run in different environments [96]. As such, we induce portability by the mean of platform neutral and providing message-oriented interface for public interaction.

3.4 Summary

In this chapter we proposed a comprehensive broker-based service-oriented content adaptation framework. Specifically, we discussed in detail the service-oriented architectural as it provide platform neutral message oriented interaction to the public on the Internet. A flexible platform for exchanging of the original content, partially adapted, or fully adapted content across the distributed proxies and server location using communication models (OCM and SCM) is proposed. These included the explanation on the interaction protocol the required components to manage clients' requests. Within ADTE, we explained in depth the context mapping, analysis using rule-based technique and task scheduling. We discuss the need for service discovery. Also, we proposed score tree scheme for determining best possible path. SLA is used to guarantee the client to get the required content version, delivered within the negotiated QoS. Then, we elaborated the Web Services, as the requirement for enabling content adaptation as services provided by third party service providers. Web Services interaction model; related protocols (UDDI and SOAP); and definition of services using WSDL, are discussed based on the theoretical and practical perspectives. We also provide an example of defining a Web service as an adaptation service.

Chapter 4

Service Discovery Protocol

In service-oriented content adaptation scheme, content adaptation functions are provided as services by multiple providers that located across wide area network. To benefit from these services, clients must be able to locate and discover them in the network. This makes service discovery as an important component. In this chapter, we develop the service discovery protocol for service-oriented content adaptation systems based on the industry standards. The protocol takes into account the searching space, searching time, QoS offered by the service providers and physical location of the potential service providers.

4.1 Introduction

Service-oriented content adaptation scheme makes use of the same standard specifications provided by several communities and institutions (e.g., W3C and OASIS Consortium). Content adaptation is provided by service providers to clients as Web services. This enables interoperability, scalability and platform independence. The common usage architecture for Web services describes three main entities: the consumer, the producer and the registry. These entities work together to perform publishing, searching and binding operations [61]. In the service-oriented content adaptation scheme, there are many services located across wide-area network. Such a variety of service types offers a variety of content adaptation functions. Adaptation functions offered include content aggregation, filtering, annotation, transcoding, translation, conversion and extraction. To benefit from these services, a client must be

able to both locate them in the network and invoke them. Service discovery enables these capabilities. To benefit from these services, a client must be able to both locate them in the network and invoke them. This makes service discovery as an important component of service-oriented content adaptation.

There are several issues need to be considered to provide service discovery for the service-oriented content adaptation scheme. A client's content request may be composed of multiple different content objects. To get a version that is tailored the specific contexts (e.g., client device, preference) from the original content; one or more content adaptation tasks are required [77]. Each service performs a particular content adaptation function that potentially be provided by multiple providers located across the wide-area network. These services need to be discovered first. Since, content adaptation is a task that should be performed promptly; a discovery protocol that quickly locates potential services is required. Also, it should take into account quality of services (QoS) offered by these service providers to match with the client QoS requirements. QoS includes criteria such as timing, distance, cost and reputation. In term of distance, the protocol needs to locate closer potential service providers to avoid such a high latency hops.

Although there are many service discovery protocols for service-oriented applications, there is none specifically developed for service-oriented content adaptation systems [1, 16, 18, 38, 97] or has solved the aforementioned issues simultaneously. The existing protocols (both pure keyword-based and QoS-based) tend to perform extensive searching (i.e., search for all available matched services from the corpus) which makes it time consuming and tedious. The work in this chapter aims to address the aforementioned issues.

4.2 Models

4.2.1 System Model

In the system, there are N content adaptation service providers $S = \{s_1, s_2, \dots, s_N\}$, $B = \{b_1, b, \dots, b_{|B|}\}$ service brokers and $SR = \{sr_1, sr_2, \dots, sr_i\}$ business service registries geographically distributed across wide area network. The service registries are based on UDDI reference model that allows for indexing, searching and retrieving

services through their descriptions. Also, UDDI enables service registries to be replicated.

A service provider may provide one or more content adaptation services $T = \{t_1, t_2, \dots, t_M\}$. The service providers advertise their services along with a set of $Q = \{q_1, q_2, \dots, q_K\}$ quality of services (QoS) in business service registries using web service description language (WSDL). QoS includes response timing, adaptation cost and availability. Table 1 provides the content adaptation service types along a brief description and examples.

A broker is responsible to obtain the required tasks in order to provide a content version required by the client and this can be achieved using the adaptation taking engine (ADTE). ADTE systematically analyses the required version with the original content is used to produce required content adaptation tasks. The output from ADTE is a composition of tasks. The broker then locates and selects potential services to perform these tasks.

We assumed that brokers and service providers have the knowledge of local registries, i.e., these registries are known a priori. Each broker keeps and maintains a list of local service registries information. It uses the pinger logic to check the actual reachability and the responsiveness of the service registry(s). The pinger logic uses the User Datagram Protocol to send an 18 bytes proximity measurement query (as UDP packet) of the format "PING time CRLF" with a timeout period of 1000 milliseconds. UDP does not require acknowledgement of packets received, which causes less messaging overhead than TCP [99]. Upon receiving the proximity measurement from each accessible service registry, the broker lists these registries in the ascending order of the proximity measurement.

4.2.2 Content Adaptation Request

There are many different services for content adaptation. Table 4.1 provides the content adaptation service types along brief description and examples.

Table 4.1: Example of content adaptation types.

Content adaptation task types	Function description	Example
Filter	Remove information, content	Remove audio track from the karaoke file
Annotate	Add information to content	Adding English subtitle into the movie file
Transcode	Change to different format (within the same content type)	Change an image format from BITMAP to JPEG extension
Translate	Change to different content language	Translate a web page with English text to Spanish text
Convert	Change to different content type	Convert a web page with English text into English audio
Extract	Extract keyword/ summarize from content	Summarize a long sport news into a short highlight

A request for content adaptation may require a series of services that may or may not be provided by the same service provider. The service broker is responsible to discover the services based on the user request requirements. An example of a content adaptation request is given as the following: Given an original movie file that is encoded in AVI format with 1200 by 960 resolutions and without subtitle. Suppose a version of the movie in MPEG4 format encoded in 800 by 480 resolutions with English subtitle, is required. To achieve the desired content form (i.e., a transcoded movie file with English subtitles), the content adaptation request will require at least two tasks: (a) video transcoding (i.e., format conversion and resolution resizing); and (b) English subtitle annotation. For each required task, the broker lookups for potential service from accessible registry. As a result, a set of services will be located to serve the request.

4.2.3 Matching Request and Service

During lookup, each service will be matched with the request requirements. The matching process compares the service's adaptation function and offered QoS with the required content adaptation task and QoS requirements from the request, in a manner similar to [98]. The matching type of each service is assigned according to service matching classification. Clients requirements are defined in the service level agreement (SLA) that both client and broker agree to and that the broker refers to when locating potential services.

Specifically, let R be the client requirements defined in the form of $R = (F_R, CQ_R)$, where F_R is the particular content adaptation task required and CQ_R is the set of QoS levels specified by the client. For positive monotonic QoS such as reputation and rating, the client may indicate a lower bound value. A potential service has its QoS equivalence or higher value than desired by the client. Meanwhile, the client indicates an upper bound value for negative monotonic QoS such as cost and time. On contrary, a potential service has its QoS equivalence or lower value than desired by the client. The matching category between the request and the service can be divided into three: match, partial match and non-match. Suppose that the client i requirements $R_i = (F_{Ri}, CQ_{Ri})$ are given, matching categories are defined as the following:

Definition 4.1 (Match category) Let a content adaptation function and QoS (both attribute q and its value $q.v$) of a service n be $F_{Sn}, Q_{Sn} \in S_n$. If $F_{Sn} \equiv F_{Ri}$ (i.e., accurate service's function and required task matching) and for each attribute $q \in Q_{Sn}$, it exists $q' \in CQ_R$ such that attribute $q = q'$and $q_k^{'}.v_{min} \leq q.v \leq q_k^{'}.v_{max}$, then it is a match. Note that the QoS matching operator is $\geq$ for positive monotonic QoS and $\leq$ for negative monotonic QoS.

Definition 4.2 (Partial match category) Given $F_{Sn}, Q_{Sn} \in S_n$. If $(F_{Sn} \equiv F_{Ri})$and only for some attributes $q \in Q_{Sn}$, it exists $q' \in CQ_R$ such that attribute $q = q'$and $q_k^{'}.v_{min} \leq q.v \leq q_k^{'}.v_{max}$, then it is a partial matching category. Specifically, partial match is assigned to the service where only parts of its QoS matched the client QoS requirements.

Definition 4.3 (Non-match category) Given $F_{Sn}, Q_{Sn} \in S_n$. If $(F_{Sn} \equiv F_{Ri})$and for each attribute $q \in Q_{Sn}$, it does not exists $q' \in CQ_R$ such that attribute $q = q'$and $q_k^{'}.v_{min} \leq q.v \leq q_k^{'}.v_{max}$, then it is a non-matching category. Also, if $F_{Sn} \not\equiv F_{Ri}$ (i.e., a service's function does not matched the required task), then it is a non-matching as well. Specifically, non-match is assigned to the service where none of its QoS value matched the client QoS requirements or the service's function unmatched the particular adaptation task.

4.2.4 Problem Statement

Given a set of $S = \{s_1, s_2, \dots, s_N\}$ content adaptation service providers, each with one or more content adaptation services $T = \{t_1, t_2, \dots, t_M\}$, the service discovery problem is to locate a set of potential service providers that will satisfy the required content adaptation functions and the client's QoS requirements.

Table 4.2 lists the common notations used throughout this paper.

Table 4.2: List of notations.

Notation	Description
S	Set of available services for the tasks T
T	Set of adaptation tasks
Q	Set of the service QoS
CQ	Set of the client QoS
s_{ij}	Service j for task i
SR	Set of service registries
P	A path of composite services serving T
N_T	number of the current services being searched
N_{MIN}	minimum number of services to be searched
$\bar{S}$	match category services
S_i^b	partially matched service
$\bar{S}_{max}$	maximum number of services in $\bar{S}$
$\bar{S}_{count}$	the number of services currently store in $\bar{S}$
T_{start}	start time when the broker's agent arrived at a particular registry
T_{wait}	the agent's waiting time at a particular registry to get respond
$T_{respond}$	maximum waiting time specified by the broker

4.2.5 Performance Metrics

In this chapter, we will use the discoverability metric to measure the performance of the proposed service discovery protocol. Discoverability metric D is formulated as given in equation below.

$$D = (\alpha.w_\alpha + \beta.w_\beta + \gamma.w_\gamma)/n \ . \tag{4.1}$$

where α is the search space, β is the aggregate match type of the discovered services, γ is simulation search time, w_f is the weighting for each factor where $\sum w = 1$, and n is

the number of considered factors. The discoverability metric is developed based on the observation of the following factors.

If the search space (of providers) is increased, the better assessment is made. Hence, it is directly proportional to discoverability metric i.e., discoverability is better if search space is wider. The equation is given as the followings:

$$\alpha_D = \left(\sum_{t=1}^{T}\left(\frac{N_T}{N_{total}}\right)\right)/T. \quad (4.2)$$

where N_T is the number of searched providers by the end of search execution for a particular task, N_{total} is the number of available providers for a task and T is the number of content adaptation tasks required.

If the matching category of the discovered service providers belongs to match; the better client will be served, as the client is provided with exact or better QoS levels than the requirements and with the accurate service. This implies that better trade value is achieved by the client. Hence, matching category is directly proportional to discoverability i.e., discoverability is better if returned service(s) matching category is match. In addition, both match and partial match categories return services (including its function) that accurately matched the required task (definitions 4.1 and 4.2). The equation is given as the followings:

$$\beta_D = \left(\sum_{t=1}^{T}\left(\frac{\sum_{s=1}^{S_n} value}{S_n}\right)\right)/T, \text{ where } match\ value = \begin{cases} 1 & match \\ 0.5 & partial \\ 0 & non\ match \end{cases}. \quad (4.3)$$

In term of searching time T_{ST}, generally, the longer searching time is taken, the later service providers will be discovered. Also, longer searching time relatively increases the amount of time required to provide the client with adapted content version. However, instead of penalizing the approach with longer searching time, we incentivize the one with shorter searching time. Hence, shorter searching time is directly proportional to the discoverability i.e., discoverability is better if searching time is shorter. The equation is given as the followings:

$$\gamma_D = \frac{T_{ST}min}{T_{ST}}. \quad (4.4)$$

where $T_{ST}min$ is the minimum searching time between protocols being compared and T_{ST} is the searching time for the particular protocol.

4.3 Service Discovery Protocol

4.3.1 Architecture

Figure 4.1 presents the layered service discovery architecture composed of a service layer and a look-up layer. The interactions between entities are performed according to the protocols defined by XML-based (Extensible Markup Language) standards. XML-based representation is used for extensibility and openness.

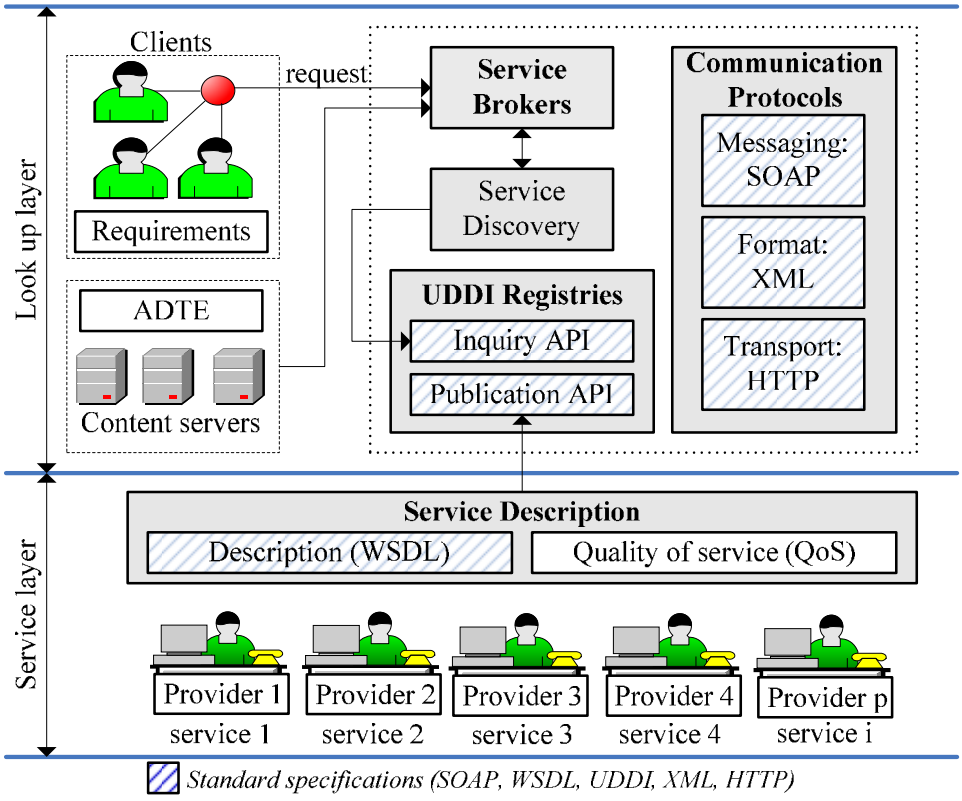

Figure 4.1: Service discovery architecture for service-oriented content adaptation.

There are N content adaptation service providers $S = \{s_1, s_2, \dots, s_N\}$, $B = \{b_1, b, \dots, b_{|B|}\}$ service brokers and $SR = \{sr_1, sr_2, \dots, sr_i\}$ business service registries geographically distributed across wide area network. The content of the service registries is regarded to appropriately being partitioned in disjointed sets thus; no service duplication filtering is required.

At the service layer, the service provider publishes their offered services. The publication API is used by the provider to publish and update service descriptions. Update operation include create, modify and delete. The description must comply with the UDDI data model (specified in UDDI version 2 schema). It consists of four core types of information: *businessEntity*, *businessService*, *bindingTemplate* and *tModel*, stored as a set of attribute-value pairs. A *businessEntity* (i.e., the service provider) contains information about the provider publishing the services and the collection of *businessService* (i.e., the service), with descriptive data (e.g., service function, QoS, handle, pre-conditions and post-conditions, input and output, effect) about each service. Each *businessService* is related to a *bindingTemplate* element that describes the technical information necessary to use a service (e.g., unique access point or handle for invoking the service). Additional *publisherAssertion* data can be used to declare relationship between two business entities. The *bindingTemplate* contains references to *tModel* that allows classification through description for services and taxonomies. *tModel* can be used to enriched service specification to provide detail searching. New technologies like semantic description (e.g., Web ontology language-OWL) and QoS description (e.g., OWL-Q, DAML-QoS) are used for rich specification.

Look-up layer contains service discovery and communication protocols. Communication protocol is responsible to assist communication between brokers, registries and providers. The communication message is formatted according to XML schema and transported through Internet using protocols such as HTTP or SMTP. A client may initiates one or more requests. Each request is sent to the broker. For each request, the broker uses ADTE to produce required tasks. Using this output, the broker initiates the service discovery protocol. Figure 4.2 presents the detailed service discovery protocol.

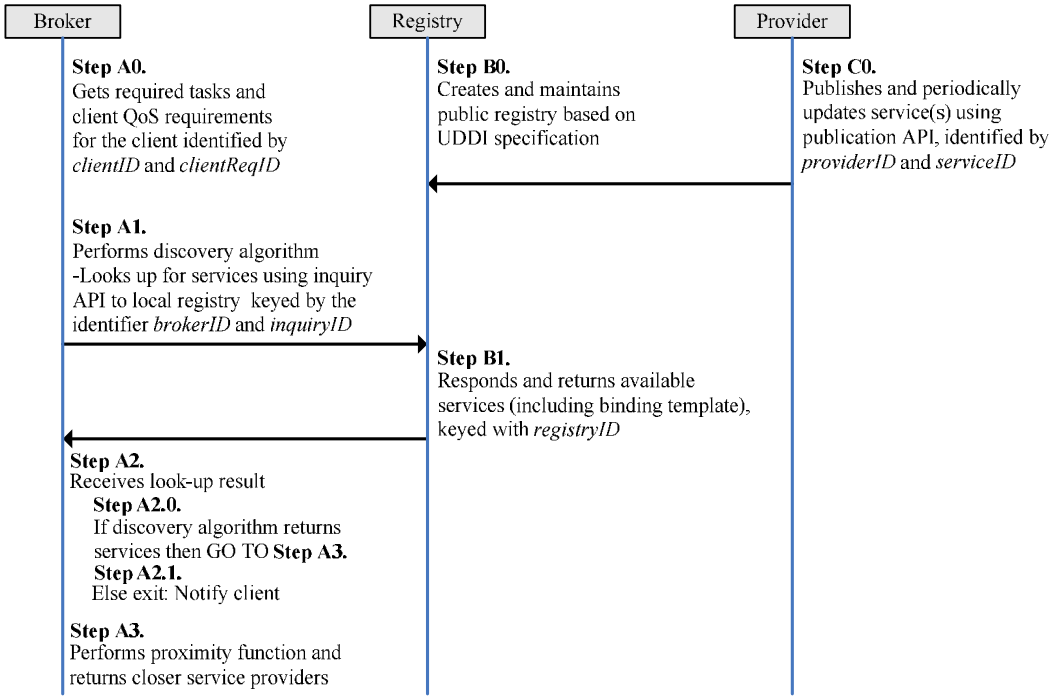

Figure 4.2: The proposed service discovery protocol.

Upon receiving the required tasks for the specific client including the corresponding QoS requirements (*Step A0*), the broker performs the discovery algorithm and initiates look up to the accessible local service registry(s). Each request is uniquely identified by the client ID and the client's request ID. The broker has to correctly store this detail and keeps track of each look up inquiry to the registry(s).

The registry is responsible for caching and maintaining services' advertisements and allows for service look up (*Step B0*). Providers are responsible to publish and periodically update their service(s) (*Step C0*). Through inquiry API, the broker sends a runtime message using SOAP version 1.2 via HTTP protocol to UDDI public business XML-based registry, to perform browse and drill-down query (*Step A1*). Each inquiry made by the broker is identified by the broker ID and the inquiry ID. The inquiry API (implemented by UDDI API version 2) allows the broker to locate and obtain details entered in a UDDI registry. The browse and drill-down query combines 'find' operation and 'get' operation made on the key attribute associated with data retrieved.

The query message contains the look up for available services for performing the content adaptation tasks. The query includes information for specific adaptation functions, QoS levels, IP address and binding template (i.e., handle). At the registry, it

responds to the query made by the broker and returns the message containing services and related information (including the binding template), keyed by the registry ID (*Step B1*). The binding template provides the command at which a program can start interacting with the service [61].

Upon receiving the reply message from the registry (*Step A2*), the following assessment is carried: if the discovery algorithm returns services (*Step A2.0*), the broker performs proximity assessment function and returns potential service providers in proximity order (*Step A3*). Otherwise, the broker informs the client that content adaptation is not performable due to the service unavailability and terminates its operation (*Step A2.1*). The potential service providers can be automatically invoked using their binding templates.

Algorithm 4.1: Service Discovery

INPUT: *T, S*, S*R*
OUTPUT: $\bar{S}$ for all tasks
BEGIN
1: Initialization
2: **FOR** ($sr_i \in SR;\ i = 1; 1{+}{+}$) **DO**
3: **Registry accessibility assessment**
4: **FOR** each service **DO**
5: **DO**
6: Retrieve service (s_i)
7: Find service matching category
8: $\bar{S}$ **update**
9: Increase N_T by 1
10: **WHILE** (($\bar{S}_{count} \leq \bar{S}_{max}$) **OR** ($N_T \leq N_{MIN}$))
11: **IF** (($N_T = N_{MIN}$) **AND** ($\bar{S}$ = empty))**THEN**
12: Append partial match service S_i^b into $\bar{S}$ //constraint relaxation
13: **END IF**
14: Attach $\bar{S}$ with proximity measurement and IP address
15: **Proximity assessment**
16: **END FOR**
17: **END FOR**
18: **WHILE** (($\bar{S}_{count} \leq \bar{S}_{max}$)
19: **RETURN** $\bar{S}$ for all tasks
END

Figure 4.3: Service discovery algorithm.

Figure 4.3 outlines the pseudo-code of the service discovery algorithm. The inputs to the algorithm are a set of required services and a set of available service providers for

each task published at the registry. At the initialization, the algorithm sets number of services in the sorted list $\bar{S}_{count}$, timer *t*, partial match service S_i^b and number of searched services N_T to zero. Also, the broker sets the acceptable waiting period for each requested registry to respond $T_{respond}$, number of minimum services to be searched N_{MIN} and the maximum number of match category services $\bar{S}_{max}$. The service's description to be retrieved includes the service's function, QoS, handle and IP address.

It then dispatches the broker's agent to the first nearest accessible business registry. The agent gets executed at the registry and then moves to the next registry for services collection. Upon arrival at the registry, the agent performs the *registry accessibility assessment*. It sends a drill-down query for information. The agent reads and stores the current time upon arrival as T_{start} and calculate time to get reply $T_{wait} = T_{start}$ + timer *t*. If the waiting time T_{wait} is bigger than $T_{respond}$, it aborts query execution; otherwise, for each task, it starts retrieving services advertised by providers at the registry, until $\bar{S}_{max}$ or N_{MIN} is achieved. For each service retrieved, it evaluates the matching category based on definitions 1 to 3, performs $\bar{S}$ *update* algorithm (as given in figure 4.4) and updates the increment of N_T by 1. If either of both conditions is met (i.e., $(\bar{S}_{count} \leq \bar{S}_{max})$ **or** $(N_T \leq N_{MIN})$), the algorithm breaks the DO loop (line 5). Consequently, this will terminate searching from the consecutive registries as the number of matched services already maximum (line 18).

Then it carries the following assessment: if either N_T is equal to N_{MIN} or the match category service list $\bar{S}$ is empty; the partial match service from S_i^b will be appended into $\bar{S}$ (i.e., QoS relaxation is imposed). Otherwise, it attaches the providers' proximity measurement and handle for each service stored in $\bar{S}$. Then, the discovery algorithm performs the proximity assessment function. The look up and proximity assessment processes are repeated for each task and this will eventually return $\bar{S}$ for all tasks.

Figure 4.4 depicted the $\bar{S}$ *update* algorithm. Upon receiving the input (i.e., service and its category) from the main discovery algorithm, it performs the following assessments. Matching services are appended into $\bar{S}$. The $\bar{S}_{count}$ counter is increased by 1. The first service that belongs to partial match category is stored into S_i^b including it's $s_i.q_{total}$as $S_i^b.q_{total}$, if it is empty. If a partial match service already stored in S_i^b, it will be replaced by the current partial matching if $S_i^b.q_{total}$ lesser than $s_i.q_{total}$ of the current partial service. The non-match or partial match (with lesser matched QoS)

service will be discarded. For partial matching service, it records the number of matched QoS into $s_i.q_{total}$.

Algorithm 4.2: $\bar{S}$ Update

INPUT: service and its category
OUTPUT: Updated $\bar{S}$
BEGIN
1: **IF** ((category = match) **THEN**
2: Append and sort service into $\bar{S}$
3: Increase $\bar{S}_{count}$ by 1
4: **ELSE IF** ((category = partial) **AND** ($s_i.q_{total} > S_i^b.q_{total}$)) **THEN**
5: Replace service into S_i^b
6: **ELSE**
7: Discard service (s_i)
8: **END IF**
END

Figure 4.4: Update $\bar{S}$ algorithm.

To perform the proximity assessment function, proximity measurement and physical location of each potential provider is required. There are several methods of attaining proximity measurement such as using estimation and pinger logic. The estimation method utilizes the requirement for each provider to update their service(s). When update is required, the registry initializes the timer at its end and waits until the update is received from the provider‘s end. The registry takes the average time between several updates for a more precise proximity measurement. On the other hand, the registry can use the pinger logic to measure the provider’s proximity in the same manner the broker measures proximity to the registry. The service provider’s physical location can be obtained by translating its IP address. IP translator tool [100] provides accuracy up to 90% of country-level granularity, given that each service proxy is located at the registered IP address. Then, the function sorts service providers in ascending order of proximity measurement and groups them according to their country of origin, and immediately returns $\bar{S}$ in this form to the main discovery algorithm. In this way, a set of closer providers can be located in the network. Finally, the discovery algorithm returns the list $\bar{S}$ for each task to the broker.

Upon receiving the output from the discovery algorithm, for each task, the broker can randomly select one of them from $\bar{S}$, especially the one with the higher order and

located within the same locality (with potential providers for other tasks). Table 4.3 provides an example of the result returned by the proposed service discovery protocol (from a single registry) for the case of three tasks with $\bar{S}$ equal to 5 for each task. For each provider, the protocol attaches the corresponding handle to enable further communication between the broker and each service provider.

Table 4.3: Example of result returned by the proposed protocol.

Order	$\bar{S}$											
	Task 1				Task 2				Task 3			
	Provider ID	Match type	Proximity (ms)	Country	Provider ID	Match type	Proximity (ms)	Country	Provider ID	Match type	Proximity (ms)	Country
1	S12	M	15	AU	S11	M	20	AU	S04	M	10	AU
2	S13	M	16	AU	S23	M	23	AU	S33	M	11	AU
3	S14	M	16	AU	S17	M	25	AU	S06	M	12	AU
4	S22	M	19	AU	S26	M	27	AU	S13	M	24	SG
5	S25	M	25	SG	S10	M	27	AU	S35	M	35	MY
Legend: M = matching, AU =Australia, SG = Singapore, MY = Malaysia												

Based on the example given in table 4.3, the broker can randomly select either one of S12, S13, S14 or S22 for task 1; S11, S23, S17, S26 or S10 for task 2; S04, S33 or S06 for task 3. All these providers are located within the same country i.e., Australia. Eventually, this will give a set of closer providers. Alternatively, the broker can use QoS criteria to select the best possible composition of service providers in a manner presented in the next chapter (Chapter 5).

If at the same time a given broker received requests from multiple clients, searching operation during service discovery can be optimized. Specifically, optimization can be performed in two ways. First, if clients' requests arrived at the same time or almost same at the broker's end, the broker only initiates one query to the registry instead of two. For example, assume the broker received requests from two clients, at the same period of time. Each client required two tasks. Further assume that the two tasks for both clients are different. In this case, the broker only initiates one query to the registry looking for services for four tasks, instead of initiating two

different queries i.e., message optimization is performed. In this way, the number of message communication is significantly reduced.

Second, the broker carries assessment for condition 4.1. The assessment of condition 4.1 can be conducted during (*Step A2*) of the service discovery protocol. Query optimization however, might not be feasible as QoS requirements between clients may vary even for a similar task. If only condition 4.1 holds, then the broker combines the searching for two tasks (from two clients) into searching for one task.

Condition 4.1. Let client A and B requirements be $F_{RA}, CQ_{RA} \in R_A$ and $F_{RB}, CQ_{RB} \in R_B$, respectively. If $(F_{SA} \equiv F_{RB}) \wedge (\forall q \in CQ_{RA} \approx \forall q \in CQ_{RB})$ then the searching for both tasks can be combined into searching for one task i.e., query optimization.

Consider the same example: the broker received requests from two clients, at the same period of time; each requiring two tasks. Further assume that for both clients, one task (and its QoS requirements) is similar. Likewise, the broker only initiates one query to the registry looking for services for three tasks, instead of looking for four tasks i.e., query optimization is performed. In this way, the number of message communication and searching are reduced.

The proposed discovery algorithm has several strengths. It finds services that matched the client requirements. After specified search space or a number of match category services is achieved, the algorithm is quickly terminated. Thus, it avoids the algorithm to perform extensive searching. This significantly reduces searching time. Also, it returns sets of closer composite providers to select from. Also, if required, to check actual responsiveness of the top services, it requires RTT (round trip time) measurement to only a small number of providers.

4.3.2 Analysis of Protocol

In this subsection, we will discuss the analysis of the proposed discovery protocol in terms of service accuracy, service completeness and closer provider. Assume network connectivity exists between the broker and the corresponding registries during interaction.

4.3.2.1 Service Accuracy

Theorem 4.1: Service Accuracy - The proposed discovery protocol finds services that satisfy the requesting client's requirements in a finite time if the requested services are published in the looked up registry(s).

Proof: Let us assume that the discovery protocol cannot find the requested services for the requesting client even though there are services that satisfy the requesting client's requirements and published in the looked up registry(s). Each provider publishes its service(s) at the registry(s) and updates their services periodically as in *Step C0*. Each registry maintains and publishes services accordingly as in *Step B0*. On behalf of the client, the corresponding broker performs a drill-down query using inquiry API for services (that capable of performing required tasks) at the local registry(s) as in *Step A1*. The registry checks for the particular service(s) and replies to the requesting broker with the result (that includes the services) as in *Step B1*. If the broker does not receive reply within the predetermined respond time $T_{respond}$ from the registry, it aborts query execution. Otherwise, the algorithm receives the reply (*Step A2)* and performs matching operation as in *definition 4.1 to 4.3.* Specifically, it returns the services including the provider detail that satisfies adaptation function and the client QoS requirements as in *Step A2.0*. Otherwise, the discovery protocol returns *exit message: notify client* as in *Step A2.1*. Thus, the broker will receive a reply of the result in a finite time. Therefore, the broker will eventually find the requested services published in the looked up registry(s) satisfying the client's requirements in a finite time. This is a contradiction. Therefore, our assumption is false and the theorem is proved. □

4.3.2.2 Service Completeness

Theorem 4.2: Service Completeness – Assume that there are many services published in the looked up registry(s) that satisfy the client requirements; the broker will receive the information of all the searched services that matched the requirements (including its handle) for all tasks in finite time until specified search space is achieved or match category service list $\overline{S}$ is full.

Proof: Let us assume that the proposed discovery protocol cannot find all the searched services published in the looked up registry(s) that satisfies the requesting client's requirements for all tasks in a finite time even though the registry(s) replied. For each broker, the proposed protocol constructs correct request and reply as established in Lemma 1. The broker aborts query execution at the particular registry if it does not receive reply within the predetermined respond time $T_{respond}$, otherwise it receives the corresponding reply from the registry. Since the accessible registry(s) replied to the request made by the broker through the inquiry API (*Step B1)*, the requesting broker will receive the reply message including services as in *Step A2*. This includes the service's handle used to invoke the service provider. Then the broker uses the algorithm (i.e., matching operation) to categorize services according to their matching category (*as in definition 4.1* to *4.3*). The algorithm will return all searched services satisfying *definition 4.1* as in *algorithm 2: line 1*, until specified search space is achieved or $\overline{S}$ is full *algorithm 1: line 10*. If no match category service exists for a particular task, the algorithm returns a partial match service (*definition 4.2*) as in *algorithm 1: line 11*. This enables the request to be served. Thus, the broker will find all matched services (including its handle) published in the looked up registry(s) satisfying the client's requirements for all tasks in a finite time, until specified search space is achieved or $\overline{S}$ is full. This is a contradiction. Therefore, our assumption is false and the theorem is proved. □

Lemma 4.1: Correct multiple requests and replies – In the proposed service discovery protocol, if there is more than one broker (each representing one or more clients) or more than one registry (each with unique *registryID*), then it participates correctly in the communication of multiple requests and replies to or from the broker and registry(s) simultaneously.

Proof: Let us assume that the proposed discovery protocol cannot provide correct multiple requests and replies, to and from different brokers and registries even though every unique ID (i.e., *clientID*, *clientReqID*, *brokerID*, *inquiryID* and *registryID*) is included in each message. In the proposed protocol, every client request to the representing broker has a unique *clientID* and *clientReqID* as in *StepA0*. The broker

submits each query to the registry identified by *brokerID* and *inquiryID* as in *StepA1*. Each registry replies for the specific *inquiryID* and *clientID* with its *registryID* as in *StepB1*. Thus, each request will have correct reply, and eventually multiple requests from different brokers are getting the correct reply(s) from the corresponding registry(s). This is contradiction. Therefore, our assumption is false and the lemma is proved. □

4.3.2.3 Closer Providers

Theorem 4.3: Our proposed discovery protocol always returns closer service providers from list $\overline{S}$ in a finite time.

Proof: Let us assume that the proposed discovery protocol is unable to return closer service providers from list $\overline{S}$ in a finite time. Let $\overline{S}$ stores the match category services or partial match service (if no match category is found) for each task returned by the algorithm as in *algorithm 1: line 18*. The algorithm retrieves providers' details of all services in $\overline{S}$ including the proximity measurement and IP address as in *Step A1* and receives the reply message including the proximity measurement and IP address details as in *Step B1*, within the predetermined respond time $T_{respond}$. The proximity measurement (from the provider and the registry) and IP address are used to estimate the provider's proximity and physical location. The proximity measurement provides the network proximity between the provider and the registry while the IP address provides the provider's physical location (i.e., country of origin). Then the algorithm sorted the service providers based on proximity measurement and grouped by country of origin as in *proximity assessment function*. The broker then selects one service provider for each task that located in the same country and closer to the registry. This eventually gives the broker a set of closer providers. Hence the proposed discovery protocol will always returns closer service providers from list $\overline{S}$ in a finite time. This is a contradiction. Therefore, our assumption is false and the theorem is proved. □

4.4 Performance Evaluation

We formulated a metric that computes the discoverability of the service discovery protocol. The discoverability metric [0…1] quantitatively expresses the mixture of three

important factors: searching time, number of searched service providers and the matching category of each service returned for a discovery protocol to serve a given request. Unlike discoverability metric defined in [101], ours incorporate the service's matching category factor and is evaluated for request with multiple tasks i.e., multiple service types.

As experiments may not fetch searching time from protocols being compared and lead to difficulty of estimating total available services in the registry(s), we simulate the system with all possible cases considering variations in multiple factors: search space, matching type of each returned service provider for each service, and simulation searching time. Also, simulation allows for the simulated environment to be controlled and the exact setting to be repeated. We assume that broker is capable of performing lookup. Three different simulations were conducted to study the discoverability metric towards (1) number of tasks (i.e., content adaptation services), (2) number of service providers, and (3) number of QoS. These variations are chosen to evaluate scalability and reliability of the proposed protocol compared with others.

We followed the verification methodology described in [102]. For each simulation, 20 runs were conducted and the mean value v_m is taken. The confidence interval is 95% i.e., with confidence level 95%, any of the run lays in the mean value v_m interval (with 5% margin is observed). At each run, we generated the number of tasks (T) to be between 1 to 5. We set the number of services (S) for each task in the range of 5 to 20. The total number of QoS (Q) for a particular task is set in the range of 1 to 4. Data to represent QoS values is generated in a manner similar to [13]. This is practical to represent QoS values between service providers [5]. The value we used for each parameter is in line with the current literature and also reflects the actual environment. The number of tasks and QoS are in line with the work of [1, 5]. The number of service providers is chosen based on [1].

We used two well known service discovery protocol to compare the proposed approaches with. Keyword-based protocol (e.g., Sun's Jini and IBM's Salutation) and more recent QoS-based protocol (e.g., DAML-QoS [75], Mixed Integer Programming [68]) are chosen because they are widely accepted and comparable to our policy (i.e., wired network application). A keyword-based protocol is characterized with the ability of matching the adaptation function. On the other hand, a QoS–based discovery

protocol has the ability to return services that matched required adaptation functions and the client QoS requirements. Both protocols perform extensive searching.

4.5 Results and Discussion

Extensive simulation analysis of the proposed algorithm has been carried out. Discoverability depicts the capability of the compared protocols towards reducing searching time while locating matching category services within adequate search space.

Figure 4.5 shows the discoverability ratio (y axis) as a function of the tasks (x-axis). In this simulation, we varied the number of tasks from 1 to 5. The number of registries, services, QoS, sorted list $\bar{S}$ and N_{MIN} are set to 1, 10, 1, 5 and 5 respectively. The service matched type is also randomized (matching type is to be between 70% to 90%). As can be seen from figure 4.5, there is a very small decrement of the discoverability for the proposed algorithm compared to others along x-axis. The proposed discovery algorithm provides higher discoverability while keyword-based algorithm provided the least. The proposed algorithm constantly produces around 85% of the discoverability. There is a considerable different of 8% average between the proposed algorithm with QoS-based and average of 15% between the proposed algorithm with keyword-based along x-axis. This figure indicates that the proposed algorithm is more stable towards task variations compared to others. This is due to early termination when specified search space is achieved (N_{MIN} is met), thus minimizing search time, while at the same time returning matching type services. For others, extensive searching has resulted the considerable increasing decrement when number of tasks is increased. In addition, keyword-based suffers the most as it tends to return every single service regardless the matching type. Task variations have a considerable impact on keyword-based and QoS-based.

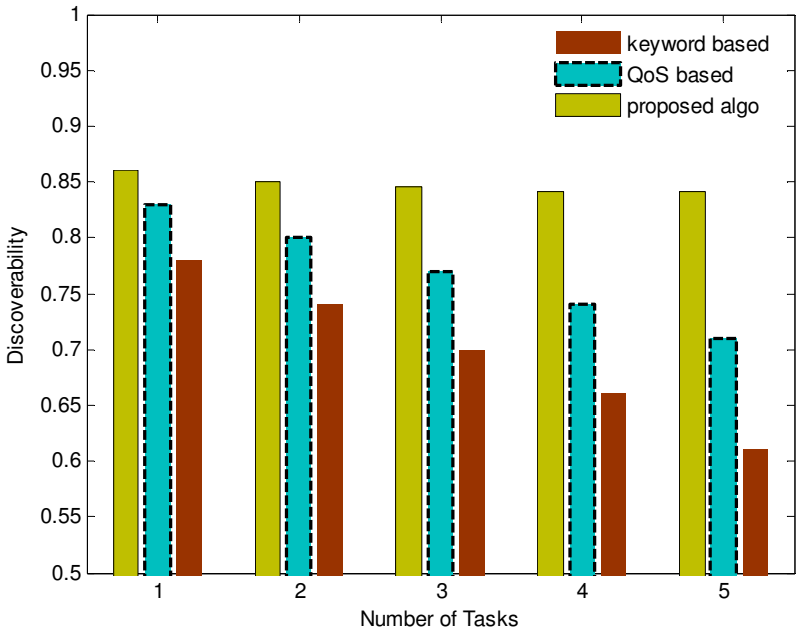

Figure 4.5: Algorithm discoverability towards task variations.

Figure 4.6 shows the discoverability ratio (y axis) as a function of the service providers (x-axis). In this simulation, we varied the number of service providers from 5 to 20. The number of registries, tasks, sorted list $\bar{S}$ and QoS are set to 1, 2, 5 and 1 respectively. N_{MIN} is set to be rounded half of total available services and the service matched type is randomized (matching type is to be between 70% to 90%). As can be seen from figure 4.6, there is a very small and small decrement of the discoverability for the proposed discovery algorithm and others respectively, along x-axis. The proposed algorithm generated higher discoverability (around 84%) while keyword-based algorithm provided the least (around 70%). The reason behind this is because keyword-based approach searched and returned all available services that contain unmatched category. There is a small different of 6% between the proposed algorithm with QoS-based, and 15% between the proposed algorithm with keyword-based along x-axis. The slight decrement between the proposed algorithm with QoS-based (compared to figure 5) is due to the fact that the proposed discovery algorithm quickly terminate (N_{MIN} is met or sorted list is full) even though number of providers is increased along x-axis. The simulation implies that number of services has a minor impact on discoverability of all algorithms. It is worth noting however, if N_{MIN} is constant (as observed in different

simulation setting); the proposed algorithm will experience a slight discoverability decrement when the number of providers increases along x-axis.

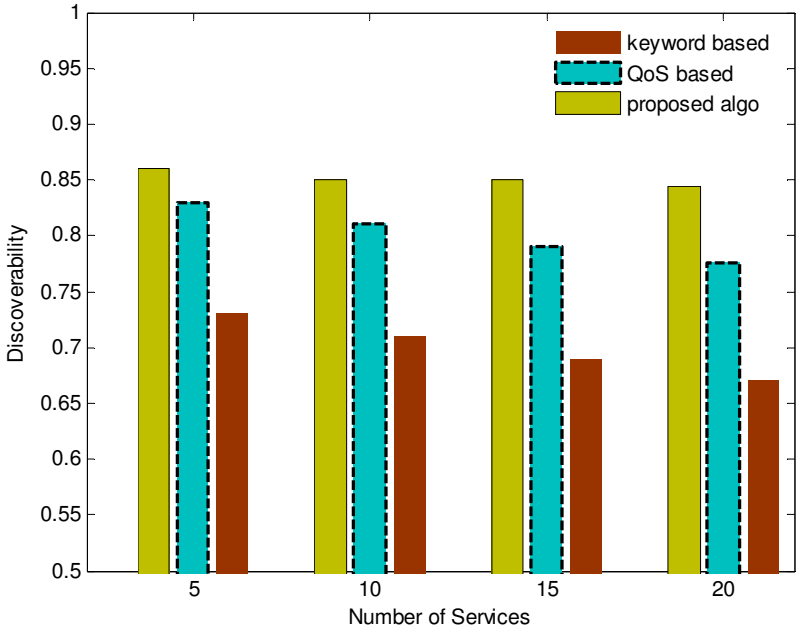

Figure 4.6: Algorithm discoverability towards service provider variations.

Figure 4.7 shows the algorithm discoverability ratio (y axis) as a function of the QoS (x-axis). In this simulation, we varied the number of QoS to be 1 to 4. The number of registries, services, tasks, sorted list $\bar{S}$ and N_{MIN} are set to 1, 10, 2, 5 and 5 respectively. The service matched type is randomized (matching type for each QoS is to be between 50%). As can be seen from figure 4.7, there is a very small decrement of the discoverability for all algorithms along x-axis. The proposed algorithm generated higher discoverability while keyword-based algorithm provided the least. The proposed algorithm constantly produces around 84%. There is a considerable different of 5% between the proposed algorithm with QoS-based, and 15% between the proposed algorithm with keyword-based along x-axis. This figure indicates that the proposed algorithm is more stable towards QoS variations while QoS-based affected the most. This result is due to the fact that when number of tasks and services remain constant, number of QoS increment has a minor impact on both the proposed algorithm and keyword-based. For keyword-based, this is due to the fact that it returns services without considering QoS, thus the searching time and match type remain stable along x-

axis. On the other hand, the proposed algorithm always returns services that match the client requirements or returns a service that partially matched the requirements if no matching type available. For QoS-based, the QoS increment constantly increases searching time along x-axis.

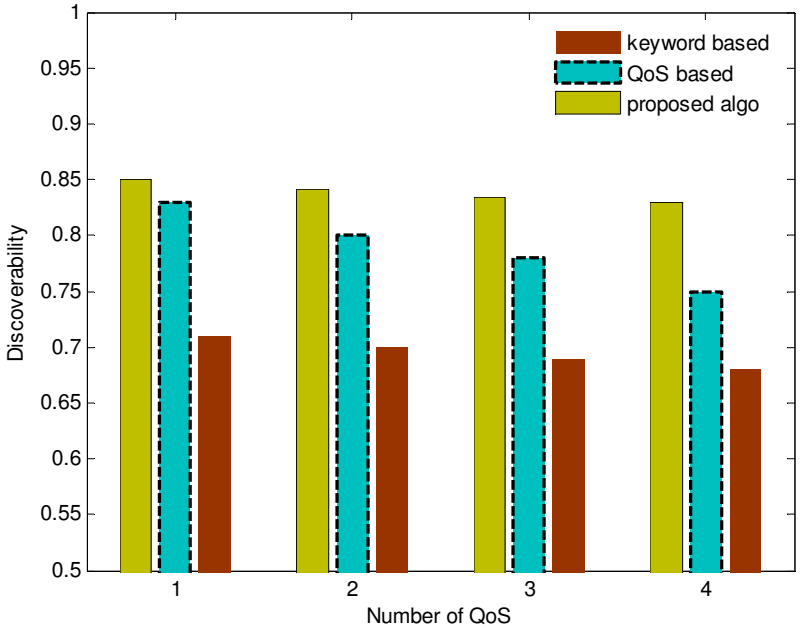

Figure 4.7: Algorithm discoverability towards QoS variations.

Figure 4.8 shows the algorithm discoverability ratio (y axis) as a function of the client (x-axis). In this simulation, we varied the number of clients (arriving at the same time) per broker from 1 to 5. The number of registries, services, QoS, tasks, broker, sorted list $\bar{S}$ and N_{MIN} are set to 1, 10, 2, 1, 1, 5 and 5 respectively. The service matched type is also randomized (matching type is to be between 70% to 90%). For each additional client, one task with the corresponding QoS requirement is set to be similar. All algorithms perform searching optimization when possible. As can be seen from figure 4.8, there is a very small decrement of the discoverability for the proposed algorithm compared to others along x-axis. The proposed discovery algorithm provides higher discoverability while keyword-based algorithm provided the least. The proposed algorithm constantly produces around 85% of the discoverability. There is a considerable different of 10% average between the proposed algorithm with QoS-based and average of 20% between the proposed algorithm with keyword-based along x-axis.

This figure indicates that the proposed algorithm is better towards client variations compared to others. This is due to the fact that for each additional client, all algorithms added the query with one additional task. The proposed algorithm quickly terminates when specified search space is achieved (N_{MIN} is met) for all tasks, thus minimizing search time, while at the same time returning matching type services. For others, extensive searching has resulted the increasing decrement when number of tasks is increased even though searching optimization is performed.

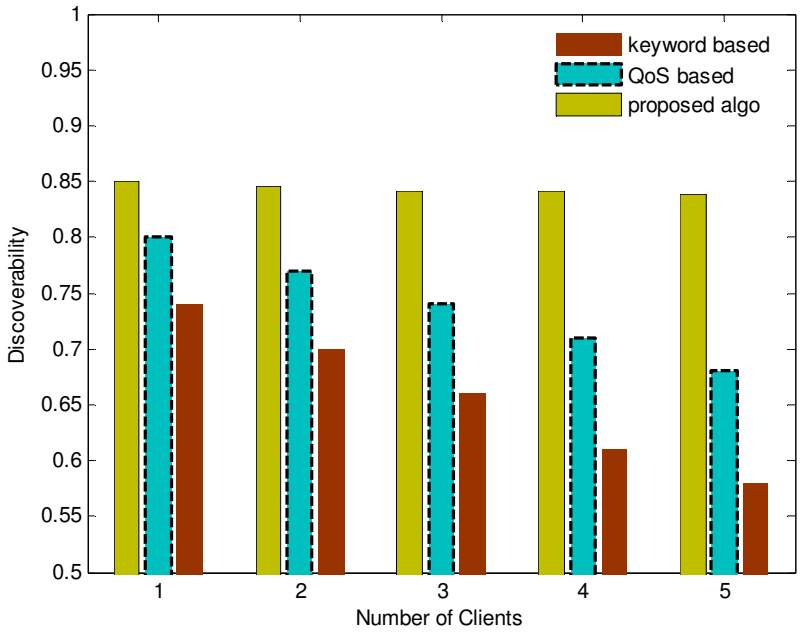

Figure 4.8: Algorithm discoverability towards client variations.

Figure 4.9 shows the discoverability ratio (y axis) as a function of the registries (x-axis). In this simulation, we varied the number of registries from 1 to 3. The number of services, tasks, sorted list $\bar{S}$ and QoS are set to 10, 2, 5 and 1 respectively. N_{MIN} is set to be rounded half of total available services and the service matched type is randomized (matching type is to be between 70% to 90%). As can be seen from figure 4.9, there is a very small and small decrement of the discoverability for the proposed discovery algorithm and others respectively, along x-axis. The proposed algorithm generated higher discoverability (around 84%) while keyword-based algorithm provided the least (around 70%). The reason behind this is because keyword-based approach searched and returned all available services from all registries including the one that

contain unmatched category. There is a small different of 6% between the proposed algorithm with QoS-based, and 14% between the proposed algorithm with keyword-based along x-axis. The slight decrement between the proposed algorithm with QoS-based is due to the fact that the proposed discovery algorithm quickly terminate (N_{MIN} is met or sorted list is full) even though number of registries (i.e., without visiting the consecutive registry(s)) is increased along x-axis. The simulation implies that number of registries has a minor impact on discoverability of the proposed algorithm due to the termination condition.

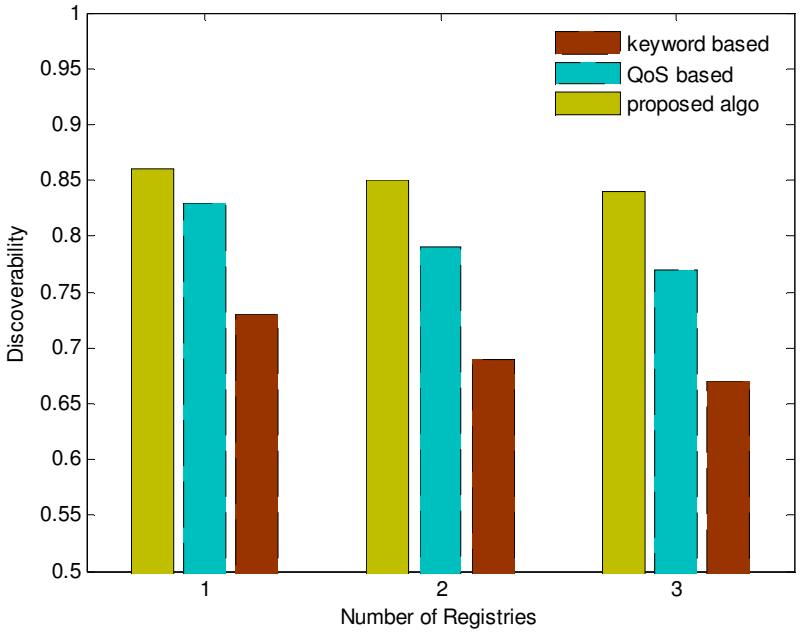

Figure 4.9: Algorithm discoverability towards registries variations.

Taken as a whole in these five simulations, some key findings were observed. First, the discoverability for all algorithms is relatively decreased for task, service, QoS, registry and client variations along x-axis. Second, there is a very slight discoverability decrement for the proposed algorithm compared with QoS-based and keyword-based algorithms for all variations along x-axis. The proposed algorithm outperforms QoS-based by 7%, and keyword-based by 15%, in average. This is due to certain factors. First, it has the same capability with (1) the keyword-based approach in term of locating service(s) that matched the required adaptation function(s), and (2) the QoS-based approach in term of discovering potential services that matched client QoS

requirements. The non-functional aspect matching (both for the proposed algorithm and QoS-based) does not notably affect the searching time if being executed in parallel [68]. Second, it has the advantageous feature that quickly terminates searching when specified search space or a number of matching category services is achieved. As a result, the proposed algorithm benefits from minimized searching time and matching services. On the other hand, both keyword-based and QoS-based approaches suffer from longer searching time due to extensive search and keyword-based additionally is penalized from non-matching service(s) returned. However, if non-functional service discovery is executed after the functional one, the extensive searching is reduced but with extra processing time due to the two levels discovery. In addition, even though the keyword-based approach does not perform non-functional matching (thus, should reduce search time), it still suffers from lower discoverability due to non-match and partial match services returned, especially when the number of services increases. Both approaches however, have a slightly higher credit than the proposed algorithm in term of wider search space. However, if the N_{MIN} of the proposed algorithm is set to a higher value (i.e., near to the actual number of available services), the search space is increases as well as the searching time. Also, we expect that the results will be similar even with larger simulation parameters due to reliability of the termination condition and searching time reduction of the proposed algorithm.

4.6 Summary

In this chapter, we proposed a service discovery mechanism for the service-oriented content adaptation platform. To the best of our knowledge, most (if not all) of the service discovery protocols did not take search space, matching type, searching time and network proximity factors simultaneously. The proposed service discovery protocol is QoS-aware, distance-aware and quickly terminate when specified search space is achieved. Consequently, searching time is significantly reduced. Relatively, the accumulated time to provide client with adapted content version is minimized as well. In summary, the proposed protocol was able to clearly meet its objective. The proposed protocol increases *discoverability* and outperforms keyword-based and QoS-based approaches. It benefits from minimized searching time due to the advantageous feature that quickly terminates searching; and matching services due to the capability of

matching request and service. In term of distance-aware accuracy, the proposed algorithm sorts the providers according to the proximity measurement and country of origin. This eventually provides sets of closer providers to select from. Also, the distance-aware approach is practical to be deployed for Internet-scale application.

We summarize our contributions into three points: (1) we proposed the service discovery architecture for service-oriented content adaptation platform that includes the service discovery protocol and related algorithms, (2) the protocol has been analysed and shown to accurately and completely locate potential closer service providers, and (3) the proposed discovery algorithm is simulated in various conditions and demonstrated to have high discoverability compared to the pure keyword-based and QoS-based approaches, and (4) the distance-aware algorithm returns closer service providers.

Chapter 5

Path Determination

In service-oriented content adaptation systems, a content adaptation request may requires one or more content adaptation tasks. To serve this request, one service provider is required for each task. A task can potentially be performed by multiple service providers. This leads to different composition possibilities of service providers. In this case, selecting appropriate service providers among the potential providers that have been located in the network is essential. This is referred as the adaptation path determination problem [5]. The proposed path determination algorithm utilizes QoS levels offered by service providers to provide the client with the best possible composition.

5.1 Introduction

A number of services are required to completely serve a given request with more than one task. Each required service can potentially be performed by multiple providers. This leads to different adaptation path possibilities (i.e., providers' composition) as such making path determination challenging.

In this chapter, we propose a path determination algorithm that enables clients to select the best possible service providers among the located candidates. The proposed algorithm consists of a greedy and single objective assignment function that is constructed on top of the possible paths. The score computation logic adopts the 'quality of functionality' view [73] by applying the comparative approach, i.e., compare the value at the i^{th} node with the maximum or minimum node value at the same level.

As a result, each compared QoS value is represented by an appropriate score. Also, we concentrate on selecting a single optimal path as opposed to those discussed in [1, 5]. Optimal path refers to the combination of service providers those possibly best suit the client's QoS criteria to facilitate the adaptation tasks i.e., the content adaptation request [1, 13]. If we have multiple optimal paths, then we need to have another decision rules to choose the best one, with which complicates the determination. Therefore, having a single optimal path is essential to increase the service selection execution performance. We take this into account to develop the solution. The performance of the proposed approach is studied in terms of efficiency of service selection execution under various conditions. The results indicate that the proposed approach performs substantially better than the baseline approach.

Next, we formulate the path determination problem.

5.1.1 Problem Statement

Let $C = \{cObj_1, cObj_2, \dots, cObj_c\}$ and $C' = \{c'Obj_1, c'Obj_2, \dots, c'Obj_d\}$ be a set of original content and adapted content version required by the client, respectively. To achieve C', the content adaptation request is composed of a series of content adaptation tasks $T = \{t_1, t_2, \dots, t_m\}$. For each task, the service discovery protocol returned a set of service providers $S = \{s_1, s_2, \dots, s_n\}$ that matched the required content adaptation and the client's QoS requirements. As each single task can potentially be performs by multiple service providers; each provider with QoS criteria $Q = \{q_1, q_2, \dots, q_z\}$, choosing appropriate service providers is an obvious requirement [1, 5, 13]. Given a set of S, T and Q, the adaptation path determination problem is how to allocate content adaptation tasks together with the content segments to service providers with the aim of achieving the best possible composite service providers. Selecting the best possible service providers maximize the QoS delivered to the clients. We assume that there is no correlation between the QoS criteria.

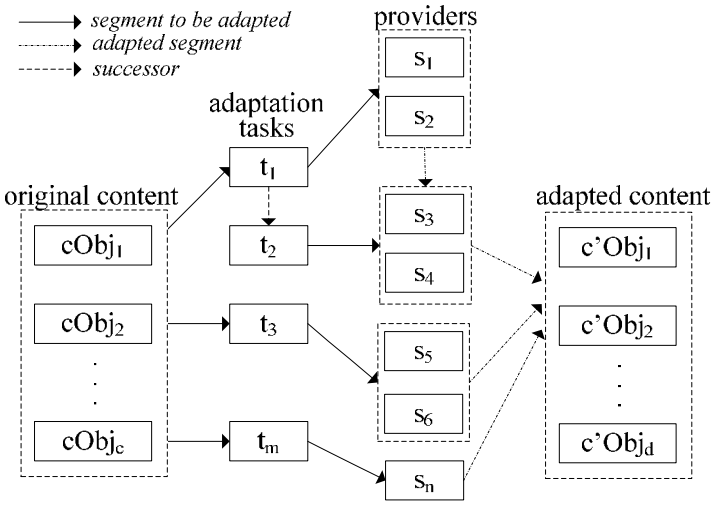

Figure 5.1: Adaptation path determination scenario.

Graphical representation of the path determination problem is shown in figure 5.1. As an example, let us take searching the Internet for a certain topic piecemeal that returns heterogeneous results comprising unstructured text, deep web data sources, web services, images and other multimedia files. In this scenario, a content object may require one or more content adaptation tasks that can be facilitated by several service providers.

5.2 Adaptation Path Determination Policy (APDC)

In this section, we describe the proposed adaptation path determination policy (APDC). Table 5.1 describes the commonly used notations in this section.

Table 5.1: List of notations.

Notation	Description
P	Path
n_l	Service provider's/node's score
q_m	QoS score of a node
w_m	Weight value for QoS
v_i	Value of the QoS
$v_i\ max$	Maximum QoS input value
$v_i\ min$	Minimum QoS input value
T	Set of content adaptation tasks
Q	Set of QoSs for the tasks
S	Set of available service providers for the tasks

Figure 5.2 shows a pseudo-code of the APDC algorithm. The inputs to APDC are the set of content adaptation tasks (i.e, $T = (t_1, t_2,\ldots, t_m)$), set of available service providers for each task (i.e, $S = (s_1, s_2,\ldots, s_n)$) and set of quality of services (QoSs) for a particular task (i.e, $Q = (q_1, q_2,\ldots, q_z)$).

Algorithm 5.1: ADPC

INPUT: S, Q, T
OUTPUT: Service provider for each task
BEGIN
20: **AP** $\leftarrow \langle C, C', T, D, Q \rangle$
21: **Construct adaptation path tree** (AP: Services, QoSs, Tasks)
22: **FOR** each path created **DO**
23: **Calculate Aggregate Score (Pi)**
24: **END FOR**
25: **FOR** each task **DO**
26: Select a service in the optimal path
27: Assign task to service provider
28: **END FOR**
29: **RETURN** service provider for each task
END

Figure 5.2: APDC algorithm.

Path determination is generally composed of at least two inter-related steps: (a) adaptation path construction; and (b) mechanism of choosing the optimal path.

5.2.1 Constructing Adaptation Path

In the construction of the content adaptation path, we used an adaptation planning based on [1] to generalize and represent the content adaptation case. We use a five-tuple to represent adaptation planning (AP) as shown in the following equation:

$$AP\ \langle C, C', T, S, Q \rangle . \tag{5.1}$$

where C is the original content requested, C' is desired content version (i.e., goal state), T is the number of adaptation tasks, S is the set of available service providers for a particular task, and Q represents the total number of QoSs for a particular task.

Using the precedence graph, the sequences of tasks are arranged based on their dependencies, which includes the original content *C* as the start point and goal state *C'* as the end point, as follows:

$$C \rightarrow t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow \cdots \rightarrow t_n \rightarrow C'.$$

Starting with task t_1, each available service for t_1 creates one different node/link to t_2, in left to right order along the paths' score tree. This step is repeated for the next consecutive tasks until t_n is reached (i.e., the desired content version *C'* is achieved). The combinations of these nodes create a number of the potential adaptation paths. For each task, the available service providers are created as different nodes. Also, each task is associated with the service selection QoSs. For instance, a suitable service provider for each task can be selected based on time and reputation QoSs.

As an example, consider the adaptation case $AP\ \langle C, C', 3, 2, Q\rangle$ such that the initial state *C* is a full video with Spanish audio and the goal state *C*' is to have a short animation version of the video with English audio. Further suppose that three adaptation tasks $T = \{t_1, t_2, t_3\}$ and $S = \{s_{11}, s_{12}, s_{21}, s_{22}, s_{31}, s_{32}\}$ service providers. An example of task t_1 is conversion of video to animation, t_2 is translation of Spanish to English audio (of the video) and t_3 is media summarization of the animation. Further assume that t_1 and t_2 are independent of each other but both are the predecessors of t_3. Further suppose that each task is performed by 2 service providers. For example, t_1 is performed by 2 service providers $\{s_{11}, s_{12}\}$, t_2 is performed by 2 service providers $\{s_{21}, s_{22}\}$ and t_3 is performed by 2 service providers $\{s_{31}, s_{32}\}$. Since there are 3 tasks and each task could be serviced by 2 service providers, we have 8 possible adaptation paths to select from.

Figure 5.3 shows the generated paths' score tree for the adaptation case $AP\ \langle C, C', 3, 2, Q\rangle$, where we have 3 tasks with 2 available service providers for each task to complete the content request. s_{ij} is the available service *i* for task *j*, P1 to P8 are the possible adaptation paths, *C* is the original content requested and *C'* (for 1 to 8) are the goal state. Note that t_1 and t_2 are independent, so their sequence is flexible. Thus, the sequences of tasks along the tree can be arranged either $C \rightarrow t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow C'$ or $C \rightarrow t_2 \rightarrow t_1 \rightarrow t_3 \rightarrow C'$.The highest aggregate score is selected as the best possible path i.e., optimal path.

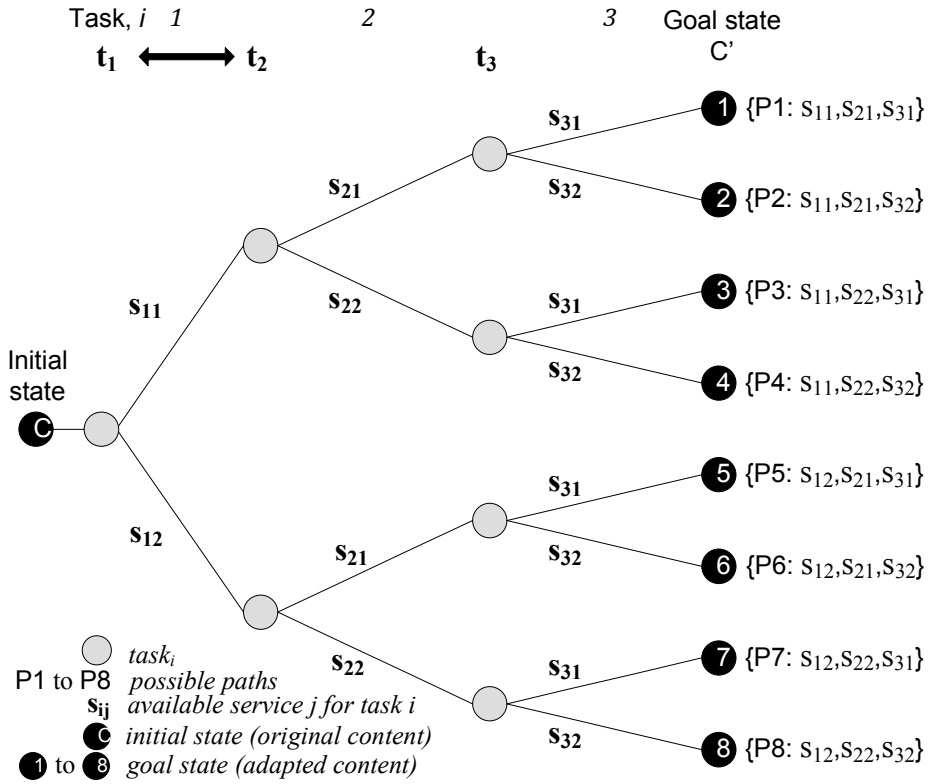

Figure 5.3: Paths' score tree generation.

5.2.2 Calculate Aggregate Score

Each path is associated with an aggregate score. A given path score computation is based on QoS monotonic type and the QoS value (i.e., QoS offered levels). A service provider (i.e., node) can be associated with one or more QoS (e.g., rating, reliability, etc.). Also, clients can rate a given QoS as more important than another QoS. For example, if a client prefers to minimize the cost rather than time, cost will have a higher weight factor (value) as compared to time. To represent such client QoS preference, we associate a weight, $0 < w_m < 1$, with each QoS specified by the client.

Given a QoS weight (i.e., w_m), a node is computed as a normalized score n_l between [0…1] and defined as follows:

$$n_l = \sum_{m=1}^{z} q_m \times w_m \,. \tag{5.2}$$

where q_m is the QoS associated with a given path and defined as either positive or negative monotonic type. Table 5.2 shows some examples of QoS categorization. Given the value for QoS (v_i) and the maximum QoS value ($v_i$max) offered by one of the comparative nodes, the q_m score for the positive monotonic QoS is determined as follows:

$$q_m = \frac{v_i}{v_i \max}. \tag{5.3}$$

In contrast, the q_m score for the negative monotonic QoS is defined as follows:

$$q_m = \begin{cases} \frac{v_i min}{v_i} & if \quad v_i min > 0. \\ 1 - \left(\frac{v_i}{v_i \max}\right) if & v_i min = 0. \end{cases} \tag{5.4}$$

where $v_i$max is the maximum QoS value offered by one of the comparative nodes.

Note that the case $v_i$min = *0* can only occur for the adaptation cost QoS because, it is possible to have free adaptation cost for the time being. However, when content adaptation services become commercial, service providers will definitely charge at least a minimum cost for each service consumed. As such, the probability of using the second equation in (5.4) for the negative monotonic QoS is very low. Also note that, the QoS monotonic type can be dynamically specified in a manner similar to SLA, i.e., the QoS monotonic type for a given QoS can be changed dynamically based on application.

Table 5.2: Examples of QoS categorization.

Service's QoS	
Positive monotonic QoS	Negative monotonic QoS
Rating	Service cost
Reliability	Adaptation time
Reputation	Transport time

Let *AgS (P)* be a function that computes the aggregate score for a given adaptation path *P*. *AgS (P)* is defined as follows:

$$AgS\ (P) = \sum_{l=1}^{k} n_l. \tag{5.5}$$

From equation (5.5), the aggregate score for path P_i is computed by adding the nodes' scores (i.e., n_l) along the path P_i and k is the maximum number of nodes in path P_i. Figure 5.4 shows an example of the path score tree with the aggregate score.

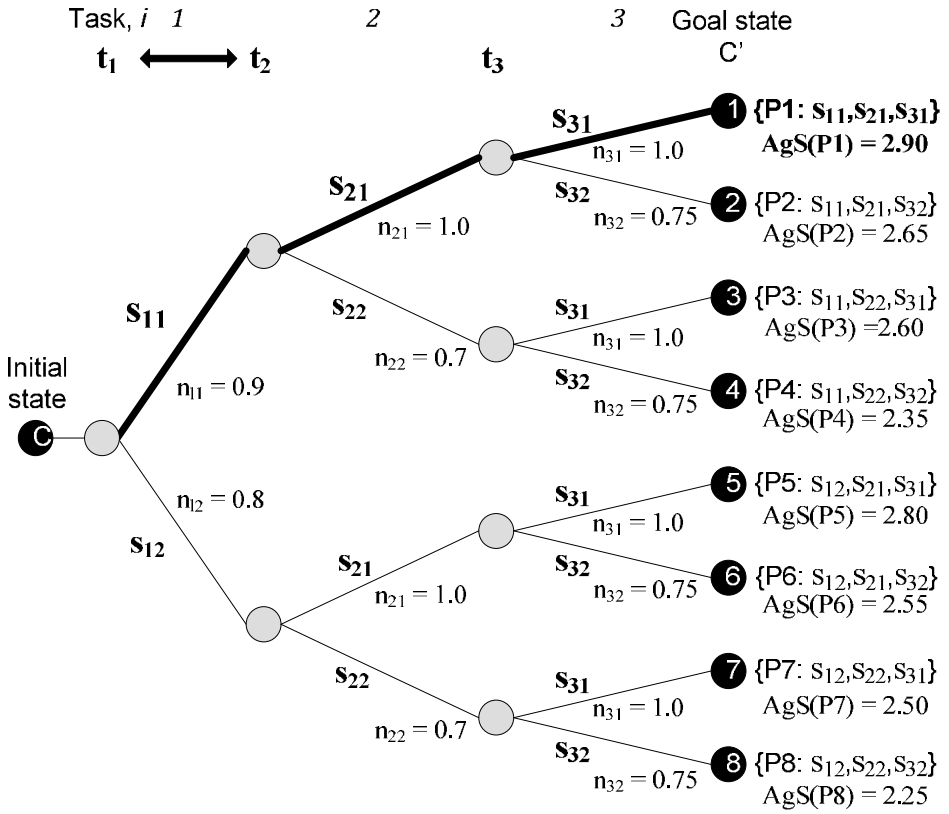

Figure 5.4: Paths' score tree with aggregate score.

For example, consider the adaptation case $AP\ \langle C, C', 3, 2, Q\rangle$ in figure 5.5. Let us further assume that a suitable service for an adaptation task is selected based on 2 QoSs $Q = \{time,\ reputation\}$ such that time (s_{11} = 0.6s, s_{12} = 1.0s, s_{21} = 0.8s, s_{22} = 1.0s, s_{31} = 0.8s, s_{32} = 1.6s), reputation (s_{11} = 4, s_{12} = 5, s_{21} = 5, s_{22} = 3, s_{31} = 4, s_{32} = 4) and both QoSs are to be equally rate (i.e., $w_m = 0.5$ each). From table 5.2, time QoS is a negative monotonic QoS while reputation QoS is a positive monotonic QoS. Figure 5.4 depicted the path score tree with the computed aggregate score.

5.2.3 Service Selection

Each path is associated with an aggregate score. For each content adaptation task, we select the best services to perform the tasks from the generated optimal path and assign the tasks to the selected service providers. The path with the highest aggregate score will be selected as the optimal path [13, 103].

Consider the case shown in figure 5.4 again. In this case, P1 is selected as the optimal path based on the highest aggregate score. Then, the algorithm selects the service provider along path P1 to perform the three tasks (i.e., s_{11}, s_{21}, s_{31} are assigned to perform t_1, t_2, t_3, respectively), as follows:

$$T \rightarrow S\ (\ f(t_1, S): t_1 \rightarrow s_{11}$$
$$f(t_2, S): t_2 \rightarrow s_{21}$$
$$f(t_3, S): t_3 \rightarrow s_{31}\ ).$$

Each chosen provider will be given the content segment and the content adaptation control information. Finally, these service providers adapt the content and send it back to the client. Figure 5.5 depicted the content adaptation control information.

Identifier {taskID, serviceID}	Content segment {cObj}	Adaptation instruction {task details, expected cObj'}	Successor {IP, Port}

Figure 5.5: Content adaptation control information.

The content adaptation control information is made of four fields, identifier, content segment, adaptation instruction and adapted content successor details. The identifier field stores the task ID and the service provider ID. Content segment to be adapted and content adaptation instruction that detailing the required task and the expected output is provided is the next two fields. The successor field enables the provider to send the adapted or partially adapted content to the immediate successor via the destination IP address and port.

5.2.4 Analysis of Algorithm

In this subsection, we will discuss some of the analyses of the proposed service selection mechanism.

Proposition 5.1: The maximum number of available paths *P(A)* to be generated is bounded by equation (5.6), where *n* is a number of service providers available for a particular task and *m* is a number of tasks having *n* services as option.

$$P(A) = 1^{m_1} \times \ldots \times (n-1)^{m_{n-1}} \times n^{m_n} . \quad (5.6)$$

Proof: Using mathematical induction approach [104]:

Basis: Product rule states that if a procedure is done by two tasks (let's say, n_1 and n_2 ways to do task 1 and 2, respectively), there are n_1 x n_2 ways to do the procedure.

Initial step: For any positive integer *m*, let *P(m)* be the product rule for *m* tasks. For the basis case, take $m = 2$ (this refer to product rule for two tasks). Now assume that *P(m)* is true. Consequently, $P(0) = 0$ is true.

Inductive step: Consider $(m+1)$ tasks. $t_1, t_2, \ldots, t_m, t_{m+1}$, which can be done in $n_1, n_2, \ldots, n_m, n_{m+1}$ ways respectively. By the product rule of two tasks, the number of ways to do this is the product (multiplicity) of the number of ways to do *m* tasks, including n_{m+1}. By the inductive hypothesis this is n_1 x n_2 x … x n_m x n_{m+1}, as desired. □

Associate basis: If $n_1 = n_2$, n_1 x $n_2 = n^2$ (in this way, we can group the tasks with the same number of ways together). Similarly, if $n_1 = n_2$, n_1 x n_2 x n_m x $n_{m+1} = n^2$ x n_m x n_{m+1} is true.

Therefore:

$1^{m_1} \times \ldots (n-1)^{m_{n-1}} \times n^{m_n}$ holds.

Example: Suppose we have the following scenario:

$Tasks = \{t_1, t_2, t_3\}$.

$Providers = \{s_{11}, s_{12}, s_{21}, s_{22}, s_{31}, s_{32}\}$.

$Mapping = \{t_1: s_{11}, s_{12}; t_2: s_{21}, s_{22}; t_3: s_{31}, s_{32}, s_{33}\}$

Here, we have 3 tasks and 6 service providers with task to provider mapping of 2. As per proposition 5.1, we have: $1 \times 2^3 = 8$ available paths. Consider another example, $Tasks\_Providers = \{t_1: s_{11}, s_{12}; t_2: s_{21}, s_{22}; t_3: s_{31}, s_{32}, s_{33}\}$. In this case, we have 2 tasks with 2 ways and 1 task with 3 ways. So, we have: $1 \times 2^2 \times 3^1 = 12$ available paths.

Proposition 5.2: $O(T\,S)$ is the time complexity for service selection initialization.

Proof Sketch: For complexity analysis, we focus on service selection initialization time. We followed the analysis methodology descibed in [27]. Let T be the total number of tasks required to process an adaptation request and S is the available service providers during the lookup process. The analysis assumes that there is constant number of service providers for each task. The time complexity for initialization tasks is $O(T)$ and for contacting service providers is $O(S)$. Thus, $O(T\,S)$ is the time complexity for service selection initialization. □

5.3 Performance Evaluation

We use simulation to study the efficiency of the service selection execution of the proposed score-based in APDC policy against the baseline, in term of optimal path determination. Simulation is used to provide a controlled environment and to ensure that the experiments are repeatable. Having a single optimal path improves the service selection execution [12]. On the other hand, having multiple optimal paths will require the system to have additional decision rules to choose the best path, with which complicates the determination. We followed the simulation and verification methodology descibed in [102]. To perform the simulation, data to represent the QoS' values is generated based on skew distribution as the skew distribution is often useful to fit observed data with "normal-like" shape of the empirical distribution but with lack of

symmetry [105]. This is practical to represent the QoS' values between services [106]. The data sets are generated from the skew distribution data generator provided by [105]. These data sets are used as input for the score-based approach in both policies.

Three different simulations were conducted to study the service selection execution towards (1) number of QoS, (2) number of service providers and (3) number of tasks. For each simulation, two different workloads were implemented, workload 1 (W1) and workload 2 (W2). In W1, each QoS has the same weight while W2 imposed QoSs with different weight. These workloads are important to represent the user preference towards the selection QoS [1, 13]. For instance, if a user prefers to minimize the cost rather than time, cost QoS will have a higher weight factor (value) as compared to time, i.e., W2 is applied. Meanwhile, W1 represents that both QoS (cost and time) to be considered equally. Table 5.3 describes the simulation workloads in detail.

Table 5.3: List of workloads.

Workload	Description
W1	Workload 1 represents that all QoS have the same priority and to be considered equally during score computation, i.e., same weight is imposed to all QoS in a node.
W2	Workload 2 represents that QoS have different priority and not to be considered equally during score computation, i.e., different weight is imposed to different QoS in a node. This workload captures the client preference towards the QoS. For instance, if a client prefers to minimize the cost rather than time, cost QoS will have a higher weight factor (value) as compared to time.

At each run, we generated the number of adaptation tasks (T) to be between 1 to 5. The total number of QoS (C) for a particular task is set in the range of 1 to 4. We set the number of available service providers (S) for a particular task in the range of 2 to 5. The values we used for each parameter are in line with the current literature and also reflect the actual environment. The number of tasks and the QoS for each task used in the experiments are in line with the work of [5, 13]. The number of service providers is chosen based on [1].

We used the static path determination criteria (SPDC) policy adopted in [5, 13, 1] as the baseline policy for the purpose of performance comparison. We chose SPDC policy as it is widely accepted and is the closest policy to our policy i.e., objective function. SPDC assigns score to each node, which is accumulated to generate aggregate score for each path. The score computation in SPDC policy is represented as equation (5.7), where Q_{ij} is the node score (before associating weight), Q_{ij}^{max} and Q_{ij}^{min} are the maximum and minimum values of the row respectively.

$$Q_{ij} = \begin{cases} \frac{Q_{ij}^{max} - Q_{ij}}{Q_{ij}^{max} - Q_{ij}^{min}} & if\ Q_{ij}^{max} - Q_{ij}^{min} \neq 0 \\ 1 & if\ Q_{ij}^{max} - Q_{ij}^{min} = 0 \end{cases}. \quad (5.7)$$

We modified the above, based on the same logic in [13], to represent the positive relation's score for the SPDC as follows:

$$Q_{ij} = \begin{cases} \frac{Q_{ij} - Q_{ij}^{min}}{Q_{ij}^{max} - Q_{ij}^{min}} & if\ Q_{ij}^{max} - Q_{ij}^{min} \neq 0 \\ 1 & if\ Q_{ij}^{max} - Q_{ij}^{min} = 0 \end{cases}. \quad (5.8)$$

For comparing the efficiency of the service selection execution of the two policies, first, we study the single optimal path generation which is based on the adaptation path aggregate score, *sop* defined as follows:

$$sop = \frac{single\ optimal\ path\ generated}{number\ of\ runs}. \quad (5.9)$$

Then, we analyse APDC's improvement on service selection execution by comparing the *sop* between the two policies. This analysis is adopted from [107] and considers the improvement, *im* as the ratio between the baseline policy and the proposed policy, defined as follows:

$$im = \frac{APDC\ sop}{SPDC\ sop}. \quad (5.10)$$

Then, we compute the average service selection execution improvement ratio,

defined as follows:

$$ar = \frac{\sum_1^c (im)}{counter,c}. \quad (5.11)$$

5.4 Results and Discussion

Extensive simulation analysis of the proposed policy has been carried out. Table 5.4 describes the commonly used QoS in this simulation and in line with [1, 13].

Table 5.4: List of QoS used.

No	Q_i	Description
1	Time	Time for one adaptation task to be completed
2	Cost	Cost to perform the specific adaptation task
3	Rating	Adaptation service rating
4	Reputation	Service provider's reputation in general

5.4.1 Single Optimal Path Generation

In this subsection, we examine the relative performance of the APDC and SPDC policy with respect to the generated single optimal path. For each simulation, 100 runs were performed.

Figure 5.6 shows the reduction ration (y axis) as a function of the QoS (x-axis). In the simulation, we varied the number of QoS from 1 to 4. The number of tasks and services remain constant (AP <C, C', 1, 2, 1-4>). As can be seen from figure 5.6, APDC generated higher percentages for single path generation for both W1 and W2 compared to the SPDC. The percentage increases steadily along x-axis for both policies. APDC constantly produces around 90% for both workloads along x-axis. There is a significant different between (1) APDC-W1 and SPDC-W1 around 15%, and (2) APDC-W2 and SPDC-W2 around 7%. There is no significant different between APDC W1 and W2 due to the score-based approach implemented in APDC. This figure implies that having difference QoS weighting does not much affect single optimal path generation in APDC. In the other hand, QoS weighting do affect the single optimal path generation in SPDC. This also indicates that APDC policy is more stable towards QoS variation compared to SPDC.

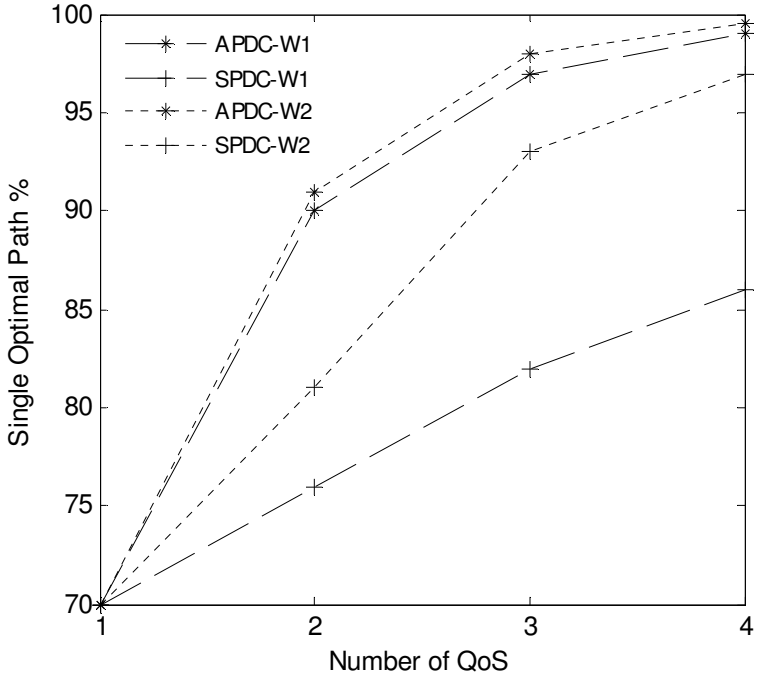

Figure 5.6: Single Optimal Path Generation towards QoS.

Figure 5.7 shows the reduction ration (y axis) as a function of the tasks (x-axis). In the simulation, we varied the number of tasks from 1 to 5. The number of QoS and services remain constant (AP < C, C', 1-5, 2, 2>). As can be seen from figure 5.7, there is a slight increment of single optimal path generation for both policies and workloads along x-axis. APDC constantly produces 80% for W1 and 90% for W2, along x-axis. Meanwhile, SPDC produces around 70% for W1 and 80% for W2. There is a significant different between (1) APDC-W1 and SPDC-W1 around 30%, and (2) APDC-W2 and SPDC-W2 around 20%. A 5% margin is observed between APDC W1 and W2 and 10% margin for SPDC W1 and W2. This implies that number of tasks has a considerable impact on single optimal path generation for SPDC and a small impact for APDC. This also indicates that APDC policy is more stable towards tasks variation compared to SPDC.

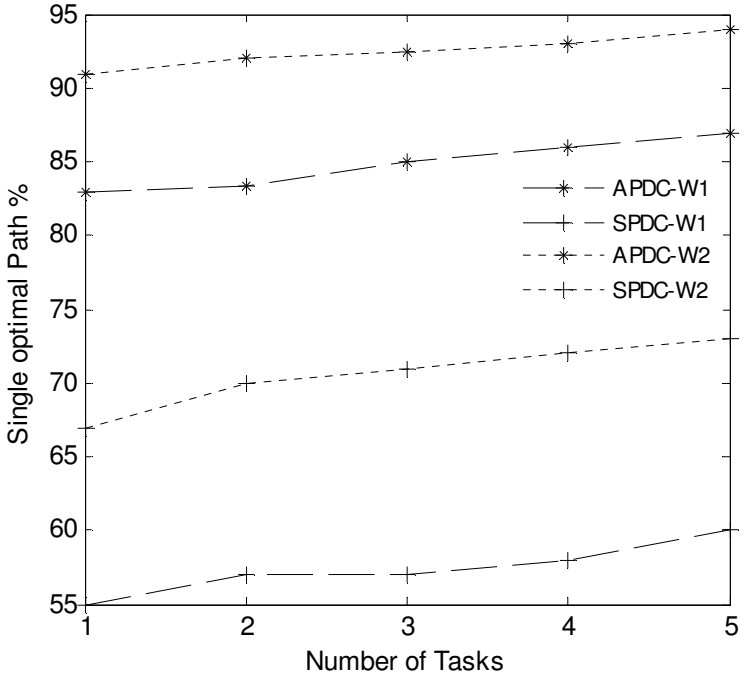

Figure 5.7: Single Optimal Path Generation towards Tasks.

Figure 5.8 shows the reduction ration (y axis) as a function of the services i.e., service providers (x-axis). In the simulation, we varied the number of services from 2 to 5. The number of tasks and services remain constant (AP < C, C', 2, 2-5, 2>). As can be seen from figure 5.8, there is a very slight increment of single optimal path generation for both policies and workloads along x-axis. APDC constantly produces 96% for both workloads along x-axis. Meanwhile, SPDC produces around 65% for W1 and 83% for W2. There is a significant different between (1) APDC-W1 and SPDC-W1 around 25 %, and (2) APDC-W2 and SPDC-W2 around 28%. A 6% margin is observed between APDC W1 and W2 and 10% margin for SPDC W1 and W2. This implies that number of services has a considerable impact on single optimal path generation for SPDC and a small impact for APDC. The score-based approach implemented is SPDC contributes to the low percentage and this aligned with discussion in [12].

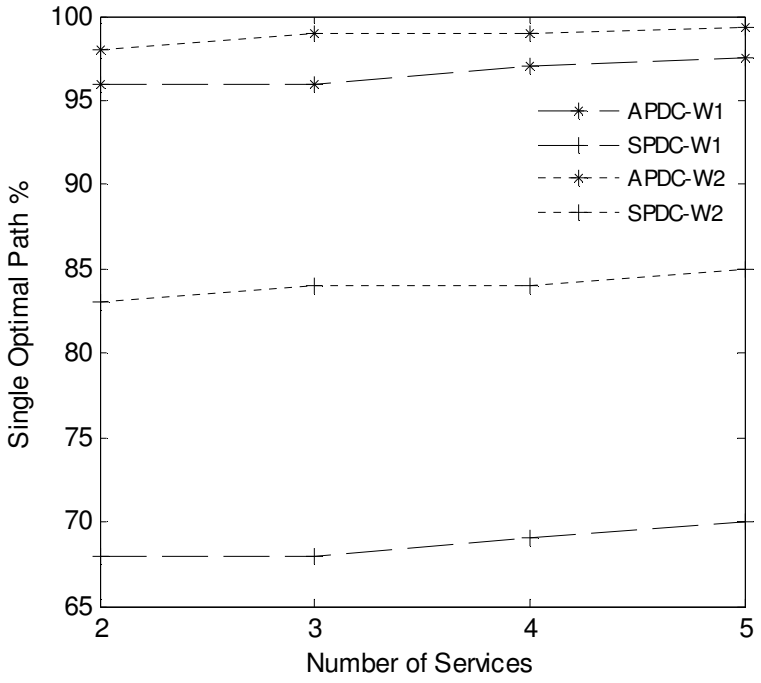

Figure 5.8: Single Optimal Path Generation towards Service Providers.

Taken as a whole in these three simulations, the single optimal path generation percentage increases for both policies and workloads along x-axis. The proposed APDC policy is notably better in every variation (QoS, tasks and services) of the simulations. In addition, we found that, applying different workload (W1 and W2) increases the percentage within a particular policy and this is aligned with [108] findings.

5.4.2 APDC's Execution Analysis

In this subsection, we examine the relative performance of the APDC's service selection execution improvement by examining *op* in APDC compared to *op* in SPDC for each simulation, using equation (5.10). An improvement means to be *t* time faster throughout *n* parameters compared to the baseline policy. Table 5.5 presents APDC average service selection execution improvement ratio generated using equation 5.11.

Table 5.5: APDC average service selection improvement ratio (ar).

Parameter	W1	W2
QoS variation	1.14	1.02
Service variation	1.48	1.31
Task variation	1.40	1.18
Overall average	**1.34**	**1.17**

As can be seen from table 5.5, APDC service selection improvement ratios towards number of QoS are around 1.14 for W1 and 1.04 for W2. This implies that number of QoS has a little impact to improve the APDC's selection execution over SPDC. APDC service selection improvement ratios towards number of services are around 1.48 for W1 and 1.31 for W2. This indicates that, varying number of services (especially when more tasks are required) has a considerable impact for APDC policy in improving selection execution over SPDC. Meanwhile, APDC service selection improvement ratios towards number of tasks are around 1.40 for W1 and 1.18 for W2. This implies that number of tasks has a major impact on APDC selection execution over SPDC. Altogether, the service variation offers the major APDC execution improvement while the QoS variation offers the least for both workloads. This also implies that service and task variations suffer the most when applying SPDC score-based approach.

In average, APDC can improve the service selection execution up to 1.3 times faster compared to the baseline policy. This improvement ratio is considerably satisfactory to improve the service-oriented content adaptation performance.

5.5 Summary

In this chapter, we proposed a path determination criteria (APDC) policy to solve service selection problem for service-oriented content adaptation systems. The proposed policy is able to represent the decision making firmly according to the justified arguments. In summary, the proposed policy was able to clearly meet its objective. The path tree construction algorithm is presented and the score models have been simulated. The simulation results showed that the APDC policy produces appropriate score towards QoS and efficient in term of improving service selection execution. The proposed policy is based on the fact that a QoS has a different relation towards the score; positive monotonic and negative monotonic, which oppose the SPDC foundation.

The proposed policy can be adapted to the social network or collaborative activity with some modifications.

Our proposed algorithm avoids the shortcomings associated with SPDC. Unlike SPDC and [108], to compute service providers scores we differentiate the QoS type into two; positive and negative monotonic types. The former defines that the score has a direct proportional (positive relationship) towards the service's QoS value. The latter defines that the score has an inverse (negative) relationship towards the service's score value. For instance, cost and time QoS have inverse relationship towards score, i.e., higher adaptation time or cost earns a lower score. In contrary, rating and reliability QoS have positive relationship towards score, i.e., higher service rating or reliability earns a higher score. This is aligned with the discussion in [69]. However, study in [69] does not consider the negative monotonic QoS with zero as the QoS value. Cost is an example of a negative monotonic QoS that can have zero value as the valid input. This renders solution in [69] error-prone.

We summarize our contributions into three: (1) we proposed a multi-criteria path determination policy that differentiates criteria in term of its appliance score relation, (2) we proposed a horizontal path score tree together with the proposition to determine the total number of available paths, and (3) APDC is proved to be substantially efficient in term of generating single optimal path and improving service selection execution. Another key finding is that, applying different QoS weighting for services variation improves service selection execution.

Chapter 6

Service Level Agreement

In service-oriented content adaptation scheme, content adaptation functions are provided as services by third-party service providers. Clients pay for the consumed services and thus demand service quality. This makes service quality assurance as an important component. In this chapter, we develop the framework for managing service level agreement (SLA) that is tailored to service-oriented content adaptation scheme. The SLA framework consists of three interrelated phases: creation, monitoring and enforcement. Then, within the creation phase, we propose a detailed strategy for SLA negotiation that exploits the concept of QoS adaptation.

6.1 Introduction

Service-oriented content adaptation scheme allows clients to use content adaptation services that geographically distributed across the network. These services are provided by third-party service providers i.e., these providers may not the owner of the content requested by the clients. Clients, through the client-side brokering, receive adapted content version and pay for the consumed services. For each service being consumed, the service provider promised a specific QoS levels to the client. Thus, service providers are obligated to deliver QoS accordingly. This makes quality assurance as an important issue and this is inline with the claim made by [72] that quality assurance is becoming increasingly important for both clients and service providers. One way to achieve quality assurance is using service level agreement (SLA).

Service level agreement (SLA) is a formal contract negotiated between a service provider and the client for the service. It outlines the relationship between parties to understand each other's needs, preferences and expectations. It should include how to perform future service delivery including QoS levels required, performance-tracking techniques, performance reports, managing problems and conflicts, security and termination [110]. During service delivery, SLA requires real-time verification [62]. If violation occurs, appropriate action (e.g., penalty, conflict resolution) should be taken. Thus, what is required is a framework for managing SLA that is tailored for service-oriented content adaptation.

However, SLA is being neglected in existing service-oriented content adaptation systems [5, 14, 16, 38, 77]. Also, few frameworks [70, 72, 111] for managing SLA exists but it does not fit well with service-oriented content adaptation. This is due to the different requirements of SLA for service-oriented content adaptation scheme. Unlike other Internet services (e.g., VoIP, content delivery network) that focus solely on monitoring network and throughput QoS levels [71, 112], SLA for service-oriented content adaptation should take into account content adaptation related QoS (e.g., translation accuracy, conversion accuracy). We have incorporated these issues while constructing the framework. Three interrelated phases are proposed as the building blocks of the framework. Also, within the creation phase, we propose a detailed strategy for SLA negotiation that exploits the concept of QoS adaptation. This strategy enables both clients and providers to negotiate for specific QoS levels before the SLA are being settled.

6.2 Requirement for SLA

In recent years, the challenge of service-oriented content adaptation is shifting from a focus on enabling content adaptation performed by a set of services, to a focus on assuring accurate adapted content version delivered within promised QoS levels. As more clients start to use content adaptation as a service discovered in a real time manner, and as the number of competing services with similar adaptation function increases, QoS is becoming a differentiating factor.

6.2.1 Quality Views

QoS is the primary factor that relates the agreement within an SLA between the client and the service provider. There are several views of quality that can be implemented in managing SLA for service-oriented content adaptation platform [66, 73]. *Quality as functionality* is measured by considering the amount of functionality that a service can offer to its clients. One service is considered better than others in one of these two cases: it provides a function (or additional function) that is not provided by other, or/and secondly it provides a better value for the same function across providers.

Quality as conformance is a view of comparing the actual QoS delivery with the promise. A good service means that it delivers no less than the stated promise. *Quality as reputation* depends on clients expectation and experience from the service. It is built collectively over the time of the service existence from clients' feedback [69]. A service with good reputation means that it consistently provided specific functionality with specific performance over the time. Most of existing Internet services (e.g., [72, 73]) based their QoS monitoring using *quality as conformance* view. Next, we describe the motivational example.

6.2.2 Motivational Example

To motivate the necessity of SLA for service-oriented content adaptation, consider this:

> *Zack went to visit his relative at Royal Women's hospital in Melbourne. Zack learns that his relative has been diagnosed with a heart complication. To get more information on the heart complication that his relative diagnosed with, Zack decided to browse, using his web-enabled mobile phone, the e-health server at the hospital. Confronted with medical jargons received from the e-health server and to make sense of it all, Zack decided to browse an e-learning server. Zack prefers a summary of the heart condition explanation in French text. However, the content on the e-learning server consists of several inter-related pages of heart and blood vessels information and each contains a long English text with some related graphics.*

Suppose Zack is a subscriber of ABC system that provides content adaptation as a service. As a client, Zack logs into the system to perform his request. During previous registration, he ticked time and cost QoS types in the client QoS requirement checkbox. The system analyses Zack's content adaptation requirements using a built-in adaptation decision-taking engine and realizes that two content adaptation tasks are required (i.e., (a) t_1: English to French text translation, and (b) t_2: French text summarization) to achieve the desired content form (i.e., a summarized French text for each Web page). The system looks up at a participant business registry r to find suitable services that is capable of performing these two tasks. Further suppose that the potential providers have been discovered.

Let us say t_1 and t_2 can be performed by services s_1 and s_2 respectively. Providers of services t_1 and t_2 advertise 'one for all' QoS level offers for both cost and time QoS. Assume both services are free (i.e., cost = 0) via trial version. Adaptation time offered by s_1 and s_2 are 13000 and 9500 milliseconds respectively. Then, providers of services s_1 and s_2 are contacted to perform related tasks. When the adapted content version is ready, it is sent back to Zack's mobile device.

To demonstrate the need for SLA, a basic experiment is conducted to measure the actual QoS delivery for this content adaptation scenario. It is used to demonstrate if non-compliance (e.g., violation) can occur. In this experiment, the monitoring apparatus only monitors the adaptation time QoS. The cost QoS is not monitored as it is freely provided to the client. Assume the required content object level for both tasks t_1 and t_2 have been delivered accordingly. The experiment is run for two requesting devices with different capability. The first client device is a mobile (i.e., iPhone 3GS) using 3G wireless HSPDA 7.2Mbps network, while the second device is a desktop PC running on 100Mbps network. Desktop device is used as a comparison in order to study the impact of QoS levels (i.e., execution time) on different device capabilities. In this experiment, adaptation time is measured when the task is initiated until the adapted content is displayed at client displays [9]. s_1 and s_2 are provided by Babelfish Yahoo! [113] and ExtractorLive [114], respectively. The content object files size varies from 15,000 to 30,000 bytes and mainly made up of text and some images [115].

Figure 6.1 captured the adaptation time QoS. The result has revealed that the adaptation time for the same task and file size is varied between two different requesting

devices. For instance, a client requesting content using a mobile device experience longer time to receive the adapted content version. Please note however, the accuracy of measured value is also influenced by network bandwidth and servers' load.

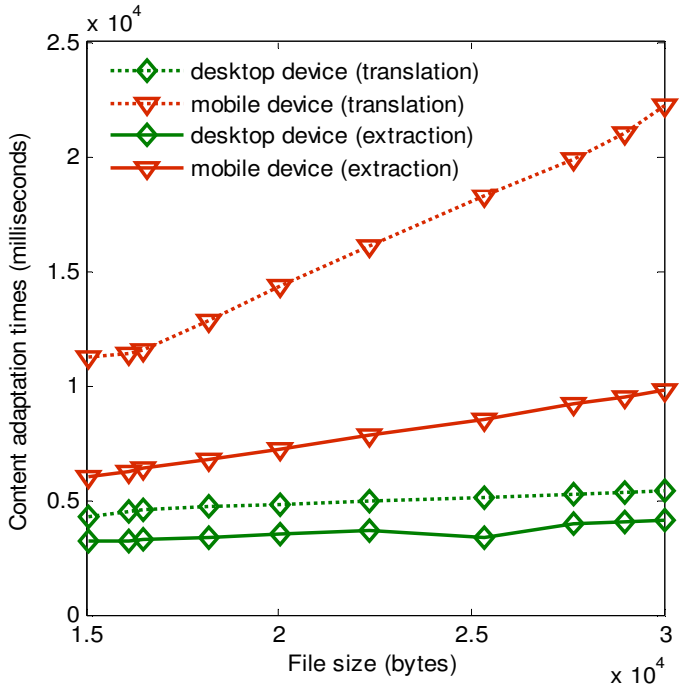

Figure 6.1: Adaptation execution against varied sizes for desktop and mobile devices.

6.2.3 Problem Statement

The implication of the experiment is threefold. First, on the client standpoint, the QoS offers being delivered can be violated. This necessitates a mechanism to monitor and assure actual QoS levels are tailored with the promised QoS levels. Second, on the service provider standpoint, 'all for one' QoS offer (i.e., fixed SLA) has impact on business (e.g., marketing). Take waiting time QoS for example - the provider may advertise waiting time QoS based on a certain server load. For instance, if the offered QoS is for the worst case scenario (e.g., heavy load), they may lose business to others that offer better QoS levels. On the other hand, if the offered QoS is for the best case scenario (e.g., fair load), it may easily lead to violation and get penalized when experiencing flash crowds. This suggests that providers should negotiate QoS offers with clients based on individual adaptation requirements. Alternatively, providers can advertise separate QoS levels for different set of client device or preference groups. This is also aligned with the claim made by [72] that QoS expectations are driving clients to

negotiate specific QoS levels with their providers. Third, on the administration standpoint, non-compliance of QoS level is not necessarily being a violation. It can be resulted from other parties, such as network or natural disaster. In this light, certain non-compliance should be considered as conflict that requires resolution rather than penalty. This necessitates the requirement for SLA management. In this context, the central problem is how to provide quality management that beneficial for both clients and service providers.

6.3 SLA Framework

SLA is a powerful mechanism for expressing all commitments, expectations and restrictions in a business transaction [116]. It formally identifies what guarantees are being offered to the client. The main objectives of SLA in service-oriented content adaptation are (a) to facilitate two-way communication between negotiating parties that includes understanding of need, priorities, and specifications, (b) to protect against expectation creep that includes the identification and negotiation of service levels, (c) to have mutually agreed standard, and (d) to gauge service effectiveness that includes the basis for performing assessment. In this section, we will give a brief description of the management framework of SLA for the service-oriented content adaptation platform. This includes the outline of essential components required for managing SLA.

6.3.1 Architectural Framework

In this framework, brokers and providers have the mechanism to establish SLA. A broker, on behalf of the client, negotiates SLA with service providers. These newly created SLA clearly express the required QoS to be maintained till the end of services execution, the required content object level to be delivered, the penalties in case of failure to provide the offered QoS and the resolution actions in case of a conflict. After the successful creation of SLAs, providers are tasked to perform adaptation. These services execution are monitored to ensure offered QoS levels and required content adaptation are obeyed. In case of any violation against these offered levels, or any conflict; either decided or reported by the client, there should be an enforcement mechanism to penalize the provider or to resolve the conflict, respectively. Figure 6.2 depicted the SLA framework for service-oriented content adaptation scheme.

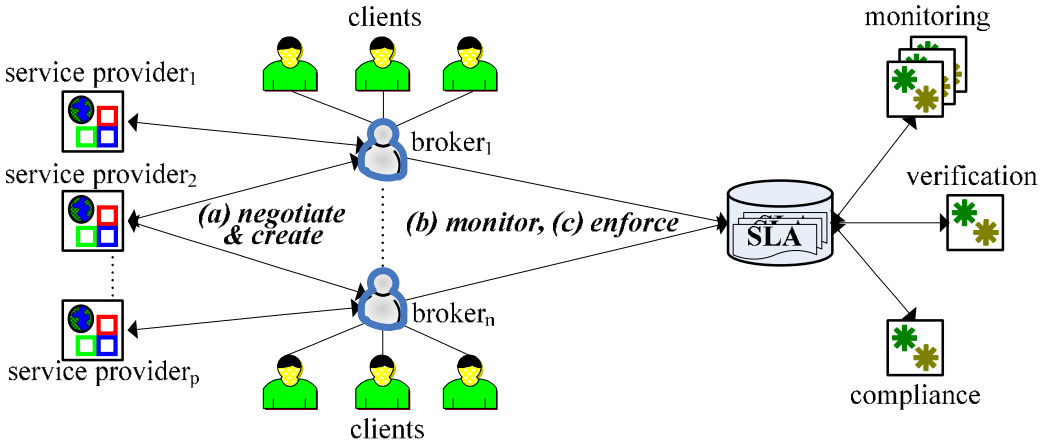

Figure 6.2: SLA framework for service-oriented content adaptation.

The framework serves a workflow to manage SLA. It is composed of three inter-related phases: (a) creation (that includes negotiation), (b) monitoring, and (c) enforcement.

6.3.1.1 Creation Phase

Creation phase is responsible for creating negotiated SLA of each service required to perform a given task. Negotiation occurs in either one of these two scenarios: (a) between the provider and the broker, or (b) between one provider with another (in case of service outsourcing or peering). In this chapter, we focus on the former case. The motivation for QoS negotiation is that a particular provider may not be able to meet the advertised QoS levels; either demanded by or promised to clients. Specifically, this can be due to several reasons such as: (a) the provider realized that its current load is heavy; and (b) the provider realizes that the current requests being served will free up some resources in a short time, thus QoS adaptation (i.e., new QoS) can be offered. For example, if a provider realizes that it cannot deliver within the advertised QoS (e.g., waiting time) due to heavy load, the potential requests being rejected can be offered with a new waiting time. In this sense, the response time (i.e., adaptation time + waiting time) for the client to get the adapted content version is being revised. Table 6.1 depicted some example of QoS.

Table 6.1: Example of common QoS for a service.

QoS	QoS category	QoS description	Example of QoS metric	Example of SLA
Adaptation time	Objective	Time required to perform the adaptation function	millisecond, second, minute	Service must be delivered ≤ (100% of negotiated adaptation time)
Waiting time	Objective	Waiting time before the request is being served	millisecond, second, minute, time unit	Waiting time ≤ (promised waiting time)
Cost	Objective	Cost charged to perform the adaptation function	cent, dollar	Cost charged ≤ negotiated cost
Rating	Subjective	Denote the rating of service delivery	Likert scale	not available

Table 6.2: Example of adaptation functions including related objective QoS and SLA.

Adaptation functions	Function description	Objective QoS	Example of SLA
Filter	Remove information, content	Accuracy	100% of required information is removed
Annotate	Add information to content	Accuracy Completeness	>90% of subtitles annotation is accurate >95% of the movie file is annotated with subtitles
Transcode	Change to different format (within the same content type)	Accuracy	Transcoded content format must be readable and contain > 95% of the original object
Translate	Change to different content language	Accuracy Completeness	95% of English translation is accurate 95% of English text is translated
Convert	Change to different content type	Accuracy Completeness	The converted version is semantically tailored with the original version The converted version must 100% be in the new content type required
Extract	Extract keyword/ summarize from content	Accuracy	>80% of the summarized text is accurate, readable (i.e., in the same language of the original version) and error free.

An SLA stores essential information that includes service definition and specification, client's adaptation requirement, performance metrics (i.e., QoS), QoS measurement and reporting, non-compliance management (i.e., penalties and conflict resolution), and bypassing conditions. Table 6.2 shows some examples of common QoS and specific performance QoS. Standard QoS description and ontology is stored in the QoS ontology database. QoS ontology is important to define standard description and specification between stakeholders. It can be developed in a manner similar to [98].

6.3.1.1.1 Negotiation

Figure 6.3 illustrates the negotiation process including the interaction sequence. Service providers advertise services in a service registry. A broker, on behalf of the client discovers available services from the registry. When suitable services are selected, the broker initiates negotiation with providers for new QoS levels to customize with the specific adaptation requirement (e.g., how much cost and time it takes to perform a task of *s* size) and to consider current load, until a final agreement is achieved. For instance, a service provider advertises one base cost c_b and time t_b for serving a request of *s* size. The broker can determine the request size and estimate the actual cost and time based on the given advertisement. At the provider's end, it performs the same cost and time calculation. If both parties agree with the new calculated adaptation time and cost QoS, SLA can be settled.

Also, the provider might anticipate the current load to evaluate whether the offered waiting time QoS is within delivery or not. If the provider realized that its current load is heavy, it rejects the incoming requests. Alternatively, it calculates the expected waiting time and negotiates for QoS adaptation with the broker of the request. If both parties agree with the new calculated adaptation time, cost and waiting time QoS, SLA is settled and the request is queued into the buffer, waiting to be served. During negotiation, both parties refer to the QoS ontology so that the same QoS specifications are used.

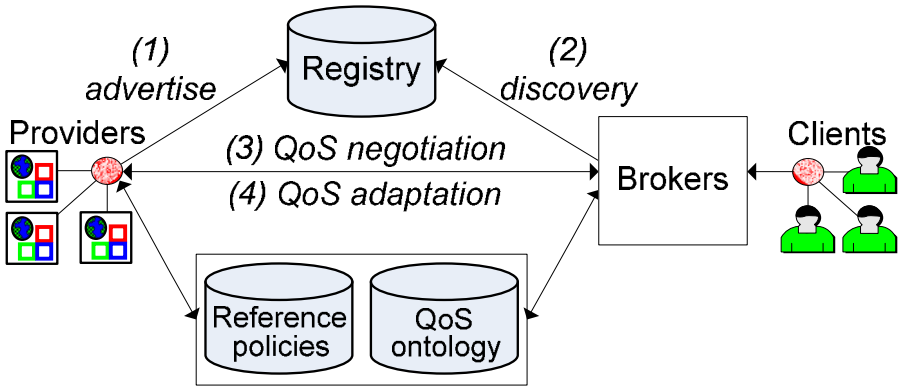

Figure 6.3: Negotiation in creation phase.

Negotiation approach can be classified into three: one-to-one, one-to-many and many-to-many. Administrator decides which approach to be implemented, as long as it can maximize clients' utility. A study by [117] provides an insight on existing negotiation approaches and strategies. In our context, one-to-one approach is used for its simplicity of bargaining [118]. For each task, the broker negotiates with the potential provider for the actual QoS offer. This is performed for all tasks, simultaneously or in sequence. The final deal must tailor at both ends when deadline is met. In future, we plan to explore one-to-many approach. The negotiation component enables providers to engage in QoS negotiation with the broker, while at the same time, providers in negotiation maintain a reference table of QoS commitments made to others. In practice, QoS negotiation is done at session level which is in sequence of service requests. Also, different QoS metrics or unit between negotiating parties can be mapped in real time, in a manner similar to [70]. When the negotiation process is successful, the negotiated SLA is stored in the SLAs repository and is achieved during verification.

6.3.1.1.2 SLA Schema

A generic SLA schema is used to document the negotiated SLA for each service. The WS-agreement acts as wrapper around this schema [116]. The schema consists of one root element called *SLA* tag. It consists of six core child elements: *SLAID, ServiceID, Expiration, Service Level Objective (SLO), Bypassing-conditions* and *Non-compliance* tags. The *SLAID* tag specifies the SLA id while the *ServiceID* tag specifies the service id for which this SLA is created. The expiration time of this SLA is specified in the

Expiration tag. *SLO* tag contains *QoS Type* with corresponding *Offered QoS*, and *Content object level Required* child elements. For each QoS $\in \{1 \ldots n\}$, descriptions containing the negotiated QoS type (e.g., cost, rating and adaptation time) is specified in *QoS Type*, whereas the negotiated QoS level is specified in *Offered QoS* tag. *Content object level Required* specifies the required adaptation level of the content object. The *SLO* tag will be later used during verification process. Suppose any non-compliance of SLO caused by the provider or reported by the client, the *non-compliance* tag is referred. It stored the consequences of violation or conflict. If a violation is decided, the penalty specified in *Violation* tag is referred, whereas if a conflict is decided, the resolution action specified in the *Conflict* tag is referred. *Bypassing-conditions* tag specifies (a) the bypassing conditions (e.g., content server failure, natural disaster, theft, etc.), and (b) the necessary actions when a bypassing condition is declared or detected. Table 6.3 depicts the example of an SLA tags property.

Table 6.3: Example of an SLA tags property.

SLA ID	Service ID	Expiration	Service level objectives	Non-compliance
001.s_m	s_m	(dd/mm/yyyy)	QoS levels:- Q_1^p, cost: [type: cost]; [offer value:50]; [unit: cents]. Q_2^p, transcoding accuracy: [type: accuracy]; [offer value: > 90]; [unit: %]. Q_3^p, waiting time: [type: time]; [offer value: 200]; [unit: millisecond]. Content required:- Q_4^p, output required: [file type: movie], [unit format: AVI]; [unit fps: 25fps]; [unit res.: 800*480].	Violation [penalty]; Q_1^p: [receipt adjustment]. Q_2^p and Q_4^p: [p_1: 50% cost reduction], [p_2: re-adaptation] Q_4^p: [30% cost reduction]. Conflict resolution [action]; Q_1^p, Q_2^p, Q_3^p and Q_3^p: [notice of conflict] Bypassing conditions: [ISP failure, disaster] [Action: terminate SLA]

To formally describe the SLO within an SLA, assertion is used. This formalization provides a feasible solution for SLO monitoring, measurement and compliance management. Each assertion a_n is an atomic statement, and reflects the obligation of a service through the relationship that constraints the agreed variables according to the SLO. Table 6.4 describes the commonly used notions.

Table 6.4: List of commonly used notation.

Notation	Description
A_s	A set of assertions for service S_{ij}
S_{ij}	Service j for task i
Q_k^p	A set of offered Q_s levels and required adaptation level for the service S_{ij}
Q_k^d	A set of actual Q_s and Q_c being delivered for the service S_{ij}
SC	SLA compliancy of A for a particular S_{ij}

From operational standpoint, statement in an assertion made up of logical predicates to relate between values [111]. A logical predicate is composed using variable and logical operators (e.g., constraints $\{\leq, \geq, =, \neq, >, <\}$) that are imposed on those variables. Variables must reflect the operation or measurement of a service. In our context, assertion is used for the verification purpose between the offered QoS with the actual QoS and the required content object's level with the delivered version. For instance, suppose a service is required to provide a specific content object's level of images with a certain QoS levels. When the system notifies the provider to perform the particular task, the provider must agree to provide the service with the required content object's level and within offered QoS. In this case, the agreed terms are the QoS and content object's level and relationships are the obligations of QoS and content object's level, respectively. In our context of SLA monitoring, we adopted the *quality as conformance* view to evaluate the QoS and content object's level. That is, offered QoS or required content object's level for delivering service s is denoted as Q_k^p while Q_k^d is the actual QoS or content object's level delivered by the provider. Then the assertion for the service s_{ij} is given by;

$$SLA_{s_{ij}}: a_k = f\left(Q_k^p, Q_k^d\right). \tag{6.1}$$

where f is the function that measures the conformance between Q_k^p and Q_k^d of service s_{ij} and $k \in \{1, \dots, b-1, b, b+1, \dots c-1, c\}$ is the total number of QoS (denoted by b) and content object's level (denoted by c). Each SLA is made up of one service id and consists of a set of assertions. Each assertion within a service is unique, i.e., no assertion created in an SLA is identical.

The total number of assertions to be created, for a given content request is bounded by equation (6.2), where q is the total of QoS and content object level with 1 and k are the lower and upper bounds respectively, and s_q is the number of services having a particular q.

$$A(p) = \sum_{q=1}^{k} q \times s_q \,. \tag{6.2}$$

Assume we have 3 tasks served by 3 services, with service to q mapping of 2. As per equation (2), we have: $(1 \times 0) + (2 \times 3) = 6$ assertions are created.

For time complexity analysis, we focus on SLAs initialization time. We followed the analysis methodology descibed in [27]. Let S be the total number of services and A is the total number of assertion for each service. The analysis assumes that there is constant number of A for each service. The time complexity for initialization services is O (S) and for initialization assertions is O (A). Thus, O (S A) is the time complexity for the SLAs initialization.

The next step is to verify each assertion using the monitoring apparatus during the service execution.

6.3.1.2 Monitoring Phase

Every service must be capable of being measured and the result being analysed and reported. Monitoring is a formal phase to verify whether the actual QoS and content object's level are delivered within the negotiated SLA during service execution. The monitoring apparatus can be placed at the broker (or at a third party location) and the monitored QoS and content object's level are updated to the corresponding service's assertions for verification or auditing process. Unlike others Internet services (e.g., VoIP service), which focus on network based performance QoS parameters such as throughput, packet loss ratio and packet delay, SLA parameters for service-oriented

content adaptation focus on adaptation-related parameters. Examples of these parameters are presented in table 6.1. Figure 6.4 depicts the example of monitoring apparatus that measure QoS and content object level during service delivery.

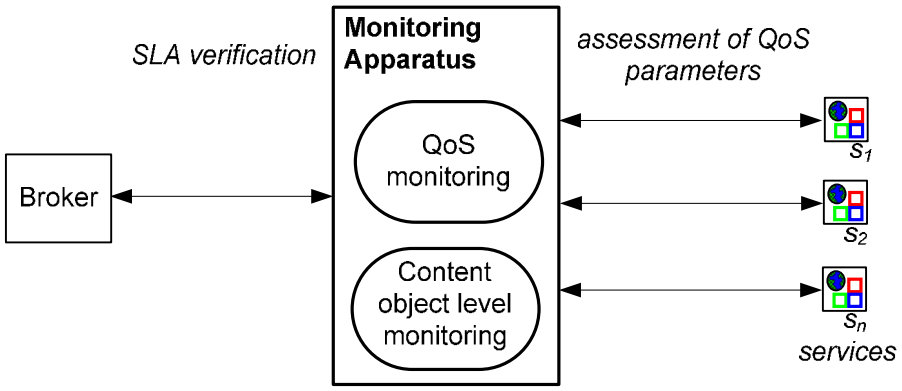

Figure 6.4: Monitoring phase.

6.3.1.2.1 Verification

The monitored QoS levels are compared with assertions created. An assertion can be either TRUE or FALSE. It depends on the obligation of the QoS or content object's level during verification. For positive monotonic QoS (e.g., accuracy), the assertion's obligation to be met is that the actual QoS must be equal or greater than (i.e., ≥) the promised QoS, whereas for negative monotonic QoS, the actual QoS must be equal or less than (i.e., ≤) the promised QoS. For positive monotonic QoS, if $a_i = \{ Q_k^p \leq Q_k^d \}$ is met, then the assertion is TRUE, vice versa. On the other hand, for negative monotonic QoS (e.g., time and cost), if $a_i = \{ Q_k^p \geq Q_k^d \}$ is met, then the assertion is TRUE, vice versa. For content object's level, the assertion's obligation to be met is that the actual content object's level must be equal with (i.e., =) the required content object's level, and if $a_i = \{ Q_k^d = Q_k^p \}$ is met, then the assertion is TRUE, vice versa. TRUE assertion is equal to 1 while FALSE assertion is 0. An SLA compliancy *SC* for a given service s_{ij} is denotes as following:

$$SC_{s_{ij}} = \prod_{k=1}^{K} a_k \tag{6.3}$$

where $\forall k \in \{1, \dots, K\}$, 1 and *K* are the lower and upper bounds for *a* as in equation 6.1.

If all the assertions for the service s_{ij} are TRUE, the SLA is in compliance, while the SLA is in non-compliance form if any of the assertion is FALSE, i.e., $SC = 0$. When a non-compliance case detected, the broker invokes the enforcement phase. This phase determines the non-compliance type and the corresponding action. A non-compliance case may also be the result of reported cases received from clients. For example, a client can report a case if the adapted content version received at his end is not as expected.

6.3.1.3 Enforcement Phase

The enforcement phase is used to (a) decide whether a non-compliance case is a direct violation, a conflict or a result from bypassing conditions, (b) to enforce the necessary action and (c) to provide a real-time compliance reporting to clients. It takes non-compliance attributes as the input, refers to the rule repository and the particular SLA tag, and uses the decision engine to determine the output (i.e., the non-compliance case). When violation is decided, it determines the corresponding penalty to its producer. On the other hand, when a conflict is decided, the corresponding resolution action is taken. Figure 6.5 illustrates the enforcement phase.

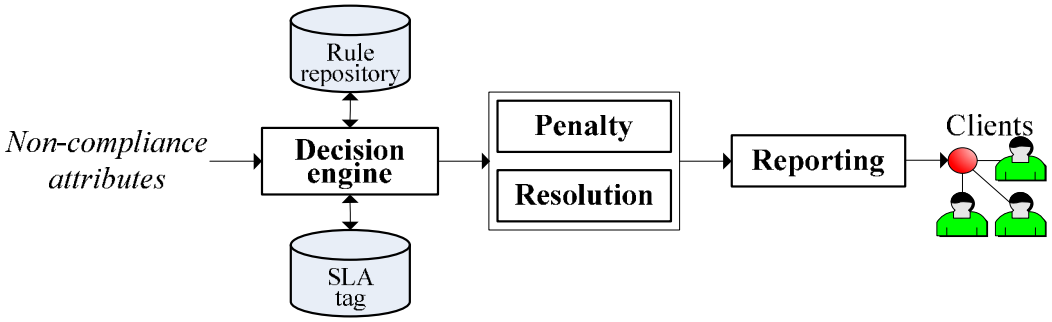

Figure 6.5: Enforcement phase.

6.3.1.3.1 Decision Engine

Decision engine utilizes the decision logic to produce non-compliance decision. The input to the engine is non-compliance attributes, i.e., "v", "w" and "x". "v" represents the form of the non-compliance; either detected during verification or complained by the client when the adapted content is received at the client end. This provides the client a way to submit complain. "w" denotes the aspect of non-compliance (i.e., QoS level, required content object or both), while "x" denotes the source (i.e., SLO, bypassing condition, external). Bypassing conditions are retrieved from the bypassing policy

stored in the policies repository. These attributes are gathered or determined by the broker in real-time. Then, the engine uses the decision logic to decide whether a violation or conflict has occurred, before assigning the corresponding penalty or resolution action. For example, a conflict can occur due to bypassing condition.

The decision logic utilizes a rule-based technique as the reasoning engine to identify non-compliance type and the corresponding act to be taken. Expert knowledge is used to model rules in a manner similar to [8, 60]. The rule engine helps the system to achieve better automation [10, 62].The rule takes a form as shown below:

$$\text{Rule } n\text{: IF (case == v \&\& w \&\& x) THEN y;} \tag{6.4}$$

where "*n*" is the particular predefined rule to be identified, and "y" is the non-compliance type (i.e., violation, conflict, bypass). The number of "v", "w", and "x" must be at least one. Rules can be added, deleted or modified, and is managed in offline mode. It is stored in the rules repository. Table 6.5 presents some examples of the non-compliance attributes.

Table 6.5: Example of non-compliance attributes.

form v	aspect w	source x	resulting y
detected	QoS level	SLO of QoS *q*	violation
complain	content object level	external	conflict
		bypass	

For example, suppose we have a non-compliance detected during verification, e.g., the QoS level of text translation accuracy is below the negotiated level. The broker identified the case as *detected*, the aspect of non-compliance is QoS level and the source is SLO of accuracy QoS. These attributes are submitted to the decision logic and represented using equation (6.4), as the following:

v = detected, w = QoS level, x = SLO of accuracy QoS

Using this information, the broker searches for the corresponding rule *n* stored in the rule repository where "v", "w" and "x" hold. The resulted rule *n* classifies the non-compliance type. When the non-compliance type has been identified, it is submitted to

either one of these services: the violation enforcement service or the conflict resolution service. The violation enforcement service get invokes when it gets the violation notification. Based on the SLA id, it refers to the penalty specified in the violation tag and invokes the appropriate penalty mechanism. In practice, a violation is penalizes in term of monetary. Other consequences of SLA violation are disbanding the services from service chain and re-selecting services. On the other hand, the conflict resolution service get invokes when it gets the conflict notification. Based on the SLA id, it refers to the resolution action specified in the conflict tag and invokes the resolution mechanism.

6.3.1.3.2 Reporting

Finally, the broker sends a real time report to the client. The report includes compliance status. As the enforcement phase and content adaptation execution are run concurrently, the compliance processing does not affect the overall time for the client to receive the adapted content form. In the case where a particular service failed, the broker notifies the discovery and selection mechanisms to re-execute, and resend the content segment together with the adaptation control information to the re-selected provider. The failed service is handled according to the SLA. In this chapter, the exploitation of the recovery mechanism is left for future work.

6.4 Negotiation Strategy

In this section, we will give a detail description of the negotiation strategy for one-to-many negotiation i.e., one provider with many brokers for SLA settlement. In this context, we focus on the waiting time QoS. Figure 6.6 illustrates the negotiation context. It consists of three substances: brokers, providers and providers' ranking for each broker. The service provider offered one waiting time QoS during advertisement. This offer may be estimated based on fair load. However, if the server is experiencing heavy load, it may not been able to all incoming requests within the offered waiting time. Hence, at any period of time instant, only some requests can be served within the offered waiting time QoS, thus providing the basis for determining priority. Also, the provider may negotiate a new waiting time for the requests with lower priority.

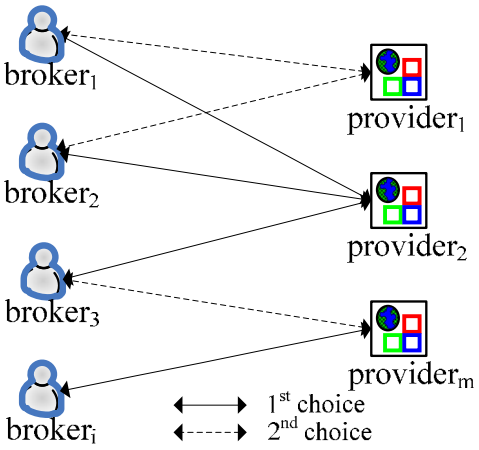

Figure 6.6: Negotiation context.

In the system, there are many brokers $\{b_1, b_2, b_3, \dots, b_i\}$ representing many different clients $\{c_1, c_2, c_3, \dots, c_i\}$, each requiring a series of adaptation tasks $\{t_1, t_2, t_3, \dots, t_m\}$. For each task $t \in T$, the broker (1) looks up at an accessible registry r for available services; (2) ranks suitable services that matched the adaptation function required and other important factors e.g., service's proxy utilization, QoS criteria and network proximity; (3) selects the top service; and (4) performs one-to-one negotiation with the service's provider. Incoming requests at the provider p are sorted by required servicing time i.e., when the servicing is required. The broker keeps the list containing services' ranking. For example, in figure 6.6, provider 2 and 1 are the first and second choices of the broker 1 to perform task t, respectively.

Assume that a provider receives a number of incoming requests at a certain period of time instance. These requests require to be serviced within the offered waiting time. Before settling SLA for all incoming requests, the provider calculates its capability in term of serving all requests within the offered waiting time QoS. It takes into account the current server load ρ, incoming requests λ and the processing requirements of each request.

For example, assume we have three incoming requests at a provider p. These requests are expecting to be served within the advertised waiting time $E(W)$. Provider p checks its current load and calculates the number of requests it can serve within $E(W)$. Further assume the provider realizes that only two requests can be served within $E(W)$.

Thus, it needs a basis to prioritize requests and to reject requests potentially being violated if SLA is settled with the advertised *E(W)*. For instance, in figure 6.6, provider 2 gets four requests from brokers b_1, b_2 and b_3. Provider 2 can accept all service requests if it can serve all requests within the advertised *E(W)*. However, if the current load is high, provider 2 needs to decide which broker(s) request should be served within the advertised *E(W)* (and settle the SLA) and which one should be rejected (or to be negotiated with the new *E(W)*). This helps the provider to avoid potential violation of SLAs. As such, negotiation strategy (including priority model) is rendered as an important component.

Some existing well known priority models such as FIFO [120] and membership can be used to prioritize requests. In FIFO (first in first out) model, the priority is based on first come first serve. Requests are prioritized based on arrival in the system. Let says the arrival of requests from brokers are in this sequence: broker b_1 $A_t^1 = 1$; b_2 $A_t^2 = 2$; and broker b_3 $A_t^3 = 3$. Assume that provider 2 can only serve two requests within the offered waiting time. In this context, provider 2 will only settle the negotiation with the first two requests, i.e., broker b_1 and b_2. Broker b_3 requests will be rejected and thus it has to consider the second choice.

The other model is using membership. In this model, assumes we have three levels of membership i.e., gold, silver and bronze. These levels can be assigned to clients (e.g., brokers) based on certain factors (e.g., usage subscription history, frequency, etc.). If broker b_3 is the gold member of provider 2, it would reject any request from the broker with a lower membership level, e.g., b_1 or b_2.

The third model is using the hybrid model. This model is a combination of the membership and FIFO models. In the first round, it prioritizes requests based on membership level. If two requests have the same priority, FIFO can be employed in the second round. For instance in the previous example, request from broker b_3 will be selected in the first round. If brokers 1 and 2 have the same priority, FIFO model will be used in the second round. Thus, request from broker b_1 is accepted and from broker b_2 is rejected by provider 2.

The first two models however, are unilateral i.e., the decision is one-sided thus neglecting the other party. FIFO is provider-side model while membership employed broker-side strategy. The hybrid model is bilateral. None of these existing strategies

utilize the history data e.g., (1) decision attributes made by the broker(s) in selecting top service (i.e., best possible service is selected based on QoS and physical proximity), (2) number of tasks required by the client of the broker, and (3) number of shortlisted services for each task. These data are available at brokers end and can be shared with service providers. As such, we propose a priority model that incorporates this history data with the requests arrival sequence. Then, QoS adaptation is offers to the broker(s) that potential being rejected due to inability of the provider to maintain expected waiting time during certain load.

6.4.1 Proposed Strategy

In this subsection, we propose a strategy for SLA negotiation. We develop a model based on the queuing theory to show how a service provider receives and serves requests.

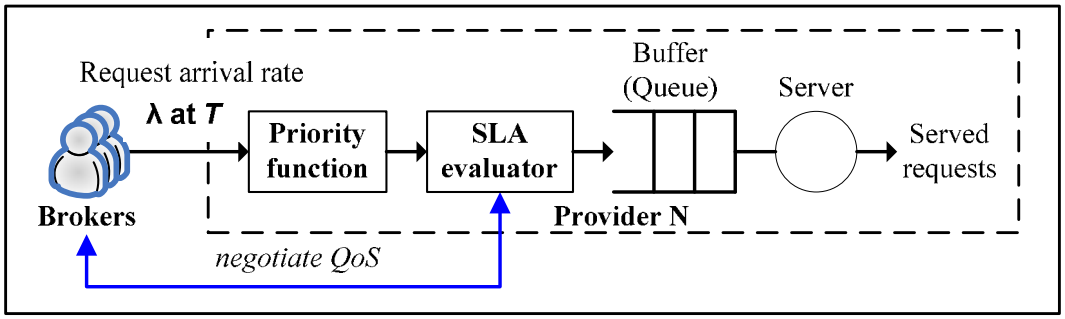

Figure 6.7: Service provider modelled as an M/G/1 queue.

Figure 6.7 depicts the SLA negotiation strategy at the service provider modelled as an M/G/1 queue. It consists of a priority function, an SLA evaluator, incoming requests, a buffer and a server. The priority function calculates the priority of each request $\in \lambda$ arriving at T and arranges them in the priority order. The SLA evaluator checks the current load and estimates how many requests it can serve within the offered waiting time. Using this information, the evaluator sends the requests (with higher priority) that can be served within $E(W)$ to the buffer and settles the SLA with the brokers. At the same time, the SLA evaluator estimates the new waiting time for the requests that potentially being rejected (due to the inability of the provider to maintain the offered waiting time during certain load).

Table 6.6 lists the commonly used notation in this paper.

Table 6.6: List of commonly used notation.

Notation	Description
λ	Requests arrival rate
E(W)	Expected waiting time
ρ	Server load
T	Mean arrival time
E(X)	Mean servicing time

6.4.1.1 Priority Function

Each request will be assigned with a priority value using the information I_i provided by each broker b_i. Given $I_i = (T^i: T_t^i, T_o^i;\ S^i: S_t^i, S_o^i; Q^i: Q_t^i, Q_{im}^i)$, T^i is the task attributes from broker b_i where T_t^i is the number of tasks and T_o^i is the order of the task to be served; S^i is the service attributes where S_t^i is the number of shortlisted services and S_o^i is the order of the provider p; and Q^i is the QoS attributes where Q_t^i is the number of QoS and Q_{im}^i is the importance of a certain QoS. A_t^i is the arrival sequence of broker b_i request. Higher value reflects higher priority being served. The priority function is formulated as the following:

$$U = \sum_{m=1}^{j} (UF_m \times w_m) \tag{6.5}$$

where, priority factor $UF = (\mathcal{T}, \mathcal{S}, \mathcal{Q}, \mathcal{A})$. $\mathcal{T}, \mathcal{S}, \mathcal{Q}, \mathcal{A}$ are task, service, QoS and arrival factors respectively. Each factor may has different importance towards the function and can be represented using different weight w_m where $0 < w_m < 1$. The total weight for all factors is $\sum w = 1$. $\mathcal{T}$ is computed using the following equation:

$$\mathcal{T} = \left(\left(\frac{T_t^i - (T_o^i - 1)}{T_t^i} \right) \times \frac{1}{T_t^i} \right) . \tag{6.6}$$

$\mathcal{S}$ is given using as the following:

$$\mathcal{S} = \left(\left(\frac{S_t^i - (S_o^i - 1)}{S_t^i} \right) \times \frac{1}{S_t^i} \right). \tag{6.7}$$

$\mathcal{Q}$ is computed using the following equation:

$$\mathcal{Q} = \left(\left(\frac{Q_{im}^i}{Q_t^i} \right) \times \left(\frac{1}{e^{\left(\frac{1}{Q_t^i} \right)}} \right) \right). \tag{6.8}$$

$\mathcal{A}$ is given as the following:

$$\mathcal{A} = \left(\frac{A_t^p - (A_t^i - 1)}{A_t^p} \right). \tag{6.9}$$

$\mathcal{A}, \mathcal{T}$ and $\mathcal{S}$ are negative monotonic factors (i.e., the lower the value, the better it is), while $\mathcal{Q}$ is a positive monotonic factor (i.e., the higher the value, the better it is). Each $\mathcal{A}, \mathcal{T}$ and $\mathcal{S}$ equation is developed by taking into account the number of attributes (e.g., number of tasks or shortlisted services) and the order or ranking of the attributes (i.e., order of the task or ranking of provider *p*), so that the computed value between two brokers differs even if (a) the number of attributes is the same but the order of the task requires servicing from provider *p* differs, and (b) the number of tasks differ but the order is the same.

Generally, the priority function takes into account multiple key factors: (a) number of tasks required by the client managed by broker b_i including the order of tasks (e.g., $\{t_1, t_2, t_3\}$); (b) number of shortlisted suitable providers for serving task *m* including the ranking of provider *p* (i.e., the provider being negotiated with) in the short listing; (c) the number of QoS including the importance (i.e., weight *w*) of each QoS - if any (e.g., time or cost QoS); and (d) request arrival sequence at the system. Note that for simplicity, two requests being arrived simultaneously will have the same arrival sequence.

Table 6.7 summarizes the characteristics of the three priority models (i.e., FIFO, membership and proposed) with regards to arrival time, priority and client utility. Please note that the proposed strategy also requires extra minimal computing overhead if compared to FIFO or membership model. FIFO and membership models however, will be countered with decision making problem if requests from brokers arrived at the same time or having the same membership levels. The proposed model is more stable towards this issue. This is an acceptable trade off gained for the extra minimal overhead. Moreover, with the proliferation of processing and memory technology, the overhead can be disregarded.

Table 6.7: Characteristics of different priority models.

Characteristic	Model		
	FIFO	**Membership**	**Proposed**
Arrival sequence consideration	**Yes**	No	**Yes**
Membership priority	No	**Yes**	No
Client/broker attributes consideration	No	No	**Yes**
Immediate rejection of N-r requests	Yes	Yes	**No (offer QoS adaptation)**

6.4.1.2 SLA Evaluator

The inputs to the SLA evaluator algorithm are the total number of requests λ, each request R_b including its priority $\overline{U}_b$, current server load ρ and $E(W)$. Using these inputs, the algorithm estimates the number of requests R_{serve} that can be performed within the $E(W)$. At the initialization, the algorithm retrieves the number of incoming requests with the corresponding priority and order, server load, and the advertised QoS offer. We assumed this information is available at the provider's system processing incoming requests. Figure 6.8 outlines the algorithm of the proposed SLA evaluator implemented at each service provider.

Algorithm 6.1: SLA Evaluator

INPUT: $\lambda, \overline{U}_b, \rho, E(W)$
OUTPUT: SLA_{id}
BEGIN
1: $R_b \leftarrow \overline{U}_b \in U$
2: $R_{serve} \leftarrow$ Estimate $(\lambda, \rho, E(W))$
3: **FOR** each R_b **DO**
4: **IF** $R_b \leq R_{serve}$ **THEN**
5: Settle agreement (R_b, SLA_{id})
6: **ELSE**
7: $E(W) \leftarrow$ Estimate (λ, ρ)
8: **IF** new $E(W)$ accepted **THEN**
9: Settle agreement (R_b, SLA_{id})
10: **ELSE**
11: Reject R_b
12: **END IF**
13: **END IF**
14: **END FOR**
END

Figure 6.8: SLA evaluator algorithm.

Based on the current server load, it estimates how many requests can be served within advertised waiting time. The estimation of *E(W)* is performed by manipulating the Pollaczek-Khintchine (P-K) formula, as the following:

$$E(W) = \lambda E(X^2)/2(1-\rho). \tag{6.10}$$

Now, we need to obtain the mean servicing time *E(X)*. Each incoming request size follows a Bounded Pareto distribution [73]. The probability density function for the Bounded Pareto $B(k, p, \alpha)$ is:

$$f(x) = \frac{\alpha k^{\alpha}}{1-(\frac{k}{p})^{\alpha}} x^{-\alpha-1}\,. \tag{6.11}$$

where α represents the task size variation, k and p are the smallest and the largest possible task size respectively, with $(k \leq x \leq p)$. Moderate variability is observed when $(\alpha \approx 2)$ and high variability when $(\alpha \approx 1)$ [73]. In order to offer new serving time *x* to the request(s) potentially being rejected, the expected waiting time *E(W)*

before a request being serviced is predicted. If the number of waiting requests *E(N)* and the mean servicing time *E(X)* are known a priori, by Little's law, the waiting time be can represented as the mean queue length. This gives the mean queue length $E(N) = \lambda * E(W)$ and load on the server $\rho = \lambda * E(X)$ [17]. Let $E(X^j)$ be the *j*-th moment of the service time distribution of the requests. Then, $E\left(X^j\right)$ is derived as the following:

$$E\left(X^j\right) = \begin{cases} \frac{\alpha p^j\left(\left(\frac{k}{p}\right)^{\alpha} - \left(\frac{k}{p}\right)^{j}\right)}{(j-\alpha)\left(1-\left(\frac{k}{p}\right)^{\alpha}\right)} & if\ j \neq \alpha \\ \frac{\alpha k^{\alpha} \ln\left(\frac{p}{k}\right)}{\left(1-\left(\frac{k}{p}\right)^{\alpha}\right)} & if\ j = \alpha \end{cases} \tag{6.12}$$

To estimate the number of requests that can be served within current load, we plot the new *E(W)*. For example, assume that a service provider advertised the waiting time QoS by taking into account the server load as fair load (0.6) and the λ as 50, as in figure 6.9.

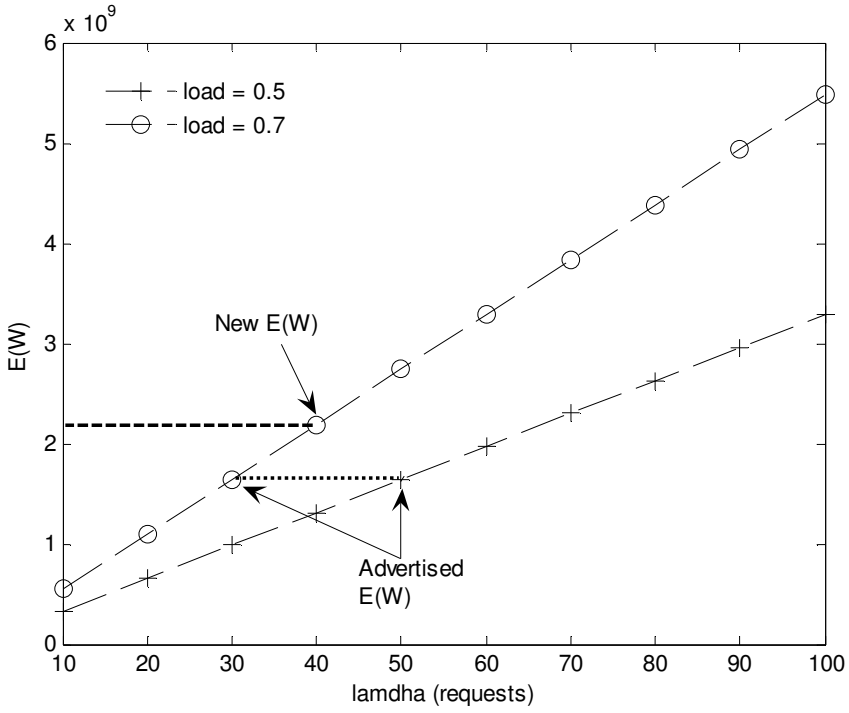

Figure 6.9: Advertised *E(W)* versus new *E(W)* to be offered based on current load.

From figure 6.9, we can see that the advertised *E(W)* is around 1.8×10^9. During actual running, the current server loads is observed around 0.7. The SLA evaluator estimates the new *E(W)* and realizes that only 30 out of 50 requests can be served within the advertised *E(W)*. It then settles the SLA with the brokers of the first 30 requests.

The SLA evaluator evaluates the new *E(W)* for the rest of the requests. For example, the next 10 requests is offered 2.2×10^9 for the new *E(W)*, while the last 10 is offered with 2.8×10^9. If the brokers of these 20 requests accept the new *E(W)* accordingly, the SLAs are settled; otherwise; the brokers' requests are rejected. For each settled agreement, both the broker and the service provider keep the same SLA version of the negotiated QoS levels.

6.4.2 Analysis of Strategy

One of the main objectives of the proposed strategy is to prioritize the requests that can be served within the advertised waiting time and to offer QoS adaptation of the waiting time (i.e., new waiting time) to the broker(s) with the request potentially being rejected. The function accurately estimates the broker(s) request priority. We present proofs on the correctness of the function. We compare the resulted priority between the two competitive brokers b_k and b_l requests. The task, service and QoS attributes of broker b_k and b_l are represented as (T^k, S^k, Q^k) and (T^l, S^l, Q^l), respectively.

Theorem 6.1: If at a certain time instant, the two competitive brokers b_k and b_l have the same (a) arrival (i.e., $A_t^k = A_t^l$); (b) number of shortlisted services (i.e., $S_t^k = S_t^l$) and ranking of the provider p (i.e., $S_o^k = S_o^l$); (c) number of QoS being considered (i.e., $Q_t^k = Q_t^l$) and importance of a certain QoS (i.e., time QoS: $Q_{im}^k = Q_{im}^l$) and (d) fixed weights during a negotiation cycle. Assume that the number of task in (T) between b_k and b_l differs (i.e., $T_t^k < T_t^l$), but the order of the task is the same (i.e., $T_o^k = T_o^l$), then the broker for the task with smaller number of task has the higher priority.

Proof: The two competitive brokers b_k and b_l have the same attributes of *A*, *S* and *Q* that can be represented as follows:

$$\{\boldsymbol{A}: A_t^k = A_t^l; \boldsymbol{S}: S_t^k = S_t^l, S_o^k = S_o^l; \boldsymbol{Q}: Q_t^k = Q_t^l, Q_{im}^k = Q_{im}^l\}$$
$$\vdash \{A^k = A^l, S^k = S^l, Q^k = Q^l\}$$

$$\therefore\ \mathcal{A}^k = \mathcal{A}^l, \mathcal{S}^k = \mathcal{S}^l, \mathcal{Q}^k = \mathcal{Q}^l\ .$$

Substituting into equation (6.5) with $\forall w^k = \forall w^l$ (i.e., $w_{\mathcal{T}}^k = w_{\mathcal{T}}^l;\ w_{\mathcal{S}}^k = w_{\mathcal{S}}^l; w_{\mathcal{Q}}^k = w_{\mathcal{Q}}^l; w_{\mathcal{A}}^k = w_{\mathcal{A}}^l$), results in different U^k and U^l that can be simplified as follows:

$$U^k = \mathcal{T}^k + \mathcal{S}^k + \mathcal{Q}^k + \mathcal{A}^k$$

$$U^l = \mathcal{T}^l + \mathcal{S}^k + \mathcal{Q}^k + \mathcal{A}^k$$

$$\because \left(T_o^k = T_o^l, T_t^k < T_t^l\right) \vdash (\mathcal{T}^k > \mathcal{T}^l) \models U^k > U^l. \quad \square$$

The priority of the broker b_k of a less number of task is higher than the broker b_l. The result is true as the less number of tasks the more relative completion percentage is achieved if the task is served.

Theorem 6.2: If at a certain time instant, the two competitive brokers b_k and b_l have the same (a) arrival (i.e., $A_t^k = A_t^l$); (b) number of shortlisted services (i.e., $S_t^k = S_t^l$) and ranking of the provider p (i.e., $S_o^k = S_o^l$); (c) number of QoS being considered (i.e., $Q_t^k = Q_t^l$) and importance of a certain QoS (i.e., time QoS: $Q_{im}^k = Q_{im}^l$) and (d) fixed weights during a negotiation cycle. Assume that the number of task in (T) between b_k and b_l is the same (i.e., $T_t^k = T_t^l$), but the order of the task differs (i.e., $T_o^k < T_o^l$), then the broker for the task with smaller order has the higher priority.

Proof: The two competitive brokers b_k and b_l have the same attributes of A, S and Q that can be represented as follows:

$$\left\{\boldsymbol{A}: A_t^k = A_t^l; \boldsymbol{S}: S_t^k = S_t^l, S_o^k = S_o^l; \boldsymbol{Q}: Q_t^k = Q_t^l, Q_{im}^k = Q_{im}^l\right\}$$

$$\vdash \{A^k = A^l, S^k = S^l, Q^k = Q^l\}$$

$$\therefore\ \mathcal{A}^k = \mathcal{A}^l, \mathcal{S}^k = \mathcal{S}^l, \mathcal{Q}^k = \mathcal{Q}^l\ .$$

Substituting into equation (6.5) with $\forall w^k = \forall w^l$, results in different U^k and U^l that can be simplified as follows:

$$U^k = \mathcal{T}^k + \mathcal{S}^k + \mathcal{Q}^k + \mathcal{A}^k$$

$$U^l = \mathcal{T}^l + \mathcal{S}^k + \mathcal{Q}^k + \mathcal{A}^k$$

$$\because \left(T_t^k = T_t^l, T_o^k < T_o^l\right) \vdash (\mathcal{T}^k > \mathcal{T}^l) \models U^k > U^l. \quad \square$$

The priority of the broker b_k of a higher order of task being required to be served is higher than the broker b_l. The result is true as the earlier the task of higher order being served, the chance to move to the next consecutive tasks is higher.

Theorem 6.3: If at a certain time instant, the two competitive brokers b_k and b_l have the same (a) arrival (i.e., $A_t^k = A_t^l$); (b) number of task (i.e., $T_t^k = T_t^l$) and order of the task n (i.e., $T_o^k = T_o^l$); (c) number of QoS being considered (i.e., $Q_t^k = Q_t^l$) and importance of a certain QoS (i.e., time QoS: $Q_{im}^k = Q_{im}^l$) and (d) fixed weights during a negotiation cycle. Assume that the number of shortlisted services in (S) between b_k and b_l differs (i.e., $S_t^k < S_t^l$), but the ranking of the service provider p is the same (i.e., $S_o^k = S_o^l$), then the broker with smaller number of shortlisted services has the higher priority.

Proof: The top two competitive brokers b_k and b_l have the same attributes of A, T and Q that can be represented as follows:

$$\{\boldsymbol{A}: A_t^k = A_t^l; \boldsymbol{T}: T_t^k = T_t^l, T_o^k = T_o^l; \boldsymbol{Q}: Q_t^k = Q_t^l, Q_{im}^k = Q_{im}^l\}$$
$$\vdash \{A^k = A^l, T^k = T^l, Q^k = Q^l\}$$
$$\therefore \mathcal{A}^k = \mathcal{A}^l, \mathcal{T}^k = \mathcal{T}^l, \mathcal{Q}^k = \mathcal{Q}^l \ .$$

Substituting into equation (6.5) with $\forall w^k = \forall w^l$, results in different U^k and U^l that can be simplified as follows:

$$U^k = \mathcal{T}^k + \mathcal{S}^k + \mathcal{Q}^k + \mathcal{A}^k$$
$$U^l = \mathcal{T}^k + \mathcal{S}^l + \mathcal{Q}^k + \mathcal{A}^k$$
$$\because \left(S_o^k = S_o^l, S_t^k < S_t^l\right) \vdash (\mathcal{S}^k > \mathcal{S}^l) \ \vDash U^k > U^l. \ \square$$

The priority of the broker b_k of a less number of shortlisted services is higher than the broker b_l. The result is true as the broker with less number of shortlisted services is having less service option if not being served thus, should have higher priority.

Theorem 6.4: If at a certain time instant, the two competitive brokers b_k and b_l have the same (a) arrival (i.e., $A_t^k = A_t^l$); (b) number of task (i.e., $T_t^k = T_t^l$) and order of the task n (i.e., $T_o^k = T_o^l$); (c) number of QoS being considered (i.e., $Q_t^k = Q_t^l$) and importance of a certain QoS (i.e., time QoS: $Q_{im}^k = Q_{im}^l$) and (d) fixed weights during a negotiation

cycle. Assume that the number of shortlisted services in (S) between b_k and b_l is the same (i.e., $S_t^k = S_t^l$), but the ranking of the service provider p differs (i.e., $S_o^k < S_o^l$), then the broker with higher ranking of provider p has the higher priority.

Proof: The two competitive brokers b_k and b_l have the same attributes of A, T and Q that can be represented as follows:

$$\{\boldsymbol{A}: A_t^k = A_t^l; \boldsymbol{T}: T_t^k = T_t^l, T_o^k = T_o^l; \boldsymbol{Q}: Q_t^k = Q_t^l, Q_{im}^k = Q_{im}^l\}$$
$$\vdash \{A^k = A^l, T^k = T^l, Q^k = Q^l\}$$
$$\therefore \ \mathcal{A}^k = \mathcal{A}^l, \mathcal{T}^k = \mathcal{T}^l, \mathcal{Q}^k = \mathcal{Q}^l \ .$$

Substituting into equation (6.5) with $\forall w^k = \forall w^l$, results in different U^k and U^l that can be simplified as follows:

$$U^k = \mathcal{T}^k + \mathcal{S}^k + \mathcal{Q}^k + \mathcal{A}^k$$
$$U^l = \mathcal{T}^k + \mathcal{S}^l + \mathcal{Q}^k + \mathcal{A}^k$$
$$\because \left(S_t^k = S_t^l, S_o^k < S_o^l\right) \vdash (\mathcal{S}^k > \mathcal{S}^l) \ \vDash U^k > U^l. \quad \square$$

The priority of the broker b_k of a higher ranking of provider p (e.g., first choice) is higher than the broker b_l (e.g., second choice). The result is true as the higher the priority of the considered provider p, the more important it is to the broker.

Theorem 6.5: If at a certain time instant, the two competitive brokers b_k and b_l have the same (a) arrival (i.e., $A_t^k = A_t^l$); (b) number of task (i.e., $T_t^k = T_t^l$) and order of the task n (i.e., $T_o^k = T_o^l$); (c) number of shortlisted services (i.e., $S_t^k = S_t^l$) and ranking of the provider p (i.e., $S_o^k = S_o^l$); and (d) fixed weights during a negotiation cycle. Assume that the number of QoS in (Q) between b_k and b_l differ (i.e., $Q_t^k > Q_t^l$), but the importance of a certain QoS is the same (i.e., time QoS: $Q_{im}^k = Q_{im}^l$), then the broker with bigger number of QoS has the higher priority.

Proof: The top two competitive brokers b_k and b_l have the same attributes of A, T and S that can be represented as follows:

$$\{\boldsymbol{A}: A_t^k = A_t^l; \boldsymbol{T}: T_t^k = T_t^l, T_o^k = T_o^l; \boldsymbol{S}: S_t^k = S_t^l, S_t^k = S_t^l\}$$
$$\vdash \{A^k = A^l, T^k = T^l, S^k = S^l\}$$
$$\therefore \ \mathcal{A}^k = \mathcal{A}^l, \mathcal{T}^k = \mathcal{T}^l, \mathcal{S}^k = \mathcal{S}^l \ .$$

Substituting into equation (6.5) with $\forall w^k = \forall w^l$, results in different U^k and U^l that can be simplified as follows:

$$U^k = \mathcal{T}^k + \mathcal{S}^k + \mathcal{Q}^k + \mathcal{A}^k$$
$$U^l = \mathcal{T}^k + \mathcal{S}^k + \mathcal{Q}^l + \mathcal{A}^k$$
$$\because \left(Q_{im}^k = Q_{im}^l, Q_t^k > Q_t^l\right) \vdash (\mathcal{Q}^k > \mathcal{Q}^l) \ \vDash U^k > U^l. \quad \square$$

The priority of the broker b_k of a bigger number of QoS is higher than the broker b_l. The result is true as the bigger number of QoS implies that the more in-depth consideration (i.e., thorough analysis) is being conducted.

Theorem 6.6: If at a certain time instant, the two competitive brokers b_k and b_l have the same (a) arrival (i.e., $A_t^k = A_t^l$); (b) number of task (i.e., $T_t^k = T_t^l$) and order of the task n (i.e., $T_o^k = T_o^l$); (c) number of shortlisted services (i.e., $S_t^k = S_t^l$) and ranking of the provider p (i.e., $S_o^k = S_o^l$); and (d) fixed weights during a negotiation cycle. Assume that the number of QoS being considered in (Q) between b_k and b_l is the same (i.e., $Q_t^k = Q_t^l$), but the importance of a certain QoS differs (i.e., time QoS: $Q_{im}^k > Q_{im}^l$), then the broker with higher importance of a certain QoS has the higher priority.

Proof: The two competitive brokers b_k and b_l have the same attributes of A, T and Q that can be represented as follows:

$$\{\boldsymbol{A}: A_t^k = A_t^l; \boldsymbol{T}: T_t^k = T_t^l, T_o^k = T_o^l; \boldsymbol{S}: S_t^k = S_t^l, S_t^k = S_t^l\}$$
$$\vdash \{A^k = A^l, T^k = T^l, S^k = S^l\}$$
$$\therefore \ \mathcal{A}^k = \mathcal{A}^l, \mathcal{T}^k = \mathcal{T}^l, \mathcal{S}^k = \mathcal{S}^l \ .$$

Substituting into equation (6.5) with $\forall w^k = \forall w^l$, results in different U^k and U^l that can be simplified as follows:

$$U^k = \mathcal{T}^k + \mathcal{S}^k + \mathcal{Q}^k + \mathcal{A}^k$$
$$U^l = \mathcal{T}^k + \mathcal{S}^k + \mathcal{Q}^l + \mathcal{A}^k$$
$$\because \left(Q_t^k = Q_t^l, Q_{im}^k > Q_{im}^l\right) \vdash (\mathcal{Q}^k > \mathcal{Q}^l) \models U^k > U^l. \ \square$$

The priority of the broker b_k of a greater importance of a certain QoS (e.g., time QoS) is higher than the broker b_l . The result is true as the greater the importance of a certain QoS, the higher priority is being given to it.

Theorem 6.7: The two competitive brokers b_k and b_l have the same (a) number of task (i.e., $T_t^k = T_t^l$) and order of the task n (i.e., $T_o^k = T_o^l$); (b) number of shortlisted services (i.e., $S_t^k = S_t^l$) and ranking of the provider p (i.e., $S_o^k = S_o^l$); (c) number of QoS being considered (i.e., $Q_t^k = Q_t^l$) and importance of a certain QoS (i.e., time QoS $Q_{im}^k = Q_{im}^l$); and (d) fixed weights during a negotiation cycle. Assume that the arrival of broker b_k and b_l at the system in term of discrete time differs (i.e., $A_t^k < A_t^l$), then the broker with earlier arrival has the higher priority.

Proof: The two competitive brokers b_k and b_l have the same attributes of *T*, *S*and *Q* that can be represented as follows:

$$\{\boldsymbol{T}: T_t^k = T_t^l, T_o^k = T_o^l; \boldsymbol{S}: S_t^k = S_t^l, S_t^k = S_t^l; \boldsymbol{Q}: Q_t^k = Q_t^l, Q_{im}^k = Q_{im}^l\}$$
$$\vdash \{T^k = T^l, S^k = S^l, Q^k = Q^l\}$$
$$\therefore \mathcal{T}^k = \mathcal{T}^l, \mathcal{S}^k = \mathcal{S}^l, \mathcal{Q}^k = \mathcal{Q}^l \ .$$

Substituting into equation (6.5) with $\forall w^k = \forall w^l$, results in different U^k and U^l that can be simplified as follows:

$$U^k = \mathcal{T}^k + \mathcal{S}^k + \mathcal{Q}^k + \mathcal{A}^k$$
$$U^l = \mathcal{T}^k + \mathcal{S}^k + \mathcal{Q}^k + \mathcal{A}^l$$
$$\because \left(A_t^k < A_t^l\right) \vdash (\mathcal{A}^k > \mathcal{A}^l) \models U^k > U^l. \ \square$$

The priority of the broker b_k of an earlier request arrival at the system is higher than the broker b_l. Using FIFO principle, the result is true as the earlier a request arrived the higher is the chance being served.

6.5 Performance Evaluation

In the simulation, we focus on the expected waiting time *E(W)* as the QoS requiring negotiation. Each arriving request enters the system, but only wiling to wait in queue for a certain advertised waiting time. Based on the current server load, the service provider estimates the actual expected waiting time for each request following their sequence in the priority buffer.

We use success rate (i.e., SLA settlement rate) to represent the number of incoming requests being accepted to be served. It is used to measure the performance of the negotiation strategy based on the expected negotiation outcome as being the fundamental evaluation criterion. Let ρ be the server load at a given provider $p \in P$. SLA settled rate is formulated as the following:

$$SLA\ settled\ rate = \frac{requests\ accepted}{number\ of\ requests}. \tag{6.13}$$

To create a controlled environment and to ensure that the experiments are repeatable, simulation is being used. We simulate the system with all possible cases considering variations in multiple factors: incoming requests, server load, and QoS adaptation being accepted (i.e., the new *E(W)* accepted by brokers). We assume that each broker is capable of communicating with each provider and both parties remain in negotiation phase until acceptance or rejection is finalized. Two different simulations were conducted to study the success rate metrics towards (1) number of incoming requests and (2) server loads.

We also simulate the potential SLA violation that might occur if the current observation of the server load contains error. The SLA evaluator estimates new *E(W)* with the observed current actual server load. We anticipate that the load observation may contain error. The potential SLA violation metric is given as the following:

$$Potential\ SLA\ violation = N_{\rho}^{0} - N_{\rho}^{error}. \tag{6.14}$$

where N_ρ^0 is the estimation of the requests can be served within the advertised E(W) based on current load ρ and N_ρ^{error} is the requests can be served within the advertised *E(W)* based on current load ρ with certain error rate of the observation.

The rejection rate is also simulated using the following equation:

$$Rejection\ rate = \frac{requests\ rejected}{number\ of\ \ requests}. \tag{6.15}$$

We followed the verification methodology described in [102]. At each run, we generated the number of incoming requests to follow Poisson process, characterized by a rate parameter λ. Table 6.8 depicted the simulation setting and parameters used in this study.

Table 6.8: Simulation settings.

Parameter	Value
λ	50 to 100
α	2
ρ	0.5 (lightly load) to 0.9 (heavy load)
k	1010.15
p	10^{10}
P.D.F of service distribution	$f(x) = \frac{\alpha k^\alpha}{1-(\frac{k}{p})^\alpha} x^{-\alpha-1}, (k \le x \le p)$
new *E(W)* acceptance rate	20% to 80%

The value we used for each parameter is in line with the current literature [73] and also reflects the actual environment. For the success rate and rejection rate analysis, we compare the proposed strategy with a baseline strategy that does not have the mechanism to perform QoS adaptation (i.e., no new *E(W)* will be offered, the requests will be rejected if they cannot be served within the advertised *E(W)*). For the potential SLA violation analysis, the baseline strategy is not load-aware. It accepts all requests without realizing the actual server load capability thus may lead to higher potential SLA violation.

6.6 Results and Discussion

Extensive simulations have been conducted. In this section, we discuss the results. Our negotiation strategy settles comparatively more requests and produces less potential SLA violation. Also, fewer requests are comparatively being dropped.

6.6.1 SLA Settlement Rate

Figure 6.10 shows the SLA settled rate ratio (y axis) as a function of the incoming requests λ (x-axis). In this simulation, we varied the λ from 50 to 100. The advertised *E(W)* at the registry takes into account λ equal to 60 and fair load (0.6). The acceptance number of QoS adaptation offered (new *E(W)*) is randomized in the range of 20% to 80%.

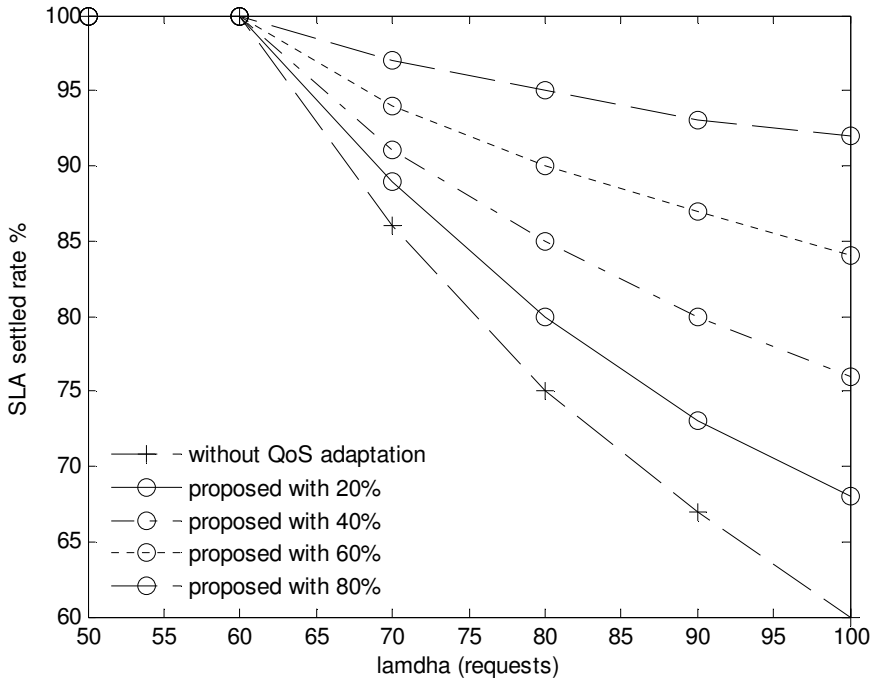

Figure 6.10: SLA settlement rate versus number of requests (server load = 0.6).

As can be seen from figure 6.10, when the current load is constant, there are a slight decrement of SLA settlement rate for the proposed approach variations and considerable decrement for the baseline approach (without QoS adaptation capability) along x-axis. The decrement margin decreases along x-axis far both approaches. The decrement margin is about 5% and 13% in average for the proposed approach variations

and baseline approach along x-axis, respectively. The proposed approach with 80% acceptance of new *E(W)* provides the highest success rate while the baseline approach (with no QoS adaptation offered) provides the least. This is due to the fact that the higher the acceptance of QoS adaptation by the broker(s) that potentially being rejected, the more agreement with the broker(s) being settled.

Figure 6.11 shows the SLA settled rate ratio (y axis) as a function of the server load (x-axis). In this simulation, we varied the current load from 0.5 to 0.9. λ is set to 60. The acceptance number of QoS adaptation offered is randomized in the range of 20% to 80%. As can be seen from figure 6.11, there is a slight decrement of SLA settled for the proposed approach variations and considerable decrement for the baseline approach (without QoS adaptation capability) along x-axis. The decrement margin increases along x-axis far both approaches. The decrement margin is about 10% and 25% in average for the proposed approach variations and baseline approach along x-axis, respectively. This is due to the fact that when the server load varies, the number of rejected requests resulted from the baseline approach is steadily increased. Thus, the inability to provide QoS adaptation (new *E(W)*) leads to significant reduction of SLA settled rate.

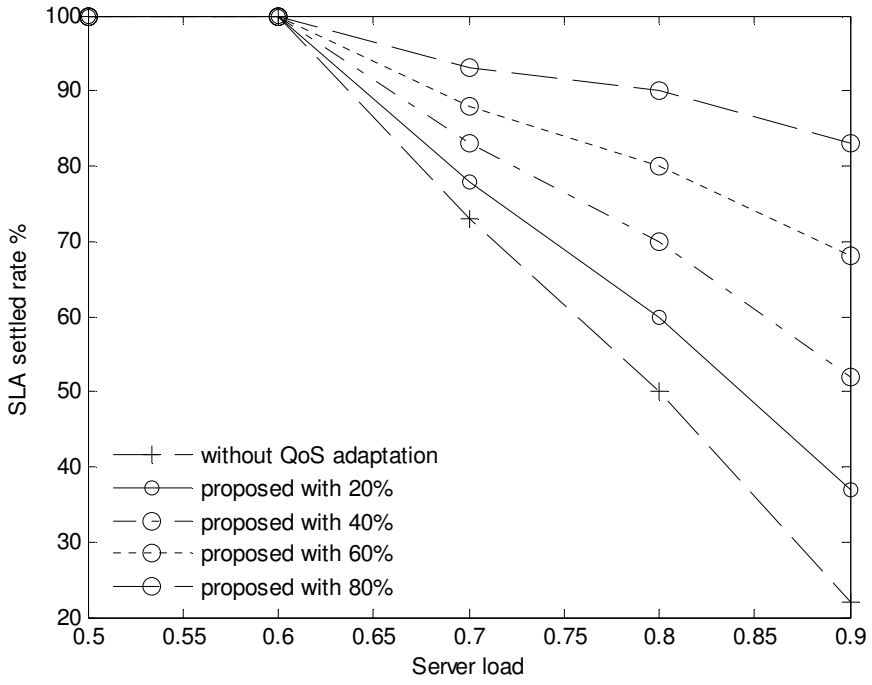

Figure 6.11: SLA settlement rate versus server load variations (λ = 60).

6.6.2 Rejection Rate

Figure 6.12 shows the requests rejection rate ratio (y axis) as a function of the server load (x-axis). In this simulation, we varied the current load from 0.5 to 0.9. λ is set to 60. The acceptance number of QoS adaptation offered is randomized in the range of 20% to 80%. As depicted from figure 6.12, the rejection rate decreases along x-axis for both approaches. For the proposed approach, the rejection rate significantly decreases when the acceptance rate of the new *E(W)* is higher. The increment margin is about 15% and 30% in average for the proposed approach variations and baseline approach along x-axis, respectively. Eventually, without the ability of QoS adaptation, the baseline approach dropped many requests when the server is operating at a higher capacity.

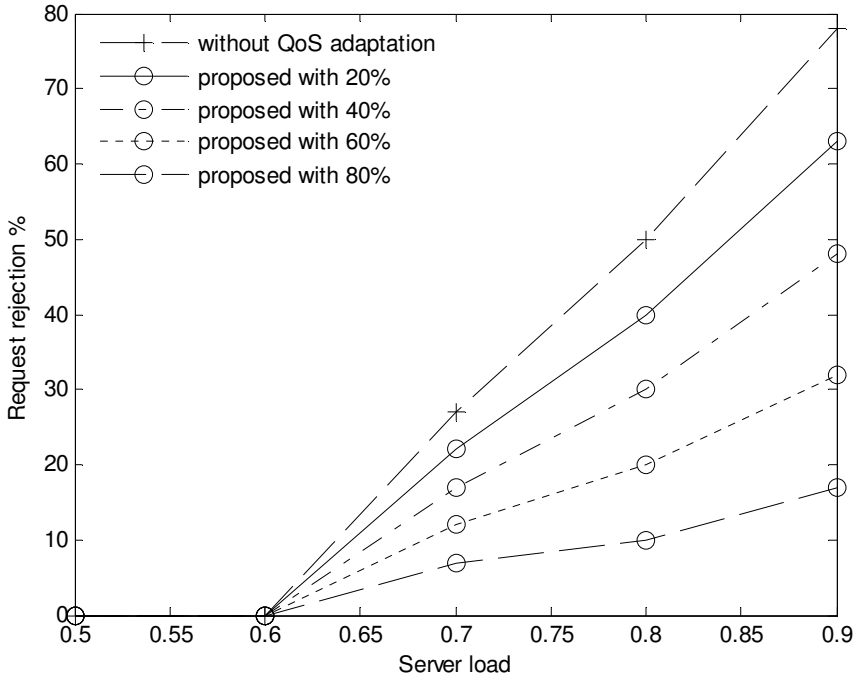

Figure 6.12: Request rejection versus server load variations (λ = 60).

6.6.3 Potential SLA Violation

Figure 6.13 shows the potential SLA violation ratio (y axis) as a function of the server load (x-axis). In this simulation, we varied the current load from 0.5 to 0.9. λ is set to 60. We set the error of the current server load observation from 0% to 10%. In this simulation, the error percentage means that the server load is under estimated, e.g., a 5%

error of 0.6 (server load) is actually 0.63; thus the SLA evaluator may allow more requests to be settled within the advertised *E(W)* without offering the new *E(W)*. The baseline approach is incapable of realizing current server load thus intend to accept all incoming requests and settled the SLA. Eventually, without the ability of realizing current server load, the baseline approach may experience significant potential SLA violation compared to the proposed approach and its error variations.

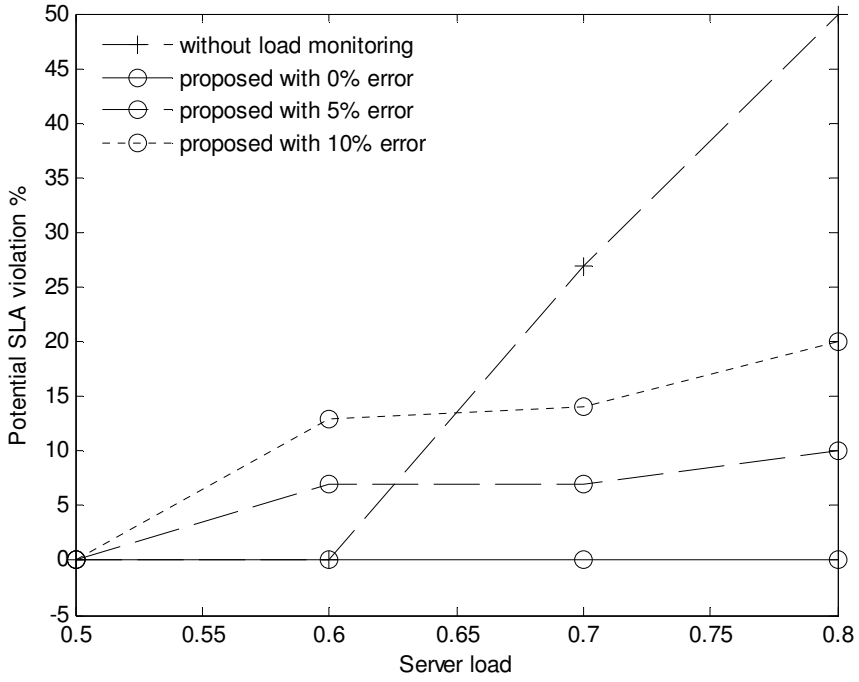

Figure 6.13. Potential SLA violation versus server load variations (λ = 60).

As can been seen from figure 6.13, the proposed approach and its error variations provide less potential SLA violation compared to the approach without the capability of load monitoring (i.e., baseline). The proposed approach with 5% error of load estimation provides around 5%, 10% error around 15% in average, and both variation does not significantly increase along x-axis compared to the baseline.

Taken as a whole, the proposed approach and its variation outperform the baseline approach in every variation of the simulations. The proposed approach benefits from to two important factors: (1) the ability to offer QoS adaptation based on current server load and (2) the new *E(W)* QoS offer being accepted by the broker(s) potentially being rejected.

6.8 Summary

In a service-oriented content adaptation scheme, clients pay for the consumed adaptation services. Thus, assuring negotiated QoS levels and adapted content version are vital. This chapter aimed to derive a framework for managing SLA. To the best of our knowledge, there is no prior work for managing SLA that is tailored to the service-oriented content adaptation platform. The framework is made of three core phases: creation, monitoring and enforcement. The creation phase includes QoS negotiation and a generic schema of an SLA tag. The monitoring phase specifies how SLA is validated and discusses essential monitoring apparatus. Compliance management in the enforcement phase includes the decision logic to determine non compliance type and the corresponding action. Within the SLA framework, we propose a detailed negotiation strategy that includes a priority function. The proposed negotiation strategy has the ability to perform QoS adaptation by taking into account the current server load.

We summarize our contributions into three: (1) we proposed a framework to manage SLA that is tailored to service-oriented content adaptation, (2) a detailed negotiation strategy is presented considering the capability of the server to serve requests within the advertised waiting time QoS, and (3) the proposed negotiation strategy is simulated in various conditions and demonstrated to perform reasonably compared to the baseline approach i.e., without the capability of QoS adaptation.

Chapter 7

Conclusion and Future Directions

The purpose of this thesis is to develop solutions to enable content adaptation being consumed as services that widely available across the network. To achieve this aim, a broker-based service-oriented content adaptation framework has been introduced. Also, we developed the enabling mechanisms for realizing the framework. This includes service discovery protocol, path determination mechanism and service level agreement management that centred on Quality of Service (QoS). Throughout the thesis, we propose relevant solutions towards realizing this aim. In this chapter, we first summarize and conclude the contributions and findings. This summary is used to perform a reality check whether our contributions are in line with the thesis objectives. Then we lay out a list of research issues for future investigations.

7.1 Conclusion

Devices, standards and software develop rapidly, but still often independently of each other. On the other hand, the current amount of digital content available online is about 487 billion gigabytes (GB) and is expected to increase rapidly [119]. To tailor the rich content suitable for a wide range of access devices with varied user preferences connected to a variety of networks, content adaptation is necessary.

Many of the existing content adaptation systems available in the literature tend to be fully or partially centralized. Problems with centralized adaptation schemes such as scalability and single-point failure are well known. In order to address these problems,

the idea of establishing distributed platform has been advocated. Present trend in content adaptation and Internet services give rise to the interest in consuming content adaptation as external services in distributed fashion. The increasing of many services offering a variety of content adaptation functions (e.g., content aggregation, filtering, annotation, transcoding, translation, conversion and extraction) shows the significance of such platform. In the thesis, the technology for interconnection of content adaptation services is termed as "service-oriented content adaptation". In this context, we have identified the fundamental research issues to address the core problems of service discovery, path determination, QoS negotiation and service level agreement. We set forth our goals to address these key issues.

This thesis investigates the existing content adaptation systems in terms of applications, features and common trends, and reveals the lack of a taxonomy to perform a categorization in terms of design themes, adaptation strategies and implementation components. To address this need, a taxonomy is developed to categorize the related solutions and being mapped to representative centralized and decentralized content adaptation systems to demonstrate its applicability. It also being used to serves as the basis for our proposed solutions related to the enabling mechanisms of service-oriented content adaptation platform.

Our analysis of the existing work in relation to service-oriented content adaptation exposes that only rudimentary frameworks exist. Moreover, there is a challenge for clients to manage content adaptation by themselves. This challenge led to the development of brokering approach to assist the clients in managing content adaptation along with required services. The proposed architecture endeavours to attain scalability, flexibility and ease of use. Based on this architecture, the broker manages the client QoS requirements, service discovery, path determination and SLA through the enabling components. Also, this architecture promotes the idea of assembling content adaptation functions into a network of services that can be loosely coupled. Establishing content adaptation as a service allows the use of a large number of adaptation mechanisms located in many places in the network. Thus, the applicability of this thesis in practical context is validated.

Service discovery protocol is developed to locate services from the network. Since content adaptation is a task that should be performed promptly; a discovery protocol

that quickly locates potential services that matched client QoS requirements is required. Also, the protocol needs to locate closer potential service providers to avoid such a high latency hops. Although there are many service discovery protocols, there is none specifically developed for service-oriented content adaptation systems or has solved the aforementioned issues simultaneously. The discovery protocol proposed in this thesis has the capability of locating closer services that matched the required adaptation functions and client QoS requirements. More importantly, it has the advantageous feature that quickly terminates search, resulting minimized searching time. The delineated claims are backed up with sufficient results to demonstrate that the proposed protocol achieves higher *discoverability*.

To select a set of services to perform the request, this thesis presents a path determination mechanism. As content adaptation request may compose of one or more tasks, at least one service is required for each task. As a single task can potentially be performed by multiple service providers, it leads to different composition possibilities. The proposed mechanism consists of a path score tree to address path construction and a single objective assignment function to choose the best possible path. The score computation logic for the assignment function compare the value at the i^{th} node with the maximum or minimum node value at the same level thus, appropriately represent each QoS value. Extensive simulation analysis, considering practical constraints and actual system parameters reveals the strengths of the proposed path determination mechanism. The proposed mechanism is proved to be substantially better than similar solution in term of generating single top path thus improving service selection execution.

The concept of SLA is quite interesting in relation to service-oriented content adaptation. In our context, a client pays for the services and for each service being consumed, the service provider promised a specific QoS levels to the client. This necessitates quality assurance as an important issue. While SLA is being neglected in existing service-oriented content adaptation systems, this thesis sets out to introduce a management framework for SLA. The proposed framework includes three interrelated phases: creation, monitoring and enforcement. Then, within the creation phase, we propose a detailed strategy for QoS negotiation that exploits the concept of priority and QoS adaptation. This strategy enables both brokers and providers to negotiate for specific QoS levels before the SLA are being settled. The proposed strategy benefits

from the ability to offer QoS adaptation based on current server load and the potential of the new QoS being accepted by the low priority requests. Simulation results demonstrate that the proposed strategy has higher SLA settlement rate, lower request rejection rate and lower potential of SLA violation compared to others.

To summarize, this thesis has laid the foundation for a broker-based service-oriented content adaptation with a novel suit of architecture model and innovative mechanisms that are used for leveraging content adaptation as services, improving performance of discovery and selection execution, and providing framework for SLA management. With these contributions, this thesis highlights some issues for future research in relation to content adaptation and service-oriented mechanisms.

7.2 Future Directions

We devised a number of future research directions in relation to this thesis. In this section, we list some potential area in relation to service-oriented content adaptation [137] and provide pointers to existing literature.

7.2.1 Solutions in Multimedia Content Distribution

Multimedia content distribution requires protocol that enables interaction and information distribution between adaptation providers. Currently, simple object access protocol (SOAP) is widely used as the protocol for messages (including content) exchange using HTTP/HTTPS or parallel HHTP in a distributed environment. It consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls (RPC) and responses [87]. Although this protocol is sufficient to handle typical content, however, a message for cross-media content adaptation requires extra elements (i.e., control information, instruction field) to be included. Thus, there is an urgent need for an "extended version" protocol that can efficiently support and manage the communication of multimedia content between adaptation providers. Researchers can refer to previous work [27, 121, 122, 123] in relation to secure session. Detailed information on the emerging protocols related to multimedia content is available in the existing literature [124, 125, 126, 127, 128, 129].

7.2.2 Service Outsourcing

In this thesis, we focus on developing efficient solution for negotiating QoS at each service provider. We foresee future research to lead to outsourcing. Service providers may not been able to meet the QoS demand by clients. On the other hand, there are many services that provide the same adaptation function. In this case, an over-committed service provider can outsource tasks to others in order to guarantee negotiated QoS levels. A new SLA can be negotiated between the new client (i.e., the provider who outsource the task) and service provider (i.e., the peer provider that undertakes the outsourced task). For this purpose, research related to peering [73, 99, 130, 131, 132] and outsourcing [133, 134, 135] can be useful. The main challenge in our context is to ensure services being outsourced are delivered according to the main SLA. In this context, a mechanism to incentivize peering or outsourcing [136] can be useful.

7.2.3 Web Design Requirements

An essential design requirement of Web content is device independence [155]. As Web applications are typically design for desktop computer screen, authors often set Web page layouts on tables and specified using fixed pixel positioning. As such, it is hard to adapt the Web content for a small device. A common solution is to create a parallel site, using cascading style sheet with device dependent styles. Clearly, this is neither practical nor feasible for authors of large volumes of content. Authors need to design Web page with device independence in mind. For example, separation of the content/data and its representation enables many different devices to be supported without unnecessary overhead during authoring. For browsing content adaptation, content might be converted from a device independent mark-up language, such as XDIME (XHTML with Device Independent Mark-up Extensions), into a form suitable for the client device such as XHTML or cHTML. The suitable device specific CSS style sheet and device specific layout might be generated from abstract style definitions and abstract layout definition respectively. In this regard, Apache Cocoon [55] and work in relation to universal design [140, 141] can be useful. For instance, Cocoon's framework keeps content and representation separate and allows parallel evolution of all aspects of a Web application. It makes it easy to support multiple output formats, thus easing the scope of content adaptation.

7.2.4 Non-technical Issues

Although content adaptation is beneficial for Web users and clients, there are some non-technical challenges need to be addressed. There can always be legal issues with content, especially when it requires modification. Content providers/owners may restrict their content from being adapted to a form that could lead to different meaning or representation. It is common that the content owner wanted to have control over the final content structure and presentation. This raises the copyright and semantic issues. The fundamental idea is to allow content adaptation while preserving the semantic of the content being adapted. In this regard, research in relation to content semantic [135, 136] and copyright enforcement [137, 138] can be useful.

7.2.5 Scalability of ADTE

We have discussed the adaptation decision-taking engine component in chapter 3 of the thesis. We foresee future research to lead to scalable ADTE. In the current implementation of existing content adaptation systems, ADTE only customized to specific adaptation contexts thus not extensible and scalable. If ADTE is provided as a service in a manner similar to other content adaptation services, the ADTE provider can always updates and maintains the decision logic to include new contexts and manage clients' adaptation context requirements. Also, ADTE should have the capability of determining the task composition and interoperability in a manner similar to [138, 139]. The client context requirements should be easily accessed by clients and can be managed in a manner similar to QoS requirements management. In this regard, research in relation to contexts management [142, 143, 144, 156] can be useful.

7.2.6 SLA Monitoring and Enforcement

Service level agreement framework proposed in this thesis is an important initial step in the service-oriented content adaptation research. In order for the framework to be fully implemented, there are several technical challenges lie ahead. Objective performance metrics for content adaptation services such as adaptation accuracy and adaptation completeness should be standardized. Each of these metrics should be able to be measured and analysed. Therefore, practical solutions in relation to the development of QoS monitoring and measurement methods are required. This should include a mechanism to evaluate the accuracy of the adapted content version. In this context, an

existing survey that includes accuracy analysis [145] can be useful. In term of SLA enforcement, the development of the decision engine for the non-compliance management is of exciting topic. An interesting aspect to explore is how this engine can learn new non-compliance cases. Also, brokers must be able to virtually manage the monitoring and enforcement of SLA. Therefore, intelligent enforcement techniques can be developed. In this regard, researchers can refer to some existing literature [146, 147, 148].

7.2.7 Energy-Aware Routing

The service-oriented content adaptation system utilizes available services that geographically distributed across wide area network. These services consume a huge amount of electricity, thus leading to high energy cost [17]. In this thesis, we focus on developing a service discovery protocol that quickly terminate when a number of matched services are located. This reduces a relatively some amount of energy required during searching. Also, the protocol always returns closer service providers but does not specifically address the routing issue. To be more energy-aware, novel techniques can be developed which reduce the energy cost through cost-aware routing strategies [149]. An energy-aware request-routing technique should take into account the providers' proximity between one to another, energy usage and cost, and incoming traffic load at the potential service providers, simultaneously. By enabling energy-efficient routing, participating providers can reduce their collective energy cost, thus reducing the environmental impact of energy consumption at the same time. Therefore, intelligent request-routing techniques can be developed. In this regard, researchers can refer to some existing literature [149, 150, 151].

7.2.8 Failure Detection and Recovery

We have discussed the models for content distribution issue in chapter 3 of the thesis. We foresee future research to lead to a fault tolerance mechanism for service-oriented content adaptation. During service execution, monitoring apparatus should also be able to detect a failed service and contact another potential service to perform the task. The main challenge is to handle failure seamlessly without seriously affecting on overall time to provide adapted content. A fault tolerance mechanism should include practical

failure detection and recovery solutions. For this purpose, research in relation to failure detection and recovery [1, 152, 153, 154] can be useful.

7.2.9 Trust Issues

The availability of a large number of networked multimedia applications and content, often required content adaptation by users, may lead to trust issues such as content being adapted by intermediaries without content's provider consent or authorization; inaccurate presentation of content semantic by intermediaries; the use of outdated content before being adapted due to caching; and addition of unnecessary objects (e.g., malicious code, marketing advertisement), that can hinder content adaptation from being widely employed and accepted by both clients and content's providers. Thus, there is an urgent need for solutions that can efficiently establish trust and manage digital right among clients, content's provider and service's provider. For this purpose, research in relation to digital right management [159] and establishing trust [158, 160] can be useful.

References

[1] Y. Fawaz, G. Berhe, L. Brunie, V-M. Scuturici and D. Coquil (2008). Efficient execution of service composition for content adaptation in pervasive computing. *Int. Journal of Digital Multimedia Broadcasting,* 1-10.

[2] R. Mohan, S. John and C-S. Li (1999). Adapting multimedia Internet content for universal access. *IEEE Trans. on Multimedia,* 1(1), 104-114.

[3] S. Ihde, P. Maglio, J. Meyer and R. Barrett (2001). Intermediary-based transcoding framework, *IBM Systems Journal* 40 (1), 179-192.

[4] S. Ardon, P. Gunningberg, B. Landfeldt, Y. Ismailov, M. Portmann, and A. Seneviratne (2003). MARCH: A distributed content adaptation architecture. *Int. Journal of Communication Systems* 16 (1), 97-115.

[5] G. Berhe, L. Brunie, and J.-M. Pierson (2005). Content adaptation in distributed multimedia systems. *Journal of Digital Information Management* 3 (2), 96-100.

[6] W. Lum and F. Lau (2003). User-centric content negotiation for effective adaptation service in mobile computing. *IEEE Trans. on Software Engineering* 29 (12), 1100-1111.

[7] M. Prangl, T. Szkalicki, and H. Hellwagner (2007). A framework for utility-based multimedia adaptation. *IEEE Trans. on Circuits and Systems for Video Technology* 17(6), 719-728.

[8] J. He, T. Gao, W. Hao, I-L. Yen and F. Bastani (2007). A flexible content adaptation system using a rule-based approach. *IEEE Trans. on Knowledge and Data Engineering* 19 (1), 127-140.

[9] J-L. Hsiao, H-P. Hung and M-S. Chen (2008). Versatile transcoding proxy for Internet content Adaptation. *IEEE Trans. on Multimedia* 10 (4), 646-658.

[10] S. Yang and N. Shao (2007). Enhancing pervasive web accessibility with rule-based adaptation strategy. *Journal of Expert Systems with Applications* 32 (4), 1154-1167.

[11] K. El-Khatib, G. Bochmann and A. El-Saddik (2004). A QoS-based framework for distributed content adaptation. *Proceedings of 1st International Conference on Quality of Services in Heterogeneous Wired/Wireless Networks*, pp. 308-312. IEEE Press: New York.

[12] M-F. Md-Fudzee and J. Abawajy (2011). QoS based service selection broker. *Future Generation Computer Systems* 27 (3), 256-264.

[13] G. Berhe, L. Brunie and J-M. Pierson (2004). Modeling service-based multimedia content adaptation in pervasive computing. *ACM Proceedings of the 1st Conference on Computing Frontiers*, pp. 60-69. ACM Press: New York.

[14] S. Tonnies, B. Kohncke, P. Hennig and W-T. Balke (2009). A service oriented architecture for personalized rich media delivery. *Proceedings of IEEE International Conference on Service Computing*, pp. 340-347. IEEE Press: New York.

[15] M-F. Md-Fudzee and J. Abawajy (2011). Request-driven cross-media content adaptation technique. In *Developing Advanced Web Services through P2P Computing and Autonomous Agents: Trends and Innovation*, Khaled Ragab, Tarek Helmy and Aboul-Ella Hassanien (eds.), pp. 91-113. IGI Global: USA.

[16] N. Nordin, W. Shin, K. Ghauth and M. Mohd (2007). Using service-based content adaptation platform to enhance mobile user experience. *Proceedings of the 4th International Conference on Mobile Technology and Applications*, pp. 552-557. ACM Press: New York.

[17] L. Snyder, F. Baskett, M. Caroll, D. Colemanm, D., Estrin, M. Furst, J. Hennessy, H. Kung, K. Maly and B. Reid (1994). *Academic Careers for Experimental Computer Scientists and Engineers*. National Academy Press: Washington.

[18] M. Forte, W. de Souza and A. do Prado (2008). Using ontologies and Web services for content adaptation in ubiquitous computing. *Journal of Systems and Software* 81(3), 368-381.

[19] M-F. Md-Fudzee and J. Abawajy (2008). Classification of content adaptation system. *Proceedings of 10th International Conference on Information Integration and Web-based Application and Services*, pp. 426-429. ACM Press: New York.

[20] D. Zhang (2007). Web content adaptation for mobile handheld devices. *Communications of ACM* 50(2), 75-79.

[21] G. Chen and D. Kotz (2000). A survey of context-aware mobile computing research. *Computer Science Technical Report TR2000-381*. Dartmouth College: Hanover NH.

[22] P. Brusilovsky (1996). Methods and techniques of adaptive hypermedia. *Journal User Modeling and User-Adapted Interaction* 6(2-3), 87-129.

[23] P. Brusilovsky, A. Kobsa and W. Nejdl (2007). *The Adaptive Web*. Springer: Berlin.

[24] R. Virgilio, R. Torlane and G-J. Houben (2007). Rule-based adaptation of Web information systems. *Journal of World Wide Web* 10 (4), 443-470.

[25] F. Adelstein, S. Gupta, G. Richard and L. Schwiebert (2005). *Fundamentals of mobile and pervasive computing - 1st edition*. McGraw-Hill: USA.

[26] S. Banerjee, S. Basu, S. Garg, S. Garg, S-J. Lee, P. Mullan and P. Sharma (2005). Scalable grid service discovery-based on UDDI. *Proceedings of 3rd International Workshop on Middleware for Grid Computing*, pp. 1-6. ACM Press: New York.

[27] K. Yunhua, Y. Danfeng and E. Bertino (2008). Efficient and secure content processing by cooperative intermediaries. *IEEE Trans. on Parallel and Distributed Systems* 19 (5), 615-626.

[28] R. Buyya, M. Pathan, and A. Vakali (eds.) (2008). *Content delivery network*. Springer-Verlag Press: Berlin.

[29] C. Canali, V. Cardellini, M. Colajanni and R. Lancellotti (2005). Performance comparison of distributed architectures for content adaptation and delivery of web resources. *Proceedings of 25th IEEE International Conference on Distributed Computing Workshops*, pp. 331-337. New York: IEEE Press.

[30] O. Buyukkokten, O. Kaljuvee, H-G. Molina, A. Paepcke and T. Winograd (2002). Efficient web browsing on handheld devices using page and form summarization. *ACM Trans. on Information System* 20 (1) 82-115.

[31] W. Lum and F. Lau (2002). A context-aware decision engine for content adaptation. *IEEE Pervasive Computing* 1 (3) 41-49.

[32] R. Farzan and P. Brusilovsky (2005). Social navigation support through annotation-based group modeling. *Proceedings of 10th International User Modeling Conference*, pp. 463-472. Springer-Verlag: Berlin.

[33] P. Brusilovsky, R. Farzan and J. Ahn (2005). Comprehensive personalized information access in an educational digital library. *Proceedings of 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 9-18. ACM Press: New York.

[34] P. Brusilovsky, S. Sosnovsky and M. Yudelson (2005). Ontology-based framework for user model interoperability in distributed learning environments**.** *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare and Higher Education*, pp. 2851-2855. AACE: Canada.

[35] Y. Fawaz, A. Negash, L. Brunie and V-M. Scuturici (2007). ConAMi: Collaboration based content adaptation middleware for pervasive computing environment. *Proceedings of IEEE International Conference on Pervasive* Services, pp. 189-192. IEEE Press: New York.

[36] H-N. Chua, S. Scott, Y. Choi and P. Blanchfield (2005). Web-page adaptation framework for PC & mobile device collaboration. *Proceedings of 19th International Conference on Advanced Information Networking and Applications*, pp. 727-732. IEEE Press: New York.

[37] F. Lopez, J. Martinez and V. Valdes (2007). Multimedia content adaptation within the CAIN framework via constraints satisfaction and optimization. *Lecture Notes on Computer System 4398*, 149-163.

[38] J. Martinez, V. Valdes, J. Bescos and L. Herranz (2005). Introducing CAIN: A metadata-driven content adaptation manager integrating heterogeneous content adaptation tools. *Workshop on Image Analysis for Multimedia Interactive Services*, pp. 1-5.

[39] B. Kohncke and W-T. Balke (2006). Personalized digital item adaptation in service-oriented environments. *Proceedings of IEEE First International Workshop on semantic Media Adaptation and Personalization*, pp. 91-96. IEEE Press: New York.

[40] I. Brunkhorst, S. Tonnies and W-T. Balke (2008). A service oriented architecture for personalized rich media delivery. *Proceedings of IEEE International Conference on Web Services*, pp. 262-269. IEEE Press: New York.

[41] W-T. Balke, and J. Diederich (2006). A quality and cost-based selection model for multimedia service composition in mobile environments. *Proceedings of IEEE International Conference on Web Services* pp. 621-628. IEEE Press: New York.

[42] J. Gecsei (1997). Adaptation in distributed multimedia systems. *IEEE Multimedia* 4 (2), 58-66.

[43] I. Augustin, A. Yamin, J. Barbosa and C. Geyer (2002). Towards taxonomy for mobile applications with adaptive behavior. Proceedings of 20th International Symposium on Parallel and Distributed Computing and Networks, pp. 224-228. ACTA Press: Canada.

[44] X. Zhou, J. Wei and C. Xu (2007). Quality-of-service differentiation on the Internet: A taxonomy. *Journal of Network and Computer Applications* 30 (1), 354-383.

[45] B. Noble, M. Satyanarayanan, D. Narayanan, J. Tilton, J. Flinn and K. Walker (1997). Agile application-aware adaptation for mobility. *SIGOPS Operating Systems Review* 31 (5), 276-287.

[46] S. Yang, J. Zhang, R. Chen and N. Shao (2007). A unit of information–based content adaptation method for improving web content accessibility in the mobile Internet. *ETRI Journal* 29 (6), 794-807.

[47] Z. Lei and N. Georganas (2001). Context-based media adaptation in pervasive computing. *Proceedings of IEEE Canadian Conference on Electrical and Computer*, pp. 913-918. IEEE Press: New York.

[48] http://www.opera.com/products/mobile/smallscreen/index (accessed on 15 September 2008).

[49] L-Q. Chen (2003). Image adaptation based on attention model for small-form-factor device. *Proceedings of the 9th International Conference on Multi-Media Modelling*, pp. 483-490. IEEE Press: New York.

[50] L-Q. Chen, X. Xie, Y. Ma, H-J. Zhang, H. Zhou and H. Feng (2002). DRESS: A slicing tree based web representation for various display size. *Technical Report MSR-TR 2002-126*. Microsoft Research: WA.

[51] Media conversion services (accessed on 15 September 2008 from http://media-convert.com).

[52] Programmable business logic components and standing data (accessed on 20 September 2008 from http://www.webservicex.net).

[53] S. Lee and K. Chung (2008). Buffer-driven adaptive video streaming with TCP-friendliness. *Computer Communications* 31 (10), 2621-2630.

[54] Java 2 Platform, Micro Edition (*J2ME*) (accessed on 10 November 2008 from http://java.sun.com/j2me).

[55] The Apache Cocoon Project (accessed on 15 November 2008 from http://cocoon.apache. org).

[56] M. Metso, A. Koivisto and J. Sauvola (2001). A content model for the mobile adaptation of multimedia information. *Journal of VLSI Signal Processing* 29 (1), 115-128.

[57] C. Muntean and J. McManis (2004). A QoS-aware adaptive web-based system. *Proceedings of IEEE International Conference on Communications*, pp. 2204-2208. IEEE Press: New York.

[58] C. Muntean and J. McManis (2006). Fine grained content-based adaptation mechanism for providing high end-user quality of experience with adaptive hypermedia systems. *Proceedings of the 15th International Conference on World Wide Web*, pp. 53-62. ACM Press: New York.

[59] L. Ramaswamy, A. Iyengar, L. Liu, F. Douglas (2005). Automatic fragment detection in dynamic Web pages and its impact on caching. *IEEE Trans.on Knowledge and Data Engineering* 17 (6): 859 - 874.

[60] M. Tong, Z. Yangand Q. Liu (2010). A novel model of adaptation decision-taking engine in multimedia adaptation. *Journal of Network and Computer Applications* 33 (1), 43-49.

[61] S. Pastore (2008). The service discovery methods issue: A web services UDDI specification framework integrated in a grid environment. *Journal of Network and Computer Applications* 31 (2), 93-107.

[62] C. Zhou, L. Chia and B. Lee (2005). Semantics in service discovery and QoS measurement. *IEEE IT Professional* 7 (2), 29-34.

[63] R. Marin-Perianu, P. Hartel and H. Scholten (2005). A classification of service discovery protocols. *TR-CTIT-05-25*. University of Twente: Netherlands.

[64] G. Meditskos and n. Bassiliades (2010). Structural and role-oriented Web service discovery with taxonomies in OWL-S. *IEEE Trans. Knowledge & Data Engineering* 22(2), 278-290.

[65] Q. Yu, X. Liu, A. Bouuettaya and B. Medjahed (2008). Deploying and managing web services: issues, solutions, and direction. *THE VLDB Journal* 17 (3), 537-572.

[66] K. Kritikos and D. Plexousakis (2009). Requirements for QoS-based Web service description and discovery. *IEEE Trans. on Service Computing* 2(4), 320-327.

[67] X. Song and W. Dou (2011). A workflow framework for intelligent service composition. *Future Generation Computer Systems* 27 (5), 627-636.

[68] K. Kritikos and D. Plexousakis (2009). Mixed-integer programming for QoS-based Web service matchmaking. *IEEE Trans. on Service Computing* 2(2), 122-139.

[69] Z. Xu, P. Martin, W. Powley and F. Zulkarnain (2007). Reputation-enhanced QoS-based Web services discovery. *Proceedings of IEEE Conference on Web Services*, pp. 249-256. IEEE Press: New York.

[70] I. Brandic, D. Music, P. Leitner and S. Dustdar (2009). VieSLAF framework: enabling adaptive and versatile SLA-management. *Lecture Notes on Computer System 5745*, 60-73.

[71] D. Menasce (2002). QoS issues in web services. *IEEE Internet Computing* 6 (6), 72-75.

[72] J. Park, J. Baek and J. Hong (2001). Management of service level agreements for multimedia Internet service using a utility model. *IEEE Communications Magazine* 39 (5), 100-106.

[73] M. Pathan and R. Buyya (2009). Architecture and performance models for QoS-driven effective peering of content delivery network. *Multiagent and Grid System* (5) 2, 1574-1702.

[74] A. Sahai, A. Durante and V. Machiraju (2002). Towards automated SLA management for web services. *Technical Report HPL-2001-310*. Hewlett-Packard Labs: CA.

[75] C. Zhou, L. Chia and B. Lee (2004). DAML-QoS ontology for web services. *Proceedings of IEEE International Conference on Web Services,* pp. 472-479. IEEE Press: New York.

[76] A. Keller and H. Ludwig (2003). The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and System Management* 11 (1), 57-81.

[77] S. Merat, A. Abdelhak and A. Hamid (2008). Content adaptation: requirements and architecture. *Proceedings of 10th International Conference on Information Integration and Web-based Application and Services*, pp. 626-629. ACM Press: New York.

[78] D. Booth, et al. (eds) (2004). W3C working group note 11: Web services architecture (accessed on 6 Nov. 2008 from http://www.w3.org/TR/ws-arch/#stakeholder)

[79] S. Hashimi (2001). Service-oriented architecture explained (accessed on 6 Nov. 2008 from http://www.ondonet.com/pub/a/dotnet/2003/08/18/soa_explained.html)

[80] Http-hypertext transfer protocol (accessed on 15 Sept. 2009 from http://www.w3.org/Protocols/)

[81] Z. Malik and A. Bouguettaya (2009). Reputation bootstrapping for trust establishment among Web services. *IEEE Internet Computing* 13(1), 40-47.

[82] M. Prangl, H. Hellwagner and T. Szkalicki (2006). Fast adaptation decision taking for cross-modal multimedia content adaptation. *Proceedings of International Conference on Multimedia Expo*, pp. 137-140. IEEE Press: New York.

[83] U. Chakravarthy, J. Grant and J. Minker (1990). Logic-based approach to semantic query optimization. *ACM Transactions on Database Systems* 15(2), 162-207.

[84] J. Feldman (2008). Theory research at Google: On distribution symmetric streaming computations. *ACM SIGACT News* 39 (2), 12-14.

[85] W. Huang, Y. Shi, S. Zhang and Y. Zhu (2006). The communication complexity of the hamming distance problem. *Journal of Information Processing Letters* 99 (4), 149-153.

[86] T-S. Jayram, R. Kumar and D. Sivakumar (2008). The one way communication complexity of hamming distance. *Journal of Theory of Computing* 4 (1), 129-135.

[87] P. Rajesh, S. Ranjiith, P-R. Soumya, V. Karthik and S. Datthathreya (2006). Network management system using Web services and service oriented architecture – A case study. *Proceedings of 10th IEEE/IFIP Network Operations and Management Symposium,* pp. 1-4. IEEE Press: New York.

[88] E. Christensen, F. Curbera, G. Meredith and S. Weerawarana (2001). Web Services description language (WSDL) 1.1. *W3C Note 20010315* (accessed on 6 Nov. 2008 from http://www.w3.org/TR/wsdl)

[89] I. Jorstad, S. Dustdar and D. Thanh (2005). A service oriented architecture framework for collaborative services. *Proceedings of 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pp. 121-125. IEEE Press: New York.

[90] M. Brock and A. Goscinki (2008). Publishing dynamic state changes of grid resources through state aware WSDL. *8th IEEE of International Conference on Web Services*, pp. 449-456. IEEE Press: New York.

[91] UDDI version 3.0.2 (accessed on 25 Nov. 2009 from http://www.uddi.org /pubs/uddi_v3.htm)

[92] D. Comer (2006). *Internetworking with TCP/IP: Principles, protocols and architecture (5th edition)*. Prentice Hall Press: England.

[93] Fielding, R. (2000). *Architectural styles and the design of network-based software architectures*. Unpublished doctoral thesis. University of California: Irvine.

[94] D. Pountain and C. Szyperski (1994). Extensible software systems. *Byte Magazine*, 57-62.

[95] M. Kassoff, C. Petrie, L-M. Zen and M. Genesereth (2009). Semantic email addressing. *IEEE Internet Computing* 13 (1), 48-55.

[96] C. Ghezzi, M. Jazayeri and D. Mandrioli (1991). *Fundamentals of software engineering*. Prentice-Hall Press: New York.

[97] S-H. Liu, Y. Cao, M. Li, P. Kilaru, T. Smith and S. Toner (2008). A semantics-and data-driven SOA for biomedical multimedia systems. *Proceedings of 10th IEEE International Symposium on Multimedia*, pp.553-558. IEEE Press: New York.

[98] A. Dastjerdi, S. Tabatabaei and R. Buyya (2010). An effective architecture for automated appliance management system applying ontology-based cloud

discovery. *Proceedings of 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid*, pp.104-112. IEEE Press: New York.

[99] M. Pathan and R. Buyya (2009). Resource discovery and request-redirection for dynamic load sharing in multi-provider peering content delivery network. *Journal of Network and Computer Applications* 32 (5), 976-990.

[100] IP address to location translator (accessed on 15 January 2010 from www.ipaddresslocation.org)

[101] D. Chakraborty, A. Joshi, Y. Yesha and T. Finin (2002). GSD: A novel group-based service discovery protocol for MANETS. *Proceedings of 4th IEEE Conference on Mobile and Wireless Communications Networks*, pp. 140-144. IEEE Press: New York.

[102] B. Javadi, J. Abawajy and M. Akbari (2008). An analytical model of interconnection networks for multi-cluster computing systems**.** *Int. Journal of Concurrency and Computation: Practice and Experience* 20(1), 75-97.

[103] B. Gleb, P. Ana and C. Tomasa (2007). *Aggregation functions: a guide for practitioners*. Springer-Verlag: Berlin.

[104] K. Roses (2007). *Discrete Mathematics and Its Applications, 6th editions*. McGraw Hill: USA.

[105] A. Azzalini and A. Capitanio (2003). *Distributions generated by perturbation of symmetry with emphasis on a multivariate skew-t distribution. Journal of Royal Statistic Society* 65 (series B), 367-389. (Generator tool accessed on 10 May 2009 from http://azzalini.stat.unipd.it/SN/sn-random.html)

[106] A. Lo (2005). Reconciling efficient markets with behavioral finance: The adaptive markets hypothesis. *Journal of Investment Consulting* 7, 21–44.

[107] I. Aziz, N. Haron, M. Mehat, L. Jung, A. Mustafa and E. Akhir (2009). Solving Travelling Sale Problem on Cluster Compute Nodes. *WSEAS Trans. on Computers* 8(6), 1020-1029.

[108] E. Al-Masri and Q. Mahmoud (2007). QoS-based discovery and ranking of Web services. *Proceeding of 16th International Conference on Computer Communications and Networks*, pp. 529-534. IEEE Press: New York.

[109] G. Klir and B. Yuan (1995). *Fuzzy Logic: Theory and Applications*, Prentice-Hall: NJ.

[110] J. Trienekens, J. Bouman and M. Zwan (2004). Specification of service level agreements: Problems, principles and practices. *Software Quality Journal* 12 (1), 43-57.

[111] P. Bhoj, S. Singhal and S. Chutani (2001). SLA management in federated environments. *Journal of Computer Networks* 35 (1), 5-24.

[112] J. Prokkola (2007). QoS measurement methods and tools. *Easy Wireless Workshop of 16th IST Mobile and Wireless Communications Summit, Budapest, Hungary.*

[113] Yahoo! language translator (accessed on 15 may 2010 from http://babelfish.yahoo.com).

[114] Text keyword extractor (accessed on 16 May 2010 from http://www. extractorlive.com/ on_line_demo.html)

[115] E-Health web site (accessed on 15 May 2010 from http://www.uri.edu/e-health /index.php)

[116] P. Balakrishnanand and T. Somasundaram (2011). SLA enabled CARE resource broker. *Future Generation Computer Systems* 27 (3), 265-279.

[117] G. Kersten and H. Lai (2007). Satisfiability and completeness of protocols for electronic negotiations. *European Journal of Operational Research* 180 (2), 922-937.

[118] S. Fatima, M. Woolridge and N. Jennings (2006). Multi issue negotiation with deadlines. *Journal of Artificial Intelligence Research* 27 (1), 381-417.

[119] J. Gantz and D. Reinsel (2010). The digital universe decade- Are you ready? *IDC Digital Universe Report May 2010* (accessed on 21 Mac 2011 from http://www.emc.com/digital_universe).

[120] J-F. Cordeau, D. Amico and M. Iori (2010). Branch-and-cut for the pickup and delivery traveling salesman problem with FIFO loading. *Computer Operating Research* 37 (5), 970–980.

[121] C. Dilloway and G. Lowe (2008). Specifying secure transport channels. Proceedings of 21st IEEE Computer Security Foundations Symposium, pp. 210-223. IEEE Press: New York.

[122] OASIS Security Services Technical Committee (2005). Security assertion markup language (SAML) v2.0 technical overview (accessed on 15 Mac 2011 from http://www.oasis-open.org/committees/security/).

[123] M. Bugliesi and R. Focard (2008). Language based secure communication. *Proceedings of the 21st IEEE Computer Security Foundations Symposium*, pp. 3-6. IEEE Press: New York.

[124] S. Boll and W. Klas (2001). ZYX - A multimedia document model for reuse and adaptation of multimedia content. *IEEE Trans. on Knowledge and Data Engineering* 13 (3), 361-382

[125] S. Amir, Y. Benabbas, I. Marius and C. Djeraba (2011). MuMIe: A new system for multimedia metadata interoperability. *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, pp. 1-8. ACM Press: New York.

[126] F. Pereira, A. Vetro and T. Sikora (2008). Multimedia retrieval and delivery: Essential metadata challenges and standards. *Proceedings of the IEEE* 96 (4), 721-744.

[127] S. Amir, I. Bilasco, M-S. Haidar and C. Djeraba (2010). Towards a unified multimedia metadata management solution. In *Intelligent Multimedia Databases and Information Retrieval - Advancing Applications and Technologies*, Y. Li and M. Zongmin (eds). IGI Global: USA.

[128] Sharable Content Object Reference Model (SCORM) (accessed on 15 June 2011 from http://www.adlnet.org).

[129] Metadata Encoding & Transmission Standard (accessed on 15 June 2011 from http://www.loc.gov/standards/mets).

[130] Z. Lu, J. Wu, C. Xiao, W. Fu and Y. Zhong (2009). WS-CDSP: A Novel Web services-based content delivery service peering scheme. *Proceedings of 6th IEEE International Conference on Services Computing*, pp. 348–355. IEEE Press: New York.

[131] Z. Lu, J. Wu and W. Fu (2010). Towards a novel Web services standard-supported CDN-P2P loosely-coupled hybrid and management model. *Proceedings of 7th IEEE International Conference on Services Computing*, pp. 297-304. IEEE Press: New York.

[132] J. Wang, C. Huang and J. Li (2008). On ISP-friendly rate allocation for peer-assisted VoD. *Proceedings of the 16th ACM International Conference on Multimedia*, pp. 279–288. ACM Press: New York.

[133] A. Conry-Murray (2008). Bye-bye independent managed security providers. Network Computing 17 (24), 18-18.

[134] D. Dey, M. Fan, and C. Zhang (2010). Design and analysis of contracts for software outsourcing. *Information Systems Research* 21 (1), 93-114.

[135] H. Motahari-Nezhad, B. Stephenson and S. Singhal (2009). Outsourcing business to cloud computing services: Opportunities and challenges. *Technical Report HPL-2009-23.* Hewlett-Packard Labs: CA.

[136] J. Jaramillo and R. Srikant (2010). A game theory based reputation mechanism to incentivize cooperation in wireless ad hoc networks. *Ad Hoc Networks* 8 (4), 416-429.

[137] M-P. Papazoglou, P. Traverso, S. Dustdar and F. Leymann (2007). Service-oriented computing: State of the art and research challenges. *IEEE Computer* 40 (11), 38-45.

[138] H. Motahari-Nezhad, B. Benatallah, F. Casati and F. Toumani (2006). Web services interoperability specifications, *IEEE Computer* (39) 5, 24-32.

[139] A. Brogi, S. Corfini and R. Popescu (2008). Semantics-based composition-oriented discovery of Web services. *ACM Trans. Internet Technology* 8 (4), 1-39.

[140] W. Chisholm and M. May (2008). *Universal design for web application: Web applications that reach everyone*. O'Reilly Media: USA.

[141] G-R. Loretta and S-W. Andi (2008). WCAG 2.0: a web accessibility standard for the evolving web. *Proceedings of International Cross-disciplinary Conference on Web Accessibility*, pp. 109-115. ACM Press: New York.

[142] A. Carreras, J. Delgado, E. Rodríguez, V. Barbosa, M. Andrade, A-H. Kodikara, S. Dogan and A. Kondoz (2010). A Platform for context-Aware and digital rights management-enabled content adaptation. *IEEE Multimedia* 17 (2), 74-89.

[143] P. Davys and B. Yolande (2005). Adaptive context management using a component-based approach. *Lecture Notes in Computer Science 3543*, 1080-1082.

[144] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan and D. Riboni (2010). A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing* 6 (2), 161-180.

[145] V. Adzic, H. Kalva and B. Furht (2011). A survey of multimedia content adaptation for mobile devices. *Multimedia Tools and Applications* 51(1), 379-396.

[146] M. Macías, G. Smith, O. Rana, J. Guitart and J. Torres (2010). Enforcing service level agreements using an economically enhanced resource manager. In *Economic Models and Algorithms for Distributed Systems*, D. Neumann, M. Baker, J. Altmann and O. Rana (eds.), pp. 109-127. Birkhäuser Basel Press: German.

[147] W. Funika, B. Kryza, R. Slota, J. Kitowski, K. Skalkowski, J. Sendor and D. Krol (2010). Monitoring of SLA parameters within VO for the SOA paradigm. In *Parallel Processing and Applied Mathematics*, R. Wyrzykowski, J. Dongarra, K. Karczewski and J. Wasniewski (eds.), pp. 115-124. Springer: Heidelberg.

[148] C. Marinos, V. Pouli, M. Grammatikou and V. Maglaris (2010). Inter-domain SLA enforcement in QoS-enabled networks. In *Remote Instrumentation and Virtual Laboratories*, F. Davoli, N. Meyer, R, Pugliese, Roberto and S. Zappatore (eds.), pp. 351-359. Springer: US.

[149] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag and B. Maggs (2009). Cutting the electric bill for Internet-scale systems. *ACM SIGCOMM Computer Communication Review* 39 (4), 123-134.

[150] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao and F. Zhao (2008). Energy-aware server provisioning and load dispatching for connection-intensive internet services. *Proceedings of USENIX/ACM Symposium on Networked Systems Design and Implementation*, pp. 337-350. USENIX Association Berkeley: CA.

[151] J. Zhu and X. Wang (2010). Model and protocol for energy efficient routing over mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, http://doi.ieeecomputersociety.org/10.1109/TMC.2010.259.

[152] Z. Zheng and M-R. Lyu (2010). An adaptive QoS-aware fault tolerance strategy for web services. *Empirical Software Engineering* 15 (4), 323-345.

[153] A. Liu, Q. Li, L. Huang and M. Xiao (2010). FACTS: A framework for fault-tolerant composition of transactional Web services. *IEEE Trans. on Services Computing* 3 (1), 46-59.

[154] G. Kotonya and S. Hall (2010). A differentiation-aware fault-tolerant framework for Web services. *Lecture Notes Computer Science 6470*, 137-151.

[155] T. Zahariadis, C. Lamy-Bergot, T. Schierl, K. Gruneberg, L. Celetto and C. Timmerer (2009). Content adaptation issues in the future Internet. In *Towards*

the Future Internet — A European Research Perspective, G. Tselentis et al. (eds.), pp. 283-292. IOS Press: Netherlands.

[156] M. Oskar, P. Tilanus, M. Schenk, E. Cramer and J. Adriaanse (2009). Future Internet: Towards context information brokering. In *Towards the Future Internet — A European Research Perspective*, G. Tselentis et al. (eds.), pp. 250-262. IOS Press: Netherlands.

[157] W. Zeng, J. Lan and X. Zhuang (2005). Security architectures and analysis for content adaptation. *Proceedings of SPIE/IS&T VII,* pp. 84-95. SPIE: WA.

[158] C-H. Chi and L. Liu (2006). Content delivered on Internet- How much can we trust? *Proceedings of 2nd International Conference on Semantics, Knowledge and Grid,* pp. 81-84. IEEE Press: New York.

[159] C-M. Ou and C-R. Ou (2011). Adaptation of agent-based non-repudiation protocol to mobile digital right management. *Expert System with Applications* 38 (9), 11048-11054.

[160] J. Abawajy (2011). Establishing trust in hybrid cloud computing environments. *Proceedings of IEEE TrustCom.* IEEE Press: New York.

[161] M-P. Papazoglou W-J. Heuvel (2007). Service-oriented architectures: Approaches, technologies and research issues. *The VLDB Journal* 16, 389-415.

[162] M-P. Papazoglou, P. Traverso, S. Dustdar and F. Leymann (2008). Service-oriented computing: A research roadmap. *International Journal of Cooperation Information Systems* 17 (2), 223-255.