UNIVERSIDADE DE COIMBRA

Faculdade de Ciências e Tecnologia

Departamento de Engenharia Informática

QoS for Multi-Domain Services in Next Generation Networks

(QoS para Serviços Multi-Domínios em Redes de Próxima Geração)

Fernando Menezes Matos

PhD Thesis Submitted to the University of Coimbra

(Tese de Doutoramento submetida à Universidade de Coimbra)

· 8 de Novembro de 2011 ·

QoS para Serviços Multi-Domínios em Redes de Próxima Geração

Tese de Doutoramento submetida à Universidade de Coimbra

Fernando Menezes Matos

Orientador: Professor Doutor Edmundo Heitor da Silva Monteiro

Co-orientador: Professor Doutor Paulo Alexandre Ferreira Simões

· 8 de Novembro de 2011 ·

Agradecimentos

Aos meus pais por sempre terem me dado apoio em todas empreitadas, mesmo sabendo que algumas delas me levariam para longe de casa.

Aos meus irmãos por todo o companheirismo, lealdade e por não sentirem ciúmes quando o almoço do dia era feito especialmente para mim.

À Pavla por todo o amor, paciência e carinho dedicados em Portugal e no Brasil.

Aos meus orientadores por todo o apoio, orientação e críticas que serviram para a realização deste trabalho. Obrigado também pela paciência ao sanarem todas a dúvidas e servirem de inspiração como investigadores.

Ao Dr. Edmundo Monteiro, por ter acreditado em mim e ter me dado forças para concluir este doutoramento.

Ao Dr. Paulo Simões por toda o conhecimento e incentivo dados durante este árduo caminho.

Ao Adriano, Alexandre, Dudu, Neto, Vinícius e André por todas as discussões técnicas e "filosóficas" que me ajudaram de alguma forma a concluir este trabalho.

Ao Celso, Anderson, Simon e Martinez por terem propiciado uma ótima convivência nos anos de morada em Portugal.

Aos amigos de Coimbra e internacionais que permitiram que este período de doutoramento não fosse apenas uma fase de aprendizado técnico e acadêmico, mas também uma fase de alegria e companheirismo. Dentre os amigos, cito o Adriano, Alexandre, André, Dudu, Neto, Vinícius, Alice, Martinez, Renata, Simon, Eudis, Josi, Cati, Gerardo, Dien, Juci, Celso, Anderson, Patrick, Vivi e tantos outros que fizeram parte da minha vida em Coimbra.

Aos amigos do Brasil, que mesmo de longe sempre me apoiaram.

À Fundação para a Ciência e Tecnologia (FCT), pelo apoio financeiro

Agradeço também à uma força superior, a qual eu acredito que nos guia.

Foreword

The work described in this thesis was conducted at the Laboratory of Communications and Telematics of the Centre for Informatics and Systems of the University of Coimbra within the context of the following project:

- Project OpenNET – *Open Interconnect for the Internet Community* – Project funded by the Sixth Framework Programme (FP6) of the European Union from June 2006 to May 2008. The objective of this project is to define the QoS parameter values for Premium IP services that cross multiple domains. In doing this, Internet Service Providers (ISP) will be able to define compatible thresholds of their QoS parameters, such as delay variation and loss.

The work done during this thesis resulted in the following publications:

- F. Matos, A. Matos, P. Simões and E. Monteiro. "Provisioning of Inter-Domain QoS-Aware Services". *Submitted to Springer Journal of Network and Systems Management (JNSM)*, 2011.
- F. Matos, A. Matos, P. Simões and E. Monteiro. "QoS Adaptation in Inter-Domain Services". *In the 12th IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2011, Dublin, Ireland.
- F. Matos, A. Matos, P. Simões and E. Monteiro. "A service composition approach for inter-domain provisioning". *In the 6th IEEE/IFIP International Conference on Network and Service Management (CNSM)*, 2010, Niagara Falls, Canada.
- F. Matos, A. Matos, P. Simões and E. Monteiro. "A QoS Model for Business Layers". *In the 24th International Conference on Information Networking (ICOIN 2010)*, 2010, Busan, South Korea.

The following publications concerning inter-domain VPNs were co-authored by the candidate

- A. Matos, F. Matos, P. Simões and E. Monteiro. "An IPSec mediation approach for safe establishment of inter-domain VPNs". *In 9th IEEE International Workshop on IP Operations and Management (IPOM 09)*, 2009, Venice, Italy.
- A. Matos, F. Matos, P. Simões and E. Monteiro. "A Framework for the establishment of inter-domain, on-demand VPNs". *In IEEE/IFIP Network Operations and Management Symposium (NOMS'08)*, 2008, Salvador, Brazil.

In addition, the work done during this thesis resulted in the following extended abstract:

- F. Matos, P. Simões and E. Monteiro. "QoS Models for Business Layers". *Extended Abstract at Trilogy Future Internet Summer School*, 2009, Louvain-la-Neuve, Belgium.

Resumo

Com o aumento da ubiquidade da Internet ocorrido nos últimos anos, os provedores de serviços constataram que eles teriam a possibilidade de oferecer serviços a usuários que estão além de seus domínios administrativos. Em contrapartida, devido à evolução dos equipamentos, há uma crescente demanda por parte destes usuários por serviços de melhor qualidade. Contudo, prover serviços de valor agregado em ambientes heterogêneos, tais como ambientes multi-domínios/multi-provedores, é uma tarefa complexa. No domínio da Internet, provedores operam com diferentes políticas, equipamentos e objetivos de negócios. Desta forma, quando necessitam prover serviços inter-domínios, os provedores dependem de negociações realizadas por pessoas e de configurações manuais de seus equipamentos, o que dificulta o estabelecimento de serviços de curta duração e que necessitem estar disponíveis imediatamente, tais como vídeo conferências e transmissões de TV sobre IP (IPTV).

Assim sendo, o trabalho apresentado nesta tese foca no problema de provisionamento de serviços com qualidade em ambientes inter-domínios. Neste contexto, um modelo de qualidade de serviço chamado Quality of Service (QoS) for Inter-Domain Services (QIDS) foi proposto. QIDS tem por objetivo possibilitar que serviços inter-domínios sejam definidos, configurados e adaptados de maneira automática, dinâmica e sob-demanda. Isto pode ser alcançado através dos seguintes mecanismos: (i) utilização de um canal de comunicação comum aos provedores (camada de negócios), onde eles podem publicar e procurar por serviços e interagir com outros provedores para contratar e gerir estes serviços; (ii) definição de modelos para especificar as características técnicas e de negócios dos serviços; (iii) composição automática de serviços utilizando elementos de serviços (serviços menores), levando em consideração parâmetros de desempenho, específicos de serviços e de negócios; (iv) estabelecimento automático de acordos de nível de serviço (contratos) entre as partes envolvidas de forma a garantir os requisitos do serviço; (v) criação e execução de regras de configuração nos equipamentos de rede; e (vi) adaptação da qualidade do serviço inter-domínio caso ocorra violações de contrato.

Com o objetivo de validar o QIDS, um protótipo foi implementado e testes foram realizados neste protótipo. O cenário empregado na realização dos testes consiste no estabelecimento de uma rede privada virtual (VPN) inter-domínio utilizando as tecnologias Border Gateway Protocol (BGP) e Multiprotocol Label Switching (MPLS) (BGP/MPLS VPN) para dar suporte ao provisionamento de serviços de transmissão de vídeo. Os resultados dos testes são apresentados e discutidos nesta tese. Estes resultados corroboram os objetivos principais do QIDS, que constituem em compor e estabelecer serviços inter-domínios com qualidade e adaptar estes serviços de forma eficiente e dinâmica.

Abstract

In the last years providers have come to realize that, with the increasing ubiquity of the Internet, they can reach customers that are beyond their administrative domains. At the same time, there is a growing customer demand for services of better quality as the network equipment evolves to support these services. However, providing value-added services in multi-domain/multi-provider environments is a complex task due to the intrinsic heterogeneity of these environments. In the Internet, providers are likely to have different policies, equipment and business goals. As a result, when providers want to establish an inter-domain service, they have to rely on human-based negotiation and manual configuration of equipment, which makes it difficult to establish more granular, "immediately available" and short-term services, such as video conference or Internet Protocol Television (IPTV) transmission.

For that reason, this study addresses the problem of providing QoS-aware services in inter-domain environments. In this context, Quality of Service (QoS) for Inter-Domain Services (QIDS) is proposed to allow inter-domain QoS-aware services to be defined, configured and adapted in an automatic, dynamic and on-demand fashion. This can be achieved by the following measures: (i) using a common communication channel (business layer) through which providers can publish and search for services, and interact with each other to contract and manage these services; (ii) defining templates to specify the business and technical characteristics of the services; (iii) automatically composing services by means of service elements (smaller services) in accordance with QoS performance, service-specific, and business parameters; (iv) automatically establishing Service Level Agreement (SLA) contracts between the involved parties to guarantee the service requirements; (v) creating and enforcing configuration rules on underlying equipment; and (vi) adapting the QoS of the inter-domain service in case of contract violations.

A prototype was implemented and tests were carried out in order to validate QIDS. The scenario employed to conduct the evaluation tests consisted of the establishment of inter-domain Virtual Private Networks (VPN) using Border Gateway Protocol (BGP) and Multiprotocol Label Switching (MPLS) technologies (BGP/MPLS VPN) to support the provisioning of video streaming services. The results of these tests are presented and discussed in this thesis. These results satisfy the main objectives of QIDS which are to efficiently and dynamically compose and establish inter-domain QoS-aware services and adapt the QoS of these services in a dynamic multi-domain environment.

You cannot depend on your eyes when your imagination is out of focus.

Mark Twain

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

3GPP	3rd Generation Partnership Project
AAA	Authentication, Authorization and Accounting
AcE	Access Element
ADE	Adaptation Decision Engine
AS	Autonomous System
AGAVE	A liGhtweight Approach for Viable End-to-end IP-based QoS Services
API	Application Programming Interface
AQUILA	Adaptive Resource Control for QoS Using an IP-based Layered Architecture
ApE	Application Element
BGP	Border Gateway Protocol
BI	Business Indicator
BL	Business Layer
BPEL	Business Process Execution Language
BSS	Business Support System
CADENUS	Creation and Deployment of End-User Services in Premium IP Networks
CC	Content Consumer
CEI	Customer Entry Interface
CM	Context Manager
DAIDALOS	Designing Advanced network Interfaces for the Delivery and Administration of Location independent, Optimised personal Services
CoE	Connection Element
CORBA	Common Object Request Broker Architecture
CoS	Class of Service
CP	Content Provider

CS	Content Server
CSP	Constraint Satisfaction Problem
CPE	Customer-Premises Equipment
CVV	Common Communication Vehicle
DDQoS	Distributed Dynamic Quality of Service
DiffServ	Differentiated Services
e-QC	Extended QoS Class
EO	Element Owner
ENTHRONE	End-to-End QoS through Integrated Management of Content, Networks and Terminals
EQ-BGP	Enhanced QoS Border Gateway Protocol
ESB	Enterprise Service Bus
EST	Element Specification Template
ETSI	European Telecommunications Standards Institute
EuQoS	End-to-end Quality of Service support over heterogeneous networks
GBF	Global Business Framework
GRE	Generic Routing Encapsulation
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
IntServ	Integrated Services
IOS	Internetwork Operating System
IP	Internet Protocol
IPSec	IP Security Protocol
ISP	Internet Service Provider
IT	Information Technology
ITU	International Telecommunication Union
ITU-T	Telecommunication Standardization Sector
LDAP	Lightweight Directory Access Protocol
l-QC	Local QoS Class
MA	Monitoring Agent

MCOP	k-Multiconstrained Optimal Path
MESCAL	Management of End-to-end Quality of Service Across the Internet at Large
MPLS	Multiprotocol Label Switching
MTOSI	Multi-Technology Operations System Interface
MUSE	Multi Access Service Everywhere
NGN	Next Generation Network
NGN-GSI	NGN Global Standards Initiative
NGOSS	New Generation Operations Systems and Software
NP	Network Provider
NSIS	Next Steps Signalling
OSS	Operational Support System
P2P	Peer-to-Peer
PaaS	Platform as a Service
PAM	Policy Adaptation Manager
PBNM	Policy Based Network Management
PDM	Policy Decision Manager
PDP	Policy Decision Point
PEP	Policy Enforcement Point
QIDS	QoS for Inter-Domain Services
QoE	Quality of Experience
QoE2M	QoE-aware Real-time Multimedia Management
QoS	Quality of Service
QMO	QoS parameter Matching and Optimization
QoS-NLRI	QoS Network Layer Reachability Information
RACF	Resource and Admission Control Functions
RACS	Resource and Admission Control Subsystem
RCL	Resource Control Layer
RM	Reputation Mechanism
RMI	Remote Method Invocation
RSVP	Resource Reservation Protocol
RTP	Real-Time Transport Protocol

SaaS	Software as a Service
SDF	Service Delivery Framework
SDF SMI	SDF Service Management Interface
SEAL	Service Adaptation Logic
SES	Software Enabled Services
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SLO	Service Level Objective
SLS	Service Level Specification
SO	Service Owner
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	Service Oriented Computing
SP	Service Provider
SSS	Service Structuring Stratum
SST	Service Specification Template
TEQUILA	Traffic Engineering for Quality of Service in the Internet at Large Scale
TISPAN	Telecommunications and Internet converged Services and Protocols for Advanced Networking
TMF	TeleManagement Forum
UDDI	Universal Description, Discovery, and Integration
UMTS	Universal Mobile Telecommunications System
VPN	Virtual Private Network
VPSN	Virtual Private Service Network
VRF	Virtual Routing and Forwarding
VSC	Virtual Service Community
VT	Virtual Topology
XML	eXtensible Markup Language
XSD	XML Schema Definition
WSDL	Web Service Definition Language

Chapter 1

Introduction

This thesis addresses the problem of providing communication services with Quality of Service (QoS) guarantees in inter-domain Internet environments. The proposed approach aims to enhance the process of handling QoS issues and facilitate interaction between different providers, so they are able to offer, search, compose, provide and adapt inter-domain QoS-aware services in an automated, dynamic and on-demand fashion. This approach was developed in the context of the QoS for Inter-Domain Services (QIDS) model.

QIDS is a model that permits providers to dynamically and automatically compose, establish and adapt services based on QoS requirements. Moreover, it also supports providers to freely define their service classes. QIDS was deployed in the context of an architecture called Global Business Framework (GBF), which was developed at the Laboratory of Communications and Telematics (LCT) of the University of Coimbra. This architecture is based on a business layer approach that allows providers to offer and manage inter-domain services by publishing, searching and combining service parts to create composite services in a dynamic way.

This chapter is structured as follows: Section 1.1 addresses some of the problems arising from inter-domain service provisioning and discusses the driving-force behind the development of QIDS. Section 1.2 presents its objectives and the contribution it has made, while Section 1.3 outlines the thesis.

1.1 Motivation and Problem Statement

The Internet is the driving-force behind new business perspectives in the world of Information Technology (IT). Its ubiquity and user-friendliness give providers (network providers, content providers, service providers) an opportunity to increase their share of the market by enabling them to reach more customers. Yet, to achieve this goal, they must offer advanced and competitive services that can satisfy the increasing number of customer requirements. This may depend on efficient cooperation between providers, since these new customers may be located outside their domain boundaries. However, delivering services on the Internet that go beyond best-effort quality levels is an intricate process due to the wide diversity of the policies, equipment and business goals of the providers [Siekkinen 07, Rao 08].

Currently, providers have to rely on human-based interactions and manual configuration to be able to offer inter-domain services with QoS guarantees. They have to exchange fax messages and e-mails to define the service requirements and establish contracts. Moreover, the equipment is manually configured to create service paths to meet these requirements. The need for these cumbersome tasks could be avoided if versatile and automated mechanisms were employed to overcome the drawback of heterogeneity.

One problem arising from this heterogeneity is the fact that providers may offer the same service class with different definitions [Jacobs 05]. If a service with QoS guarantees has to cross more than one domain and the providers of these domains have different service class characteristics, there will be difficulties in deciding which service class the traffic should correspond with. One possible solution to this problem would be to adopt some kind of standardization of the service class definitions that the providers should follow. Although this alternative simplifies the interaction, it restrains providers from diversifying their business strategies. Another solution is to deploy a mechanism that automatically maps a service class to another, and thus gives providers the autonomy to form their own service class definitions in accordance with their policies and strategies.

Selecting and combining appropriate services from different providers that must work together to deliver a composite service, is also a challenge in multi-domain environments [Blum 09a]. Today, the customer only requires services from his/her access provider's portfolio, which had previously been contracted from a content provider. In the multi-provider environment, there should be no limitations of this kind. A provider should be able to offer services despite the existence of previous arrangements with other providers, and thus increase the range of its available services. In view of this, dynamic mechanisms should be designed to select and combine services from different providers automatically and thus meet the demands of the customers.

These dynamic mechanisms should be employed to analyze how to connect the individual service parts, by taking into account their parameters, and thus create a suitable end-to-end service path. However, the parameters used to select and compose this path should not be restricted to well-known performance parameters (e.g. bandwidth or delay) [Ibarrola 10]. The increasing diversity of services with different functionalities also compels providers to take service-specific parameters into account to fully satisfy customer demand. Moreover, business factors should also be an important part of the composition process, since in this multi-domain environment, different business perspectives and goals have to be taken into consideration [Bertin 09, Burgstahler 03, Papazoglou 07b, Pouyllau 09].

Even though the providers may represent their service class definitions in an interchangeable way and can be dynamically selected to compose a service path, they have to guarantee that the service will be delivered accordingly. In inter-domain environments, a provider can only guarantee that the service complies with the agreed QoS when the traffic crosses its own network. There is no way to ensure that the QoS level remains the same outside its domain. For this reason, it is necessary to establish contracts that state the terms of the service and penalties that will be imposed in case of violations, and thus formalize the responsibilities of each party of the contract. In addition, corrective measures should be taken if any violation of a contract is detected. These can vary from simple fines to complex reconfigurations of the service. However, in most cases, it is preferable to adapt the services than choose other alternatives [Lin 09]. On the one hand, a service reconfiguration is time consuming and may require a high resource restructuring of providers. On the other hand, if a provider pays fines on a regular basis, it might drive away customers, since it suggests that the provider often fails to fulfill its technical commitments. For this reason, adaptation mechanisms are also important in these dynamic environments.

To address the problems raised above, this thesis proposes a QoS model called QoS for Inter-Domain Services (QIDS) [Matos 10a], which employs mechanisms to provide and manage QoS-aware services in inter-domain scenarios. QIDS is an integrated, on-demand and automatic solution that allows providers to do the following: handle customer requests for services with QoS requirements; select and compose services according to business requirements and service class characteristics defined by each provider; establish contracts between the parties to guarantee the QoS service requirements; create configuration scripts and enforce them in the underlying equipment; and adapt the QoS of the service in the event of requests or contract violations. QIDS is being investigated in the context of the Global Business Framework (GBF) [Matos 08], which is an architecture that allows providers to interact with each other through a Business Layer (BL) by means of Web Services technology.

GBF is an architecture inspired by the IPsphere [IPsphere 07b] and the Software Enabled Services (SES) Management Solution (previously known as Service Delivery Framework (SDF)) [TMF 09]. It is based on a BL, which acts as a communication channel between business entities. By using the BL, QIDS is able to take advantage of a collaborative infrastructure that clearly defines the business roles of the parties involved in the service provisioning process. GBF was developed in collaboration with Alexandre Veloso de Matos, who is a PhD candidate at the Department of Informatics Engineering at the University of Coimbra.

Unlike IPsphere and SES Management Solution, which focus on the messages exchanged in the higher layers and lack details in important areas (such as the combination of service parts to create a complete inter-domain service and adaptation mechanisms), the integration of GBF with QIDS aims to create a coherent infrastructure, which relies on a communication channel, the information exchanged on this channel, and the composition and adaptation processes performed by the providers to allow them to offer, combine and manage their end-to-end services in an automatic, on-demand and dynamic fashion.

1.2 Objectives and Contributions

The main goal of this work is to contribute to the development of more sophisticated techniques in the Internet, to support providers in offering and providing advanced QoS-aware services for customers beyond their domains. This means that the service provisioning process aims not only to fulfill the technical requirements of customers, but also to satisfy the business expectations of providers. A QoS model called QoS for

Inter-Domain Services (QIDS) has been developed to achieve this goal. QIDS allows providers to do the following: define their service classes in a flexible manner, compose and adapt inter-domain QoS-aware services automatically and dynamically, establish contracts to guarantee the QoS levels, and create and enforce configuration scripts in equipment. The contributions made by this thesis can be summarized as follows:

A business layer-based framework to support inter-domain service provisioning

Global Business Framework (GBF) is a business layer-based framework that allows providers to publish, search and compose services which can be offered to their customers. It uses the Business Layer (BL) as a communication channel through which providers can exchange service templates to make business agreements. These service templates comprise information about the services that are being offered. The GBF was developed in collaboration with Alexandre Veloso de Matos, who is a PhD candidate at the Department of Informatics Engineering at the University of Coimbra.

Flexible representation of QoS class definitions for inter-domain services

QIDS allows providers to represent the QoS information of their services in a simple and flexible way [Matos 10b, Matos 11a], so they can easily change their service definitions, by adding, removing or updating service parameters and service parameter values. A provider can distinguish its services from those of competitors by using internal policies to specify its service class definitions freely. As a result, this provider can employ business strategies which attract more customers (e.g. offering a basic service class which can meet higher requirements), and thus increase its market share.

Automatic and dynamic mechanism to compose inter-domain QoS-aware services

The composition mechanism of QIDS [Matos 10b, Matos 11a] provides a dynamic and automated means of solving the problem of selecting and composing services to meet QoS requirements in inter-domain environments. Based on local service policies, QIDS can compare the service requirements with the QoS parameter of service elements (smaller services) from third-party providers with a view to selecting the ones that are most suitable. After this, QIDS builds a virtual topology (graph) representing the paths created by the selected service elements and chooses one of the paths to provide the service requested by the customer (composite service).

By using this approach to compare the QoS parameters individually, rather than the service classes, the composition scheme allows the service elements which have equivalent QoS characteristics to be combined in a way that is independent from the service classes. Furthermore, the probability of service rejections and QoS violations is reduced, since the service elements are searched for on a per-service request basis and providers can update their service element offers in accordance with their network conditions. Another important advantage of this composition process is that it is not restricted to performance parameters alone, but also takes into account service-specific parameters and business parameters to select the service elements and compose the service path. This means that, it is possible to fulfill customer technical requirements and, at the same time, satisfy the business expectations of the providers.

Automatic and dynamic mechanism to establish and adapt inter-domain QoS-aware services

By defining interfaces to interact with Operational Support System (OSS) functionalities, QIDS can request a reservation in advance of the providers' resources. Moreover, it also requests an automatic resource configuration by sending high-level QoS requirements that are translated into specific configuration instructions, in accordance with the provider equipment and QoS strategies. This translation process allows providers to maintain their own QoS model in their networks, regardless of what QoS model is deployed in the neighbor domains.

A crucial aspect of inter-domain interactions is the question of how a provider can guarantee the QoS level of its service if it depends on services from other providers. The answer is to establish contracts between the parties, so they can formalize their relationships. QIDS supports the automatic creation and establishment of Service Level Agreement (SLA), which are used to ensure that the services are delivered accordingly.

QIDS also employs a mechanism to support QoS adaptation in inter-domain services [Matos 11b, Matos 11a]. This mechanism allows providers to renegotiate QoS parameters in the event of QoS violations (or if an adaptation is requested by one of the parties involved in the service provisioning process), and thus they are able to define new QoS thresholds to guarantee service continuity. By using monitoring messages received from third party monitoring agents, QIDS can take corrective measures in the event of contract violations. If a provider fails to fulfill the agreed QoS requirements, QIDS can adapt the service requirements to prevent the customer from experiencing a service termination. Moreover, adaptation processes can also be triggered by customer requests. For instance, if a customer can no longer afford a premium video service, he/she may ask for a downgrade of the service.

Prototyping and validation of the QoS model for inter-domain services

A prototype with the QIDS mechanisms was implemented to validate the proposal. This prototype includes the mechanisms needed to compose and adapt inter-domain QoS-aware services. It also includes interfaces to interact with both the BL from GBF and the OSS/policy layer functionalities. By interacting with GBF, QIDS uses the service templates exchanged in the BL to collect information, which can be used to compose the inter-domain service. By interacting with the OSS/policy layer, QIDS can request for reservation in advance and configuration of resources in a dynamic and automatic manner.

1.3 Thesis Outline

The remainder of the thesis is divided in five chapters structured as follows:

Chapter 2 outlines the problems related to inter-domain QoS-aware service provisioning and discusses relevant related studies in this field. The objectives of this chapter are to show the need to overcome the challenges posed by the inter-domain environment and examine some of the different approaches that have been adopted to tackle these problems.

Chapter 3 presents the GBF along with the BL and service templates. It describes how providers use the BL to interact with each other in order to provide an inter-domain service and how the services are described in the service templates.

Chapter 4 describes the QIDS model. It outlines how the service classes are defined and the policies used by QIDS. It also explains the processes of composing, establishing and adapting inter-domain QoS-aware services.

Chapter 5 presents some details about the implementation of the QIDS prototype. This chapter also discusses the results of evaluation tests performed to validate QIDS. The scenario used to undertake the tests was the establishment of inter-domain BGP/MPLS VPNs to support video streaming services.

Finally, Chapter 6 concludes the thesis by summarizing its main contributions and pointing out directions for further work.

Chapter 2

Issues in Inter-Domain QoS-Aware Service Provisioning

Today, it is natural to consider the Internet as the de facto infrastructure for providing telecommunication services to users. Due to its ubiquity, Internet Service Providers (ISPs) are able to reach users that are beyond their domain boundaries, and thus increase their market share. As a result, it can be seen as the driving-force behind telecommunication service convergence. Moreover, network convergence, which is one of the objectives of the Future Internet [Gutierrez 08], also leads to service convergence scenarios where ISPs and content providers cooperate to provide end-to-end services. However, the portfolio of services resulting from this cooperation mainly consists of simple applications. The main reason for this is the intrinsic heterogeneity of the Internet. In an end-to-end service provisioning scenario on the Internet, providers will almost certainly have different equipment, their own policies, and distinct business goals. To deploy advanced services in this kind of scenario, it is necessary to devise mechanisms that allow providers to interact automatically and provide the requested services in a dynamic way. For this reason, several studies have been conducted with the aim of creating an infrastructure that improves inter-domain interaction.

This chapter addresses the question of providing inter-domain QoS-aware services and presents existing approaches that tackle this problem. Section 2.1 outlines the inter-domain QoS problem. In Section 2.2, the most relevant recommendations from standardization groups for inter-domain service provisioning are presented. Section 2.3 discusses relevant studies that examine inter-domain scenarios from a business perspective. Section 2.4 is concerned with related work on the provisioning and management of QoS. In Subsection 2.4.1, there is a discussion of related work on inter-domain service composition, while Subsection 2.4.2 examines relevant work concerned with the adaptation of QoS in inter-domain services. Finally, Section 2.5 summarizes this chapter.

2.1 The Problem of the Inter-Domain QoS

Before end-to-end services can be provided over the Internet, several providers from different domains have to connect their networks to establish links between customers or between customers and service providers. However, due to the intrinsic heterogeneity of the Internet, sophisticated applications are still not being disseminated, despite all the progress attained in network and service management over the years. This situation gets worse when these services need QoS guarantees. In the Internet, providers almost certainly have different kinds of equipment and distinct QoS models that conform to their own policies. This diversity hampers the creation of an infrastructure in which providers can offer services beyond best-effort. In the approach adopted in current service provisioning, when a customer requires a service, such as video streaming, he must request and pay a content provider for the video and use his Internet Service Provider (ISP) as an access provider (paying for this as well). In this scheme, the content provider can only guarantee that the video will be transmitted with the desired requirements (e.g. high definition), but cannot guarantee that the video will be delivered to the customer as requested. The reason for this drawback is that neither the ISP, nor the content provider, has mutual guarantees about their respective service provisioning capabilities (at least not in real time).

Figure 2.1 shows a generic inter-domain connection comprising four providers with three different QoS models and equipment from three different vendors. In this figure, there are two QoS models originally designed for wired networks and one for wireless networks. The QoS models originally designed for wired networks are the Differentiated Services (DiffServ) [Blake 98] and the Integrated Services (IntServ) [Braden 94] models, which have both been standardized by the Internet Engineering Task Force (IETF). IntServ is a per-flow based QoS model that uses Resource Reservation Protocol (RSVP) [Braden 97] to reserve end-to-end paths that can guarantee service quality. DiffServ is a per-class based QoS model, in which traffic packets are mapped into classes of

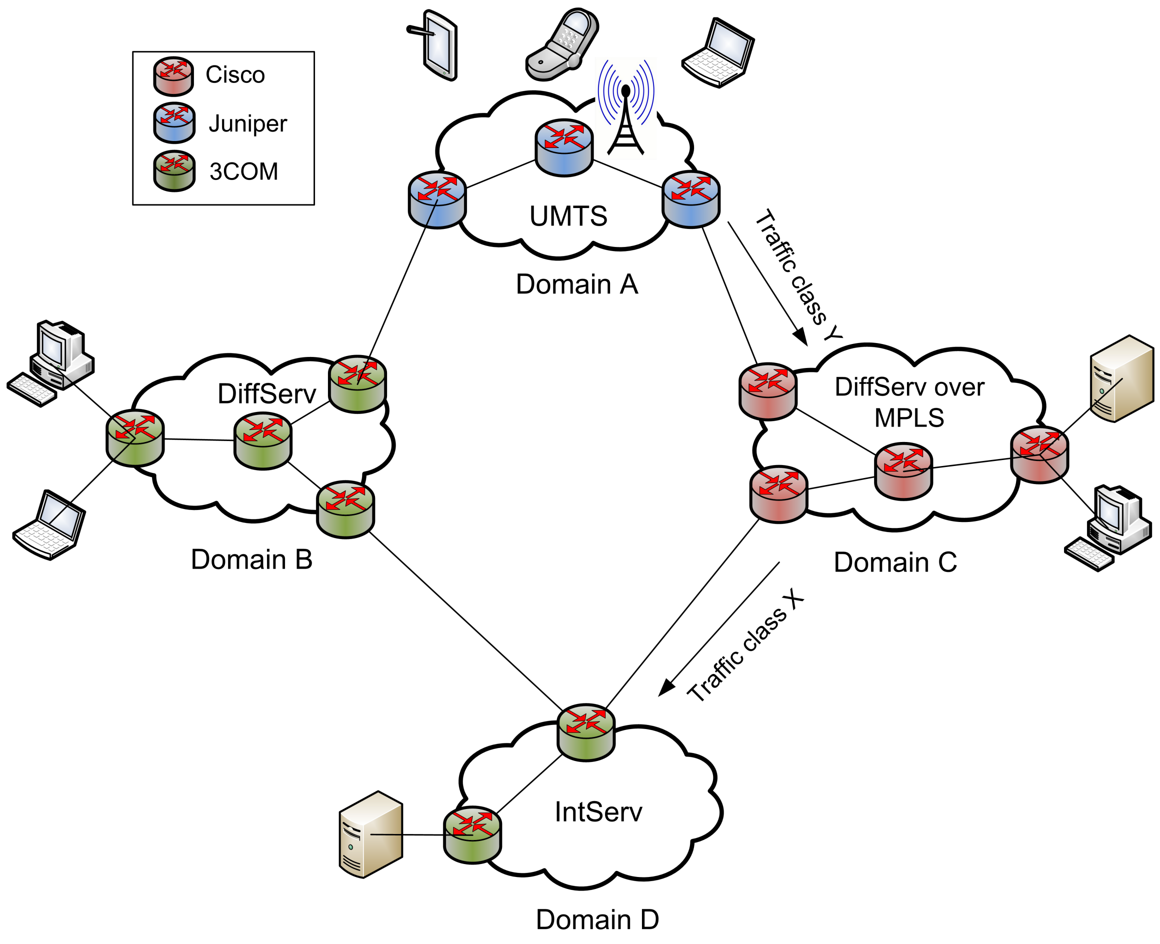


Figure 2.1: Internet heterogeneity example

services. These classes define how the equipment deals with the incoming packets (e.g. forwarding the packets to a route with more bandwidth). One of the domains that uses the DiffServ QoS model employs the DiffServ over Multiprotocol Label Switching (MPLS) strategy, in which the MPLS is responsible for performing traffic engineering by forwarding the traffic in accordance with labels assigned to each packet. With regard to the wireless environment, the Universal Mobile Telecommunications System (UMTS) QoS model [UMTS 09] conceived by the 3rd Generation Partnership Project (3GPP), defines four QoS classes that are mainly differentiated by their delay sensitivity. The model uses a mechanism (Bearer Service) to transfer information between access points and lower layers. This mechanism is deployed from the source to the destination to support the contracted QoS class by using services from these lower layers.

In the example of 2.1, a traffic flow is mapped onto a QoS level Y in the Domain A, which employs the UMTS QoS model. When this traffic flow exits from the Domain A and moves towards Domain C (which employs the DiffServ over MPLS strategy), it is not a straightforward matter to determine which QoS level it must be mapped onto, since the domains employ different policies to deal with the traffic characteristics. The same problem occurs when the traffic flow exits Domain C and moves towards Domain D.

Another obstacle that arises from the same problem of heterogeneity is that there is diversity in the business standpoints of the providers. Although the providers may have the technical competence to offer high quality end-to-end services, there is still a lack of business models that allow providers to negotiate contracts for service provisioning in an automatic and on-demand fashion. As a result, human intervention is often required, which leads to an increase in the time needed for service establishment and makes this process more error prone.

A better approach to the issue of service provisioning in inter-domain environments would be to create a scenario where the customer requests and pays only one provider for the video. This provider would be responsible for finding and assembling several other services (published/offered by other providers) so that it could compose the video provisioning path and be in a position to deliver it to the customer with the requested requirements. It would also be in charge of the process of drawing up contracts between the providers that own the services that are part of the whole video service and paying for them. Furthermore, adaptation mechanisms would also be put in place to ensure that the customer receives the service in accordance with the requirements of the contract. Otherwise, penalties would be imposed.

This multi-domain scenario also raises a new concern, which is the handling of business issues. Due to the trend of separating the service from the transport-related functions, providers have found themselves in a position where they can offer innovative services in a more dynamic way and users have a wider selection of service options. This has led providers to seek better, automatic and dynamic means of forming business relationships. This is important since it concerns multi-domain environments involving the business perspectives of many different providers. Furthermore, it is necessary to separate business-related from non-business functions to ensure the independence of the business process from the service and the network provisioning processes [Papazoglou 07b].

The scenario outlined above can only be deployed by devising robust mechanisms for the following: to mediate interactions between the providers, to exchange and translate information, to resolve conflicts arising from diverging policies, and to establish contracts that can allow end-to-end paths to be configured in inter-domain environments. As a way of offering value-added services, these end-to-end paths should be assembled based on the customer service requirements, the quality of service (QoS) capabilities of each provider along the path, the characteristics of the service content (if possible), and the business expectations of each provider. Moreover, these mechanisms should have the following pre-requisites [Papazoglou 07a]:

- On-demand: The services should be provisioned at the time a customer request takes place;
- Automatic: Reducing the need for human intervention during the service configuration process; and
- Dynamic: An ability to carry out dynamic service composition and adaptation due to changes in policies, new contexts and external factors.

Several solutions have been proposed to tackle some of the difficulties mentioned earlier and the majority of them are based on the Next Generation Network (NGN) [ITU-T 05b] paradigm, which recommends a clear separation between service-related functions and transport and network-related functions. The next Section examines what has been done so far from the perspective of NGN, with regard to service provisioning for inter-domain environments.

2.2 Inter-Domain Service Provisioning

NGN emerged as an innovatory means of changing the way that providers interact with each other and offer services to customers. Unlike the traditional vertical service provisioning structure, where the customer is only restricted to his provider's portfolio, NGN offers a new market perspective that is based on its cornerstone - the separation of transport from service-related functions. This new paradigm allows service providers to cooperate in seeking to offer services beyond their domains, without having to be concerned about the network infrastructure. In addition, the system enables providers to compose and offer their services as new value-added services. This creates a new market scenario, where providers can charge for these value-added services instead of charging for the network infrastructure. Some standardization groups have already made recommendations to achieve this objective.

The Software Enabled Services (SES) Management Solution (previously known as SDF) [TMF 09], proposed by the TeleManagement Forum (TMF), is a management structure that enables the delivery of next generation services. It defines a set of standards to manage the service life-cycle (concept, design, deploy, operate and retire), regardless of the technologies used to implement the services or to build the network infrastructure. Any service that supports management capabilities so that it can be managed through SDF is known as an SDF Service. Each SDF Service has a SDF Service Management Interface (SDF SMI), which refers to a set of exposed standardized operations that enables the management of the SDF Service and its dependencies. The SES Management Solution also introduces the concept of SDF Service Lifecycle Metadata, which provides a schema that depicts the SDF Service dependencies and properties. This schema is an abstraction of essential information used by stakeholders, that cooperate across multiple domains, to manage the SDF Services throughout their life-cycles. Some of the information included in the SDF Service Lifecycle Metadata consists of the following: properties and actions of SDF Services, cross-domain dependencies, and the state of the SDF Service. Although the SES Management Solution is an important work to create a cross-domain environment to provide services, it focuses on providing abstract descriptions of the management operations of the framework. For instance, it does not specify how two or more services can be combined to create a SDF Service. Regarding the delivery of QoS-aware services, it also does not detail how to deal with different classes of services, which is an expected situation, since it operates in a multi-domain environment.

The 3GPP standardization body set up an architectural framework, called IP Multimedia Subsystem (IMS) [IMS 07], which was designed to deliver multimedia and voice services for mobile users through UMTS technology. This system allows the creation and deployment of services by the operators, since it offers common functions by means of service enablers. Some examples of these functions are billing, resource and admission control, and application management. IMS was one of the first initiatives to deliver services in multiple domains, although it was designed for the mobile network.

In the context of fixed networks, the Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN) group of the European Telecommunications Standards Institute (ETSI) set out the TISPAN NGN specifications [ETSI 08, Kovacikova 07]. It is a subsystem-based architecture, where new subsystems are added over time to address the needs of new service classes and meet new demands. It uses the IMS core subsystem to handle applications based on the Session Initiation Protocol (SIP) [Rosenberg 02] and additional subsystems to handle non SIP-based applications. It provides an open architecture where application, control, and transport functions are provided by their respective layers.

Like ETSI, the Telecommunication Standardization Sector (ITU-T) of the International Telecommunication Union (ITU) made its own NGN architecture recommendations that are currently being maintained by the NGN Global Standards Initiative (NGN-GSI) [ITU-T 05a]. The objective of this architecture is to support the provisioning of content delivery and multimedia services and it is composed of a Service Stratum and a Transport Stratum. The Service Stratum provides several functions to facilitate and enhance service deployment and provisioning, such as user authentication, service discovery and session negotiation. The Transport Stratum provides functions to guarantee IP connectivity between the service endpoints, such as resource and admission control functions.

The TMF proposed a set of reference models, called New Generation Operations Systems and Software (NGOSS) [TMF 02, Goestl 06, Callen 06], to implement a new generation of Operational Support System (OSS) and Business Support System (BSS). Its objective was to enable providers to develop and deploy new OSS/BSS solutions, so they can be given support in their service management operations.

The Multi-Technology Operations System Interface (MTOSI) standard, which is supported by TMF, is an open interface used to achieve interoperability between providers through their OSSs [Caruso 06]. It fully adopts the Service Oriented Architecture (SOA) philosophy and proposes the separation between business logic and transport functions and supports the specifications dictated by NGOSS. The MTOSI Implementation Lab aims to produce an MTOSI reference implementation to simulate and test real scenarios. It uses a Common Communication Vehicle (CVV), based on the concept of Enterprise Service Bus (ESB) [Schmidt 05]. ESB is an infrastructure used to interconnect services, where the services are defined by meta-data, which are published in the ESB. Mediators are then used to forward service requests from requestors to the appropriate providers, by using the meta-data information as guidelines.

Although these recommendations are well-specified, and widely accepted by the research community, together with the fact that some of them already have a reference implementation (e.g. MTOSI Implementation Lab [Caruso 06]), they still need to incorporate other features that are needed to leverage the service provisioning process. These features include the following: the mediation of business relationships between the involved parties, negotiation and delivery of end-to-end QoS in inter-domain scenarios in a dynamic and on-demand fashion, automatic service composition with the aid of smaller services, and the QoS adaptation services. Furthermore, some of these recommendations (e.g. NGOSS, MTOSI) are strategies for OSS interconnection, rather than broader provider integration, thus neglecting, for instance, publication and the offer of integrated services. MTOSI also focuses on pre-scheduled customers and providers, rather than dynamic, on-demand service provisioning.

2.3 The Business Perspective

A number of requirements must be met to achieve the aims of the NGN such as service convergence, end-to-end QoS and security, just to name a few [ITU-T 04]. All the recommendations mentioned above have set approaches for meeting these requirements. Some of them cover entire modules and describe how these modules interact with each other [ETSI 08, ITU-T 05a, Caruso 06]. Although these approaches offer a plausible means of deploying NGN architectures, they are still unable to guarantee that these requirements can be met at a business level. The following studies have offered suggestions for dealing with business issues.

IPsphere Forum, which joined TM Forum in 2008 in the SDF project, outlined a framework [IPsphere 07a,IPsphere 07b,IPsphere 08] for creating and delivering services that was based on NGN principles and that allowed providers to interact through a business layer, called Service Structuring Stratum (SSS), as well as to locate, contract, initiate, operate, and terminate a service. This framework encompasses automatic service composition, and negotiation, which leads to a flexible service provisioning infrastructure, while ensuring a coherent and viable business model. According to the IPsphere framework (Figure 2.2), services offered to customers may be composed of multiple smaller services (elements) provided by different partners (the Element Owners, EO). The Administrative Owner (AO) combines these elements to create the requested service. The SSS layer provides an appropriate environment where business partners publish their offers and where AOs locate and contract elements. Customer

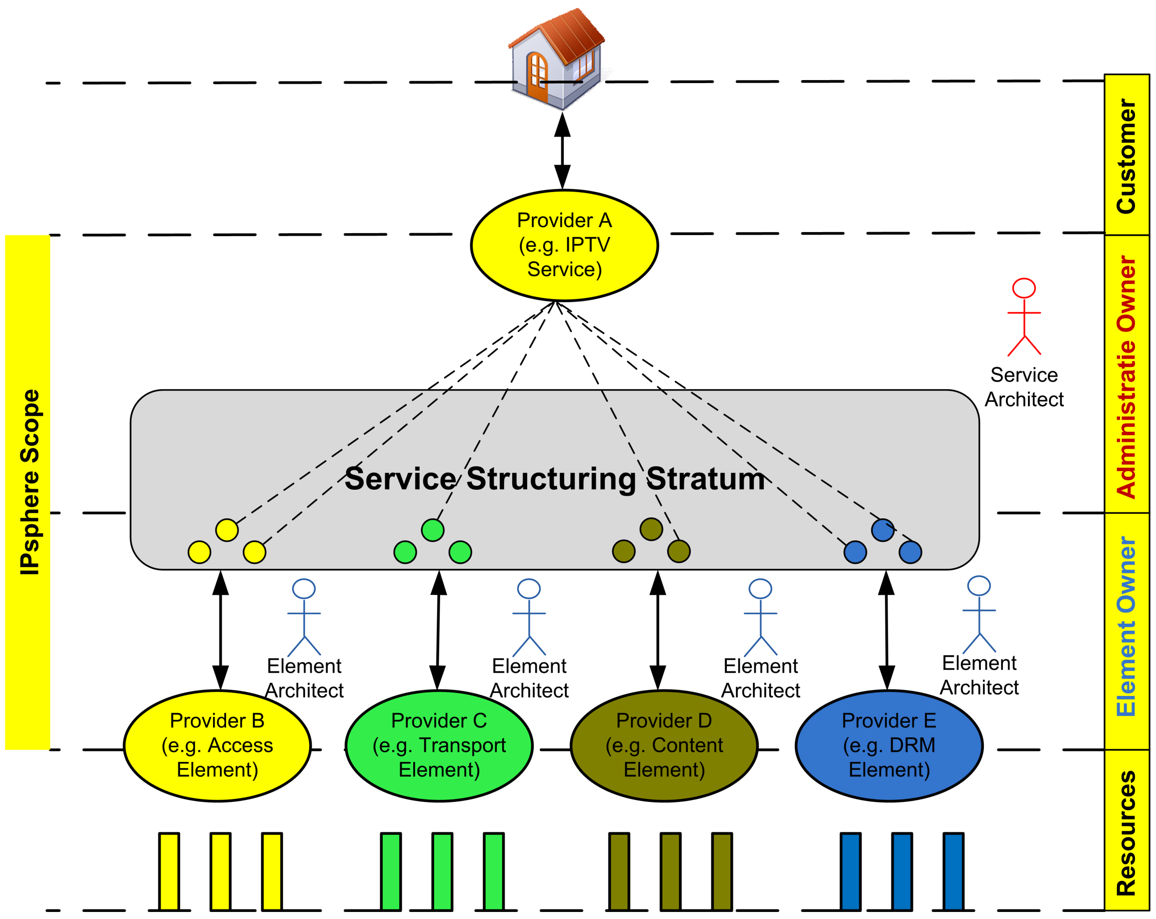

Customer
Administratie Owner
Element Owner
Resources
IPsphere Scope
Provider A (e.g. IPTV Service)
Service Architect
Service Structuring Stratum
Element Architect
Element Architect
Element Architect
Element Architect
Provider B (e.g. Access Element)
Provider C (e.g. Transport Element)
Provider D (e.g. Content Element)
Provider E (e.g. DRM Element)

Figure 2.2: IPsphere framework context diagram [IPsphere 07b]

access to service offers is obtained through the OSS of the provider or, alternatively, by third-party portals (for instance a movie rental service).

Although the IPsphere proposal makes a significant contribution to the process of inter-domain service provisioning, it is mainly concerned with the operations (message exchanges) that occur in the SSS. The processes in which its components are involved are still very generic and lack details. For instance, the framework does not specify how the elements are chosen and combined to create a service that meets the appropriate requirements. Moreover, although IPsphere states that the framework can also deliver QoS-aware services, it does not specify how to deal with different classes of services. Finally, the IPsphere framework does not provide details on how to handle adaptations in the event of a service failing to comply with the agreed QoS requirements.

A framework for achieving business-driven management in networks that employ MPLS [Rosen 01] is outlined by Loyola *et al.* [Rubio-Loyola 10]. In this framework, the authors set out mapping functions that use Business Indicators (BIs) (e.g. service satisfaction, profit and loss) to help formulate service management policies. These policies are then employed to generate network configurations that are in line with high-level business objectives: a) controlling the subscription volume of service requests; and b) controlling the service quality. Although this framework provides a means of applying business factors to manage the QoS of an offered service, it does not address the question of inter-domain environments.

Royon and Frénot propose a business model for home gateways called multiservice model [Royon 07]. At present, home gateways are tied to a single service provider. The authors state that, in the near future, these gateways will have to cater for different business actors and services (connectivity provisioning and service provisioning). Thus, they propose a middleware running in these gateways, so that they can launch "virtual" gateways per service provider. Each virtual gateway has a management agent who is responsible for the functions of each service provider. In addition, there is an end-to-end management infrastructure that uses the information from the virtual gateways and from the multiservice constraints. Thus, a home gateway can handle several services from different business actors. Although this study addresses the question of business requirements from different providers, it does not take into account how these requirements can affect a dynamic composition of services.

Bertin *et al.* [Bertin 07] presented a business viewpoint to define the relationships between the end-user and the service provider. This study defines the business processes that represent the operations that are involved in service provisioning from the end-user authentication to the management of the service session. The business viewpoint can be used by providers to ensure that, from the perspective of the end-user, there is coherence between the processes of NGN services. However, this study does not address the issue of how business information can be used to create and provide services.

2.4 End-to-End Service Provisioning with QoS

Mingozzi *et al.* [Mingozzi 09] describe the framework proposed by the End-to-end Quality of Service support over heterogeneous networks (EuQoS) European Project [Dugeon 07, Masip-Bruin 07], which is designed to ensure end-to-end QoS in heterogeneous multi-domain networks. This framework assumes that users can be attached to different access networks and relies on the Enhanced QoS Border Gateway Protocol (EQ-BGP) which is also recommended in the EuQoS context to advertise QoS information [Beben 06]. In the EuQoS project, the domains employ common class-of-service (CoS) definitions to allow traffic matching between domains. The EQ-BGP compliant routers advertise information about the QoS that is supported by a path to a specific destination. The domains along the path employ equivalent CoSs. The QoS path information is encompassed in the QoS Network Layer Reachability Information (QoS-NLRI) attribute on the EQ-BGP and a CoS-aware decision algorithm chooses the best path based on the information advertised in the QoS-NLRI attribute.

The aim of the Multi Access Service Everywhere (MUSE) project was to develop a multi-service access network [Rajan 06]. It planned a QoS architecture to allow users to request broadband services. This QoS architecture is composed of a data plane and a control plane. The data plane defines four DiffServ-based classes in which the user service request may belong to. The control plane is responsible for admission control, and monitoring the network and the resource availability.

Other projects have been implemented to achieve the same objective, which is to provide IP Premium Services over the Internet [Giordano 03]. These include Traffic Engineering for Quality of Service in the Internet at Large Scale (TEQUILA) [Mykoniati 03], the Adaptive Resource Control for QoS Using an IP-based Layered Architecture (AQUILA) [Engel 03] and the Creation and Deployment of End-User Services in Premium IP Networks (CADENUS) [Cortese 03] projects. Although these projects share the same objective, they focus on different aspects of the service provisioning. TEQUILA proposed an approach to perform service request negotiation between the provider and customer in an intra-domain environment. This negotiation originates an IP connectivity, which is described as a Service Level Specification (SLS) that is used by a set of traffic engineering tools to acquire quantitative end-to-end QoS in a DiffServ-based IP network. The AQUILA project developed a Resource Control Layer (RCL) over a DiffServ layer to control and monitor the resources. Through RCL, AQUILA offers the customer network services with different QoS characteristics and based on pre-defined classes. In its turn, the CADENUNS project established a framework composed of functional blocks. These functional blocks comprise operations such as service composition, authentication and mapping of QoS requirements onto network resource. The interactions between the blocks allow the occurrence of service creation, negotiation and provisioning, as well as equipment configuration for effective QoS delivery.

The Management of End-to-end Quality of Service Across the Internet at Large (MESCAL) project [Howarth 05] extended the work of TEQUILA to the inter-domain context. It proposes an architecture to deliver end-to-end QoS across multiple domains using the concepts of Local QoS Class (l-QC) and Extended QoS Class (e-QC) to meet customer requirements. An l-QC expresses the delivering QoS ability of a provider inside its domain, while e-QC expresses the delivering QoS ability of the local domain combined with the l-QC or e-QC of neighbor domain providers. The MESCAL architecture uses a cascaded model to negotiate the SLSs. In other words, providers only negotiate with their neighbor domain providers. Traffic engineering algorithms are responsible for finding appropriate combinations of l-QCs and e-QCs and binding them to create a service path provisioning. Mescal also suggests enhancements to BGP in support of QoS requirements.

The project called A liGhtweight Approach for Viable End-to-end IP-based QoS Services (AGAVE) introduced the Parallel Internet concept which is a union of parallel Network Planes of network providers [Wang 07]. A Network Plane is a logical layer configured to route and forward traffic related to services with similar QoS requirements. These planes are used by service providers to guarantee the end-to-end QoS required by the customer. They can be created to support standard DiffServ-based QoS classes or to support the specific requirements of service providers. AGAVE also uses BGP extension to advertise the QoS characteristics of a Network Plane beyond a single domain.

Although these projects discussed above are approaches to the problem of providing QoS services, they include some drawbacks that have to be overcome. TEQUILA only focuses on QoS-aware services for intra-domain scenarios. Some of the projects (EuQoS, MESCAL and AGAVE) rely on BGP enhancements to advertise QoS information. This may compromise routing scalability and stability on the Internet, since BGP was formerly only implemented to advertise routes. EuQoS and AQUILA define service classes that are used to map services between domains. This approach imposes constraints on the providers' business strategies, since they have to adjust their service classes according to the standards laid down by the projects. Moreover, most of the projects do not deal with dynamic composition and adaptation of services.

2.4.1 QoS-based Service Composition

In an inter-domain provisioning scenario, where the requested service may be a combination of several smaller ones, it is of crucial importance to make a correct choice of these smaller services and thus be able to compose an end-to-end service path that satisfies customer requirements. The QoS parameters of these smaller services must be compared with each other in order to create an appropriate service that meets these requirements. Moreover, given the fact that providers wish to offer better value-added services, it is advisable to devise mechanisms that take into account QoS performance, service-specific and business parameters so that they can be composed in a way that will enhance the service quality. However, it is not only important to compose a service correctly, but also to perform this process in an automated, dynamic and on-demand fashion. With the current status of technology, customers do not want to wait long periods to access a service, and this makes human-based composition and negotiation obsolete processes. In this section, there is a discussion of service and path composition.

Xiao and Boutaba [Xiao 05] set out a framework to compose end-to-end autonomic services. This framework establishes end-to-end communication paths by abstracting the different administrative domain connections in a graph. The authors then model the composition problem as the k-Multiconstrained Optimal Path (MCOP), using the domains as nodes, the connections between domains as edges and the QoS classes provided by each domain as the weights of the edges. However, this scheme is constrained by the fact that it does not deal with service-specific parameters to create the composite paths.

Gu and Nahrstedt employ a distributed framework to perform service composition called SpiderNet [Gu 06]. This framework uses a service overlay consisting of several nodes on top of an existing network infrastructure. It performs a hop-by-hop searching operation to find nodes that can meet the requirements of the requested service. In each node, SpiderNet checks the statistical QoS conditions, resource availability and inter-service dependencies so that it can create the best service composition. One drawback of SpiderNet is that it performs a resource allocation as soon it finds a suitable node. This may cause unnecessary overhead, since this node may not be selected to be part of the composite service.

In the context of the Software Services and Systems Network (S-Cube) project, a service composition approach is outlined by Rosenberg *et al.* [Rosenberg 09].

The S-Cube project aims to tackle Service Oriented Computing (SOC) research challenges, which include the composition and coordination of services, business process management, and adapting and monitoring of services, among others. In this scheme, the composition process is divided into five steps: a) Resolving the problem of meeting the feature (service operation) requirements of the desired composite service i.e., this involves searching all the service candidates that support a feature and the associated constraints; b) In this step, which takes place in parallel with the first one, each service candidate invocation is analyzed in a search for data dependencies. When all the dependencies have been determined, a service composition structure is generated; c) All the service candidates are aggregated in a Constraint Satisfaction Problem (CSP), which also aggregates the QoS constraints and can have multiple solutions, depending on the set of service candidates that satisfy the constraints of the composite service; d) In this step, the executable composite service is generated; e) Finally, in the last step, the composite service is deployed, making it available to the user. One problem with this scheme is that it does not specify how the QoS constraints should be translated into actual configurations that are used to support the composite service requirements.

Blum *et al.* [Blum 09b] outline an infrastructure based on SOA principles that enables dynamic service composition in multiple network technologies. This infrastructure uses service enablers to abstract network protocols from the service requestors. Only Web Service APIs are made visible by the service enablers, which means that specific network details are hidden. A component called Service Discovery decomposes a service request into small services, finds the corresponding service enablers of each one and transfers the decomposed service request to another component, called Service Composition. This component then creates the service workflow by using a Business Process Execution Language (BPEL)-like engine. BPEL uses XML documents to define the business rules that can be applied to a process. Thus, the service workflow defines how the Web Services of each small service must interact to provide the requested service. However, this approach seems to compose the service purely based on the functionalities of each small services, rather than checking their quality (QoS parameters) as well.

Giladi and Menachi [Giladi 09] provide a service layer that crosses providers' domains and acts as a bridge connecting the OSS of each provider. This service layer also defines a data model and a signalling protocol by means of which providers can exchange connectivity service information (delay, loss, availability). The aim of this service layer is to allow providers to calculate and create inter-domain paths which can support a service. However, this study is a solution for OSS interconnection and is only concerned with connectivity services.

The study carried out by Oh *et al.* [Oh 08] employs a tree-based algorithm to select services to perform service composition, while taking QoS criteria into account. To illustrate their proposal, the authors relied on some QoS criteria (e.g. performance, cost, reliability, etc.). Every service candidate has a value corresponding to each criterion. These values are normalized and a function is applied based on the normalization values and weights defined by the customer for each criterion. The purpose of this function is to calculate the service scores that will be used during the service selection. When all the scores have been calculated, a sorted binary tree is built with the service candidates using the scores as node values. In the end, the algorithm selects the right-most node since it has the greatest value, and hence possesses the most appropriate criteria for QoS composition. The authors also introduce a new service registry that keeps information about the services as well a record of which services are used by which customer. With regard to the customer, service trees are maintained in a cache-based mechanism. At the end of the service provisioning, the customers send the feedback information (e.g. QoS degradation) to the service registry. The registry identifies which customers are using the related service and sends them updated information. This information is used to update their service trees and possibly allow a new selection to be carried out. The problem with this approach is that, as the numbers of service candidates and customers increase, there is also a corresponding increase in the number of service trees for each customer and in the registry, which means that further memory space is required to store these trees, as well as more compute time to perform the composition. The process time for updating the information increases as well.

A Peer-to-Peer (P2P) approach to perform service composition based on QoS requirements was adopted by Lavinal *et al.* [Lavinal 09]. The authors define three generic abstractions (node, link and network) and relate them to the service management context: a node represents any service component; a link represents a virtual connection between two nodes and; a network represents a set of nodes and links that interact to provide an end-to-end service. This dynamically built network is called the Virtual Private Service Network (VPSN). The authors also introduce the concept of Virtual Service Community (VSC) which is a community of service components that possess an equivalent functionality, but with different QoS parameters. They advocate building a VSPN which aggregates the service components selected to provide the global service. To achieve this, it is necessary to select the proper VSCs based on the functional requirements of the service. Afterwards, the appropriate node in each VSC is selected on the basis of the QoS requirements. Although this proposal of a service composition meets the QoS requirements, it does not take into account the service content (service-specific) characteristics. Moreover, it does not provide details of the way it functions in an inter-domain scenario.

Another means of solving the problem of providing QoS paths is proposed by Obreja and Borcoci [Obreja 08]. In this study, logical QoS paths are established between known Content Servers (CSs) and predicted regions of Content Consumers (CCs), thus forming a complete connection graph. This graph crosses IP domains and contains all possible paths. A path is an aggregation of pipes agreed between neighboring domains and represents a specific QoS class. When a CC requests a service, the individual pipes (the path between two neighboring domains in the graph) are negotiated and allocated for the service provisioning in a cascade fashion. The work defined a negotiation protocol composed of a set of messages that are exchanged by modules which are responsible for negotiating the SLS that is needed to establish the logical path. One drawback of this approach is that it is "semi" on-demand, since the logical QoS paths are established before any service request has been made. As the authors themselves note, this results in a greater probability of failure in the negotiation process when compared to other approaches that calculate the logical QoS path when the customer performs the request.

Virtual Topology (VT) advertising is the approach proposed by Freitas *et al.* [Freitas 10]. VT is an abstraction of the domain network status. In this work, when a domain receives a client request, it solicits the VTs (expressed in the form of SLSs) from other domains. A domain can send as many VTs as it wants and each VT can reflect a different view of the same resources in the same domain. This strategy enables domains to advertise different VTs for different purposes. When the domain in charge of handling the customer request has received all the requested VTs, it calculates a QoS path to provide the service. A problem with this approach is that all the domains must advertise their VTs in response to each customer request. Thus in a scenario where there are N domains, each one receiving M customer requests, there would be *(N-1) x M x N* message exchanges (VT advertisements), given the fact that each domain advertises only one VT. Thus, the message overhead can increase considerably.

2.4.2 QoS adaptation

Ahmed and Boutaba propose an approach that involves adapting multimedia traffic in DiffServ-based networks [Ahmed 05]. In DiffServ networks, a resource at the edge of the network (e.g. router) marks incoming traffic with a pre-defined value, so it can be forwarded in accordance with pre-established configurations. In the proposed approach, a bandwidth monitor in the core of the network informs a Policy Decision Point (PDP) about the network conditions. The PDP then uses this information to send appropriate policies to Policy Enforcement Point (PEP), which are located at edge devices. In its turn, the PEP can apply these new policies to the incoming traffic, thus allowing the traffic to be treated in a way that is suited to the current network conditions. Although this proposal uses a dynamic adaptation approach, it does not deal with inter-domain environments. Furthermore, it also does not seek to adapt the traffic to meeting business demands.

In the context of the NetQoS project, Rao *et al.* [Rao 08] set out an architecture to change the QoS level of services based on policy adaptations in heterogeneous networks. The purpose of the architecture is to control the QoS management through the occurrences of events and the policies associated with these events. Its functioning is based on the publish/subscribe principle, where a module called Context Manager (CM) receives subscriptions related to specific events. Whenever an event occurs (e.g. an application launch or a policy violation), CM is informed of the occurrence and notifies the module which subscribed to it. The Policy Decision Manager (PDM) or the Policy Adaptation Manager (PAM) are usually the modules which are notified and take action to fetch QoS policies related to the event. They modules then use the policies to adapt the QoS configuration at the network and transport level entities. However, this proposal only takes account of performance parameters.

A mechanism to adapt the quality of video streams in heterogeneous networks is outlined in the context of the Designing Advanced network Interfaces for the Delivery and Administration of Location independent, Optimised personal Services (DAIDALOS) project [Szwabe 06]. This mechanism employs rate control algorithms and transformation engines to change the quality of the video stream and its format on the fly, in accordance with the network conditions. The End-to-End QoS through Integrated Management of Content, Networks and Terminals (ENTHRONE) project also devised a mechanism to adapt the QoS of audio-visual services to satisfy new requirements [Borcoci 06]. There is a description of the service context usage, which contains information about content adaptation metadata, networks and environmental constraint usage, the capabilities of the terminals, and user preferences. This description is sent to an Adaptation Decision Engine (ADE), which is responsible for selecting appropriate adaptation operations and service parameters. If there are different versions with different quality levels of the same service in a service directory, the ADE chooses the one that best suits the new requirements. Otherwise, the ADE obtains a new version of the adaptation content of the fly. Although these proposals can perform on the fly adaptations, they only take into account QoS performance parameters, such as packet loss and delay. Furthermore, they are also restricted to video streaming and audio services.

An architecture to perform dynamic adaptation at the edge of networks was provided by Cruvinel *et al.* [Cruvinel 08]. This architecture, called Distributed Dynamic Quality of Service (DDQoS), has two main components: DDQoS-Core, which resides in core routers and DDQoS-Edge, which resides in edge routers. The DDQoS-Core monitors the traffic at the core network and informs the DDQoS-Edge if any data losses are perceived in the traffic classes. The DDQoS-Edge then carries out adaptation at the network edge to improve the performance of the priority traffic (static adaptation). By using reports from the DDQoS-Core and learning techniques, the DDQoS-Edge can also perform adaptation before data losses occur in the core (dynamic adaptation). Although this work can adapt traffic flows that are degrading (or that will be degraded), it does not take account of adaptation in inter-domain scenarios.

Mu *et al.* established a framework called Quality of Experience (QoE)-aware Real-time Multimedia Management (QoE2M) to manage the quality level of end-to-end multimedia applications [Mu 09]. QoE2M is designed with well-defined interfaces that can interact with resource allocation controllers to gather information about the network status. Moreover, it also can extract video characteristics from the Real-Time Transport Protocol (RTP) header [Schulzrinne 03]. By using this information, the QoE2M is able to perform adaptations conforming to the network conditions or the user's device capabilities. For instance, it can downgrade a video application if there is not enough network bandwidth available to transmit the video or if the user's device does not support the codec. Although this work adopts a dynamic approach and is aware of QoE characteristics, it does not discuss implementation or validation issues. Moreover, it relies on a proposed protocol that is an extension of the Next Steps Signalling (NSIS) [Hancok 05], thus forcing providers to support this new protocol.

Hadjiantonis *et al.* established a service management framework for wireless environments that can adapt services by means of statistical data [Hadjiantonis 07]. This framework has a component called Service Adaptation Logic (SEAL) that employs information about user preferences and device capabilities to adapt the service in a proactive fashion. For instance, SEAL is able to detect that videos with certain parameter values, such as basic quality level (from user preferences) and DivX codec (from device capabilities) are more often requested by the users. On the basis of this information, SEAL can transcode the videos with these two parameters in advance of the user requests, thus increasing the availability of the service. Although this framework adopts a proactive approach, it only adapts services that have not yet been requested. Running services are not affected by the adaptation process.

Skorin-Kapov *et al.* proposed a solution based on QoS parameter Matching and Optimization (QMO) for enhancing the IP Multimedia Subsystem (IMS) [Skorin-Kapov 07]. This system seeks to give IMS-compliant providers the capability of adapting services, among other functionalities. In guiding the adaptation process, the authors make use of two profiles: (i) generic client profiles, which contain information about user equipment, constraints in access networks , and application preferences; (ii) service profiles, which contain information about different versions of the services that can be used, based on user preferences (the client profile can be considered as an instance of a user preference). By using this information, the QMO is able to change from one service version to another in case an event occurs. For instance, a reduction in the access network bandwidth (specified in the client profile) caused by a modification in the user preference, may trigger a change in the service version. One limitation of this work is that it is only concerned with traditional QoS parameters without taking account of service-specific or business parameters.

2.5 Summary

The evolution and the increasing ubiquity of the Internet have led to a demand for more and more advanced services of a better quality. To achieve this, it is necessary to devise mechanisms that can enhance the interaction between providers, so they can overcome the problems posed by the intrinsic heterogeneity of the Internet (different policies, equipment, QoS models, business goals, etc.). These new mechanisms have to be on-demand, automatic and dynamic.

Several initiatives from standardization groups have been proposed to tackle this problem and most of them follow NGN principles, where there is a clear separation between service and transport functions. Some studies have focused on a business perspective, where importance is attached to business requirements during the providers' interactions. Other studies tackle the problem of providing an infrastructure to deliver QoS in these inter-domain environments, while there are others which concentrate on specific problems, such as service composition and QoS adaptation.

Although several studies have addressed the problem of providing inter-domain QoS, approaches that can effectively integrate three essential functionalities are still needed - the ability to carry out service composition that takes into account performance (connectivity), service-specific and business parameters; the provision of an end-to-end service by reserving resources, establishing contracts and configuring equipment; and a capacity to perform QoS adaptation in inter-domain services. Moreover, it should be stressed that each of these three functionalities must be undertaken in a manner that is automatic, dynamic, and on-demand.

Chapter 3

Global Business Framework (GBF)

The Global Business Framework (GBF) [Matos 08] was developed with the goal of facilitating interaction between providers, so that they can provide inter-domain services in an automatic and on-demand fashion. This framework employs a Service Oriented Architecture (SOA)-based approach, where a composite service is created by combining one or more services (smaller services), which are called service elements. The framework also uses a Business Layer (BL) as a communication channel through which providers can exchange technical and business information about their services. The terms 'service' and 'composite service' represent a service comprising one or more service elements, and will be used interchangeably throughout this thesis. The GBF adopts Next Generation Network (NGN) principles, providing a business layer (BL) where providers can offer (partial) services, which are used by the service owner (the ISP that sells the service to the customer) to assemble the complete inter-domain service. Bellow the BL, GBF relies on solutions to provide the service connectivity, thus separating the service from transport issues.

The GBF was developed in collaboration with Alexandre Veloso de Matos, who is a PhD candidate at the Department of Informatics Engineering, University of Coimbra.

Some of the concepts outlined in this chapter are borrowed from IPsphere. These concepts are as follows: Service element – service components that can be combined to create a service; Service Owner (SO) – a provider of services; Element Owner (EO) – a provider of service elements; Business Layer (BL) – the place where the providers communicate with each other to offer and consume inter-domain services;

Access Element (AcE) and Connection Element (CoE) – types of service elements. Other concepts were derived from IPsphere, such as the templates that specify the definitions of the services, although some parts of the structure of the templates were greatly inspired by the TEQUILA project [TEQUILA 00] and other parts were defined in the context of this work. Finally, the remaining concepts were entirely defined in the context of this work, such as the Application Element (ApE) and the structure of the Service Order, which represents a service request performed by the customer. All these concepts are explained in detail throughout this chapter.

In the same way as IPsphere and Software Enabled Services (SES) Management Solution, GBF aims to provide an infrastructure that supports providers in managing end-to-end services. However, although IPsphere and SES Management Solution are important works concerning inter-domain service management, they are still very generic and do not provide details of crucial aspects of inter-domain management, such as the combination of elements needed to create an end-to-end path. They also fail to specify how to deal with service classes and do not provide details on how to handle service adaptations. On the other hand, the integration of GBF along with the QoS model presented in next chapter creates an infrastructure that aims to fill these gaps.

This chapter is structured as follows: Section 3.1 provides an overview of the GBF functions. Some basic assumptions are outlined in Subsection 3.1.1. Subsection 3.1.2 describes the service templates which are used to represent the providers' offers. Subsection 3.1.3 introduces the BL and the messages exchanged by providers to manage the service life cycle. Finally, Section 3.2 concludes with a summary of the topics discussed in this chapter.

3.1 Overview of GBF

GBF provides an infrastructure in which the providers (network providers, access providers, and service providers) can exchange information that can be used to publish, search, compose and provide inter-domain services. An SOA-based approach [OASIS 06] is adopted by assembling service elements to create end-to-end services. It also employs the publish/find paradigm [Atkinson 07], in which a service provider publishes services in a service directory and this in turn can be used by a service requestor to search for services. In the case of GBF, providers can search for service elements to create a composite service that fulfills a customer request. The GBF allows the processes of searching, composing and providing inter-domain services to be carried out in an automatic and on-demand fashion.

Figure 3.1 introduces the GBF architecture, where providers can play two roles: that of the Service Owner (SO), which is a provider of composite services; and the Element Owner (EO), which is a provider of service elements. Both the SOs and the EOs use the Business Layer (BL) as a channel to exchange information (which is performed by means of Web Services technology [W3C 04]) and reach agreements that regulate the provisioning of their services. The SO and the EO are responsible for publishing service and service element offers respectively, at the service directory. They use the Publisher component to manage their offers at the service directory (publish, update and withdraw). These offers are represented by service templates, which are eXtensible Markup Language (XML) documents that contain business and technical information about the services and the service elements. The Universal Description, Discovery, and Integration (UDDI) framework [OASIS 04] was chosen as the service directory, since it is a widely recognized standard registry for Web Services [Liu 05].

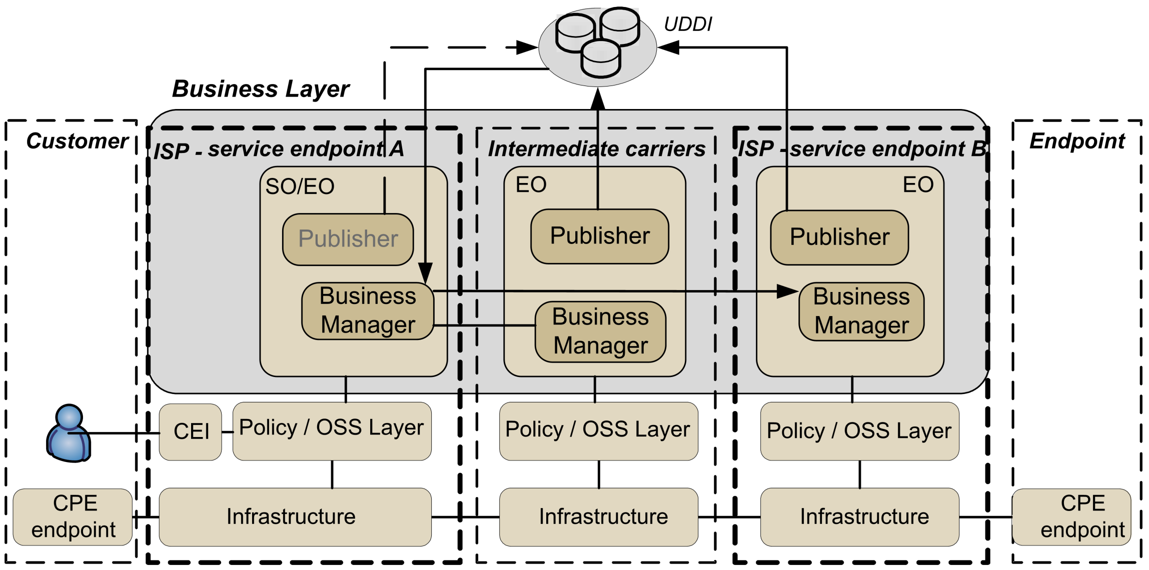


Figure 3.1: GBF architecture

The BL is at the top of a Policy/Operational Support System (OSS) layer that is responsible for providing classical OSS and policy-based management services, such as admission control, billing, monitoring, etc. By means of the Policy/OSS layer, the BL has access to information about network resources and provider management policies and this allows the GBF to adopt a Policy Based Network Management (PBNM) approach [Strassner 04] when providing a service. In doing this, a provider can select the policy that best suits its objectives when offering and configuring its resources.

The Infrastructure layer corresponds to the configuration of the network resources, where the service is effectively executed. It is this layer that is connected to the Customer-Premises Equipment (CPE) so that it can provide the requested service to the customer. It has a close relationship with the Policy/OSS layer. Equipment configuration operations and monitoring procedures are usually handled by the OSS software mentioned above, depending on the interfaces that are available in the equipment.

When a customer wants a service, he/she contacts the SO through the Customer Entry Interface (CEI) to give information about the particular requirements of the desired service. The SO can be either his/her usual access provider or a third party provider. Depending on the way it is implemented, the CEI can be either a simple web portal or a more complex OSS-hosted application accessed by the customer through a client Application Programming Interface (API). After receiving the customer request, the Business Manager component of the SO searches the UDDI for available service elements that can be used to compose the requested service. There are three types of service elements:

- Connection Element (CoE): service elements that provide connection services and/or transport services between two providers;
- Access Element (AcE): service elements that provide access services for customers; and
- Application Element (ApE): service elements that provide any type of application service apart from the two above-mentioned service elements (e.g. e-mail, video server, FTP server, billing services, file storing and banking).

In composing the service, the SO uses internal service policies that determine how it must combine the service elements. After the SO has composed the service, it contacts the EOs that own these service elements to confirm that they can be provided at the time required by the customer. After all the EOs have confirmed that the service elements are available, the SO can make an offer to the customer, by referring to the cost and service conditions as set out in a SLA. The SO also sends an SLA to each selected EO. As well as providing business information, these SLAs also include a SLS section, which describes the technical information of the service provisioning session. In the event of the customer and the EOs agreeing with the SLA terms, the SO again contacts the Business Manager components of the EOs to request a setup service. Thus, each EO configures its resource to support the requirements of the customer. This configuration is performed by means of scripts that are automatically generated at the EO. It is worth mentioning that the provider that acts as the SO may also act as the EO. This is the case when the service it sells is composed of service elements provided by partners and service elements provided by itself, for instance, when this provider turns out to be the access provider of the customer, and thus provides an access service element.

Although billing issues are beyond the scope of this study, payment distribution to the involved parties can be performed in a centralized fashion. The SO charges the customer the price of the service, which is the sum of the prices of each individual service element plus the profit of the SO. The SO is then responsible for distributing the amount received from the customer to each EO, depending on the service element costs.

GBF also supports service adaptations performed by the providers. Rather than resulting in service termination, these adaptations aim to guarantee the continuity of the service when faced with technical problems. During the provisioning of a service, the SO receives monitoring alert messages from third party agents that give information about the service status. If an alert message notifies that a violation has occurred, the associated penalties are imposed (the violations and penalties are specified in the SLAs that have been drawn up) and the SO may trigger an adaptation process. Adaptations can also be performed in response to requests made by any of the parties involved in the service provisioning process (customer, SO, and EO). For instance, a customer may request a service downgrade owing to financial constraints or an EO might request a service adaptation to give priority to other services.

When adapting a service, the SO finds out the new requirements the service must support. After that, it contacts the EOs that provide the service elements to confirm whether they are able to meet these new requirements. If they are, the SO updates the SLAs that have been previously established with the EOs and the customer. If the involved parties agree with these updated SLAs, the SO then requests the new configurations for the service elements, which are encapsulated in the SLS part of the SLA.

The last and simplest scenario supported by the GBF is service termination. This scenario arises when the service lifetime ends or an abnormal event occurs that prevents a service adaptation. If these situations arise, the SO requests the EOs for the service termination. The EOs then release the resources associated with the service elements that were part of the composite service.

Since the interaction in GBF occurs between a SO and an EO, each provider only knows details about the service element it is providing. EOs do not have access to information regarding the whole service or other service elements. By employing this approach, providers can conceal their business strategies from competitors. Nevertheless, although this approach protects sensitive business data, it does not restrict business flexibility. Both the SO and the EO can use internal policies to select their partners, on the basis of business, technical or economic criteria. For instance, a SO may select EOs in accordance with previous agreements and an EO can give discounts according to the inquiring SO.

Although the service provisioning outlined so far is an on-demand process, since the service creation is triggered by the customer requisition, the GBF also supports the provision of pre-contracted services. In this case, the customer makes a choice from services that have already been created, by using service elements from EOs that had previous agreements with the SO. The SO may also act as a service broker, by drawing attention to the service offers made by other SOs, which are available in the service directory. These scenarios can be easily handled by the GBF. However, the process of providing on-demand services is far more complex than the process of providing pre-contracted services. For that reason, this work is concerned with on-demand scenarios.

3.1.1 Assumptions

Managing communication services in inter-domain network environments is a highly complex task, because it requires dealing with several aspects of the interaction that occurs between different players involved in the process. Since GBF only tackles some of these aspects, a few assumptions have to be made to support its approach. These assumptions are as follows:

- Federation of providers: It is assumed that a "federation" of providers will be formed. Long-term contracts can be drawn up to establish this federation [Pouyllau 10], which defines some conventions that providers follow to support and guide their operations. These conventions include the following:
  - Universal identification: Every provider in the federation has its own identification which is widely used within GBF. Moreover, this unique identification can be extended to services, service elements, templates, SLAs, and so on. For instance, the global identification of a service is a local identification (identification in a provider's context) which is concatenated with the provider's identification;
  - Standard representation of SLAs and templates: The information comprised in the templates and in the SLAs follows a form of representation that is recognized by all the providers in the federation. For instance, every service template has the same tags to indicate the performance of a service;
  - UDDI accessibility: Every provider in the federation knows the UDDI address and has permission to publish, update, withdraw and search for services and service elements; and
  - Synchronized time: A synchronized time among all providers in the federation.
- Monitoring entity: Service monitoring is one of the factors that should be taken into consideration in inter-domain service management. In GBF, SO uses alert messages from monitor agents to decide if a service should be adapted. In view of this, it can be assumed that monitor agents from trusted third parties can be employed to monitor the service. Moreover, these monitor agents can be located at the edge devices and can access the traffic that is exchanged by the providers so that they can check the service conditions and send the alert messages to the SO;
- Security and billing issues: It is important to handle security and billing to create a complete inter-domain provisioning environment. However, as these are not within the scope of this study, it is assumed that providers are able to exchange information in a secure way, and that they can handle billing issues, such as the distribution of the payment.
- SLA negotiation: SLA negotiation is not examined in this thesis, even though GBF automatically creates SLAs and supports their establishment by exchanging

them between the parties involved in a provisioning process. SLA negotiation is a complex task, and has been the subject of several research studies in different areas, such as Web Services [Zulkernine 11] and Service Oriented Computing (SOC) [Ismail 10]. Nevertheless, it should be noted that GBF is flexible enough to support the deployment of SLA negotiation approaches by allowing an exchange of messages through the BL.

3.1.2 Service Templates

Service templates are used by providers in GBF as the main structure to exchange information through the BL. They are XML documents that contain business and technical information about services and service elements. Providers make their offers available by publishing service templates at the service directory. The adoption of XML as the data language structure to represent the templates facilitates the exchange of information as well as the handling of data. Since services and service templates represent the resources providers are willing to make available, the service templates can be considered as the virtualization of these resources. There are two types of service templates:

- Element Specification Template (EST): templates that represent service element offers and are published by EOs at the UDDI; and
- Service Specification Template (SST): templates that represent service offers and are published by SOs at the UDDI.

The templates can be regarded as the starting point for the SLAs that are established between the customer and SO, and SO and EOs. Significant parts of the SLAs are derived from the information contained in the templates, such as the technical section, also known as SLS. For this reason, the structure of the templates was inspired by works that aimed to define a basic set of attributes that the SLAs and SLSs should include (TEQUILA European Project [TEQUILA 00], Georgievski *et al.* [Georgievski 03], Bouras *et al.* [Bouras 05] and Dobson *et al.* [Dobson 06]).

Figure 3.2 shows the XML Schema Definition (XSD) of the EST. Owing to page size restrictions and the need to avoid compromising the readability of the XSD, Figure

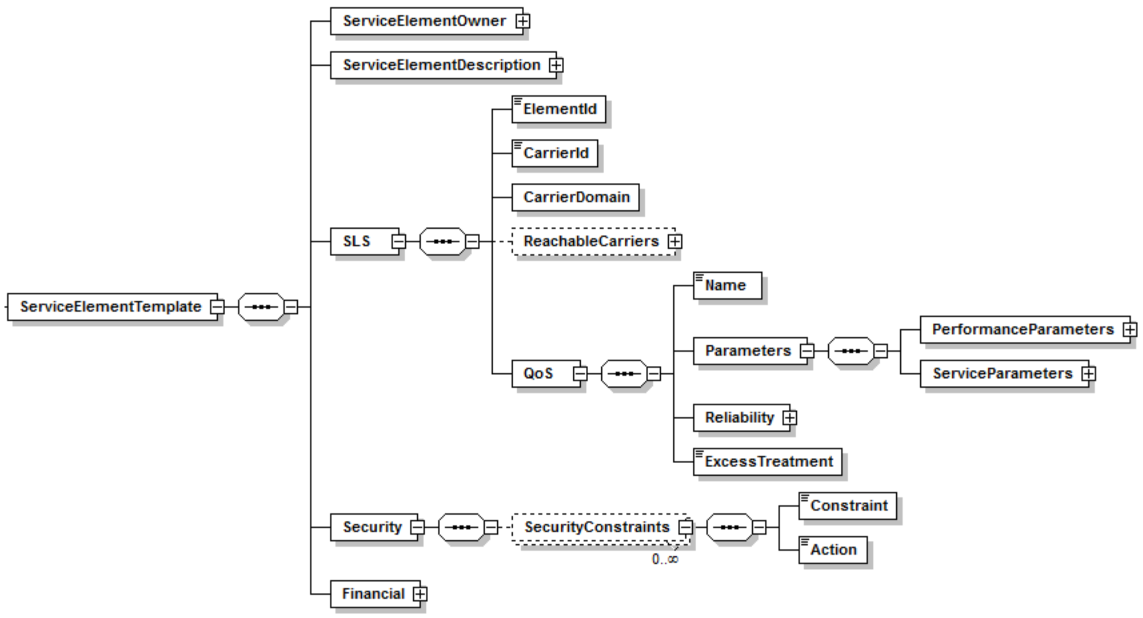


Figure 3.2: Element Specification Template

3.2 only displays a fragment of the template scheme. Appendix A provides a detailed description of the entire XSD. Although the XSD of this figure represents the structure of the ESTs, it is valid for the SSTs as well.

The *ServiceElementTemplate* (or *ServiceTemplate* in the case of a SST) element (Figure 3.2) represents the root element of ESTs (or SSTs). It is the main element of the template, and encompasses the administrative, business and technical information regarding the service elements (or services). Administrative information depicts the identification of the service element and the EO that provides it. Business information is used to establish legal relationships between the parties involved (e.g. warranty guarantees and financial settlements). Technical information is used by the SO to compose services that conform to customer requirements. The other elements of the *ServiceElementTemplate* are as follows:

- *ServiceElementOwner*: Includes the identification and the contact information of the EO that provides the service element;
- *ServiceElementDescription*: Contains a general description of the service element, such as the name, type and publication date at the UDDI;
- *SLS*: Contains technical information about the service element. It contains the following sub-elements:

- *ElementId*: The service element identification;
- *CarrierId*: The identification of the EO that provides the service element;
- *CarrierDomain*: The domain of the EO that provides the service element. The SO uses this element during the service composition process to build the graph representation based on the domain connectivity between the providers;
- *ReachableCarriers*: The domains that are reachable from the EO that provides the service element. This element is also used during the service composition process. It is optional. Only CoEs have this element, since they provide service connection between the domain of the EO (*CarrierDomain*) and the reachable domains (*ReachableCarriers*); and
- *QoS*: The QoS related information of the service element. It contains the following sub-elements:
  - *Name*: Name of the QoS class (e.g. Basic, Silver, Premium);
  - *Parameters*: This element contains the performance and service-specific parameters the EO guarantees for its service element;
    - *PerformanceParameters*: represents QoS performance parameters that the provider (EO) guarantees to the customer (SO). For instance, delay, jitter and throughput.
    - *ServiceParameters*: represents service-specific parameters that the provider (EO) guarantees to the customer (SO). For instance: encoding and frame rate.
  - *Reliability*: This element defines the mean downtime per year (MDT) and the maximum time to repair (TTR) in the case of service disruption, if applicable. Both *MDT* and *TTR* elements have a *Value* and a *Unit* sub-elements; and
  - *ExcessTreatment*: This element specifies how the out-of-profile and/or in excess packets should be handled.

- *Security*: Defines security requirements concerning the service element and the associated measures to be taken in the event of these requirements not being met. It may contain several *SecurityConstraints* elements that are optional and all of them have the following sub-elements:
  - *Constraint*: It specifies a constraint that must be satisfied; and

  - *Action*: It specifies the action that must be taken in the case the constraint is not satisfied. For instance, a constraint may state that service element can only be accessed through an encrypted tunnel. Otherwise, the EO does not provide the service element.

- *Financial*: Defines the monetary costs incurred by activities requested by the contractor of the service element (e.g. service activation and service termination).

The *Reliability*, *ExcessTreatment* and *PerformanceParameters* elements were greatly influenced by the TEQUILA project, which defined these parameters. *Financial* and *Security* elements were influenced by other studies, such as those of Bouras *et al.* [Bouras 05] and Dobson *et al.* [Dobson 06]. The *ServiceElementOwner*, the *ServiceElementDescription*, the *ReachableCarriers* and the *ServiceParameters* elements were defined in the context of this study.

Service templates allow providers to define their services and service elements in formal terms, thus creating a standard data structure where information can be exchanged in a seamless fashion. Nevertheless, there is still a need for a medium in which these providers can exchange this information.

3.1.3 Business Layer

The Business Layer (BL) concept emerged from the need to create a common infrastructure where providers can interact as a means to offer, contract and establish services in a dynamic way [IPsphere 07a, Howarth 05, Cortese 03]. It aims to alleviate the cumbersome process of establishing agreements between providers from different domains with distinct business views. This process can take several days to carry out and requires human interaction, which can lead to error-prone situations. The BL can overcome these limitations since it is able to take account of the different business views of the providers and reach agreements promptly and in a dynamic way, thus leveraging end-to-end service provisioning in multi-domain scenarios.

As mentioned earlier, in GBF, a provider can be either a SO or an EO. Both the SO and the EO can be regarded as business actors, in the sense that they play well-defined business roles when providing services. If a provider is an EO, it provides service elements to SOs. If a provider is a SO, it contracts service elements from EOs (including itself) to create a service, and provides this service to a customer. The BL strengthens this business relationship by acting as a channel through which providers can exchange messages defined in the GBF. These messages are exchanged between SOs, EOs, the service directory and the third party agents used to monitor the service (which also can be viewed as service providers).

Thus, GBF has a similar behavior to that of Ipsphere. However, IPsphere concentrates the operations it performs in the Service Structuring Stratum (SSS) in three groups of signalling messages - *Setup*, *Execute*, and *Assure*. The *Setup* messages are triggered for initial business negotiations, when the provider seeks to establish contractual agreements with each party, which in their turn can negotiate service parameters with the provider. The *Execute* messages are triggered after the providers have established the contracts. These messages start the service execution, by allocating resources and changing the management settings for each provider. Finally, the *Assure* messages monitor the service execution and report any problem to the Administrative Owner (AO) of the IPsphere.

Figure 3.3 shows the messages defined in this work and the entities that trigger each message. These messages are:

- *publishService*: this is used to publish services or service elements at the UDDI. When the EOs want to make their offers available, they create ESTs containing the information about their service elements and send them to the UDDI through the *publishService* message. At this stage, the ESTs are registered as service element offers, and since every service element has a unique identifier, they can be distinguished from one another and looked up by other providers. The same process occurs when SOs want to publish their services;
- *updateService*: This is used to update services or service elements at the UDDI. When an EO wants to change its offer, it sends a modified EST to the UDDI through this message, thus ensuring that its service element offer is updated. The same process occurs when SOs want to update their services;
- *unpublishService*: This is used to remove service or service element offers from the UDDI;
- *getAvailableElements*: This message is used by SOs to acquire the service element offers published at the UDDI. Due to the limitations of UDDI, it is only possible to search for service elements by referring to their names and types;

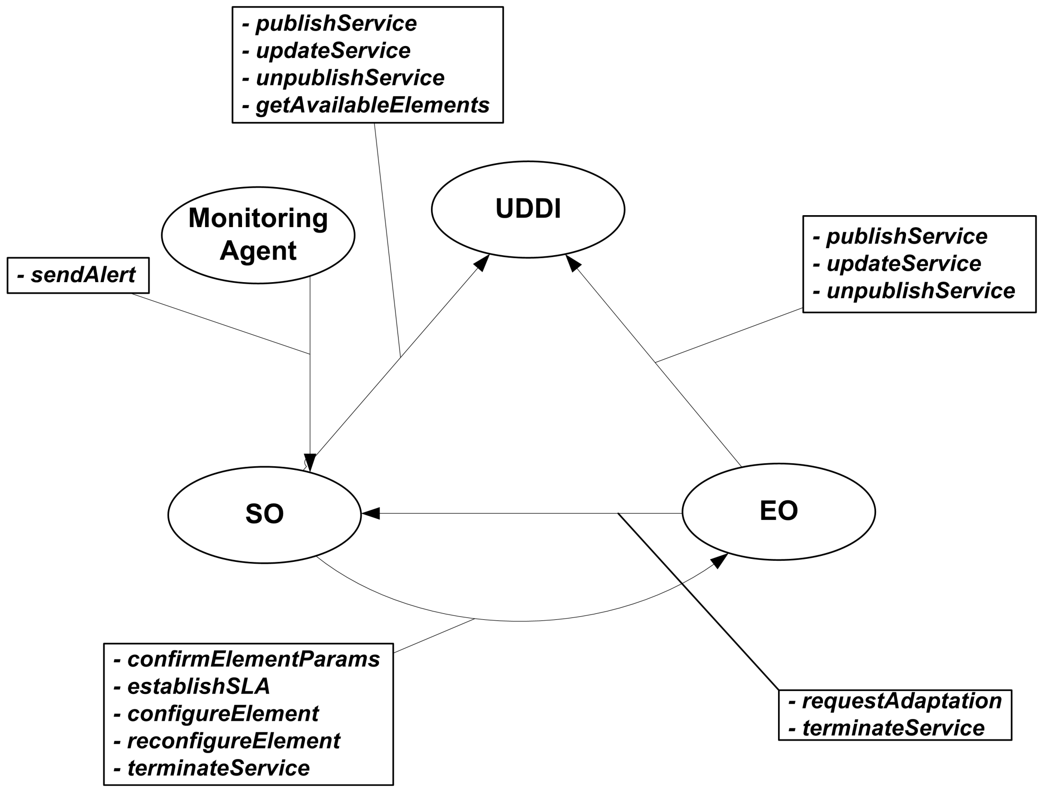


Figure 3.3: Messages exchanged in BL

- *confirmElementParams*: This message is used by the SO to determine whether the EOs can guarantee to meet the service element requirements advertised at the UDDI or not. If the EO can meet the requirements, it marks the resource as booked and thus reserves it for the SO. Otherwise, the SO excludes this EO from the provisioning process;

- *establishSLA*: This is used by the SO to send the SLAs to each EO involved in the service provisioning process;

- *configureElement*: This message starts the service provisioning process. It is used by the SO to request EOs to configure their service elements. From now on, the service is ready to be executed;

- *sendAlert*: This message is used by monitoring agents from third parties to send notifications about SLA violations to the SO;

- *requestAdaptation*: The EO sends this message to the SO to request a service adaptation. This message may trigger an adaptation process, though this depends on the terms that have been agreed. In this case, the appropriate penalties are imposed;

- *reconfigureElement*: The SO sends this message to the EOs to request service element reconfigurations. It is triggered when the SO detects an SLA violation that imposes a service adaptation or receives an adaptation request.
- *terminateService*: Both the SO and the EO can request the service termination. Several reasons may trigger this message, such as violations that do not allow a service adaptation, technical problems, etc. Appropriate penalties are applied, depending on the nature of each case.

Figure 3.4 shows the order in which the BL messages are triggered. First of all, the services or service elements must be published at the UDDI. Only then can the services or service elements be updated or unpublished. The remaining messages are triggered subsequently, in accordance with the requested processes (service establishment or service adaptation), which will be explained in detail in the next section.

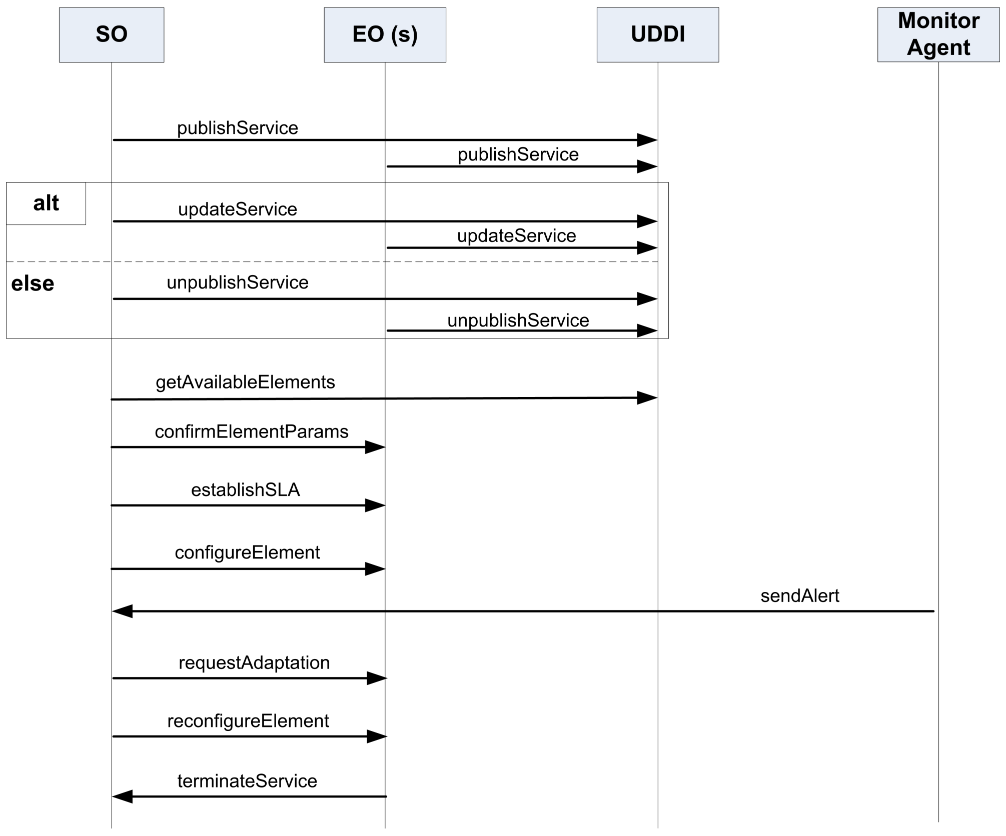


Figure 3.4: BL messages sequence diagram

All the above mentioned messages are Simple Object Access Protocol (SOAP) messages sent over Hypertext Transfer Protocol (HTTP). Apart from those that invoke operations in the UDDI, they are Web Service calls. The ESTs published at the UDDI contain the addresses of the Web Services at the EOs. Thus, when the SO obtains the ESTs, it can easily find these addresses. At the same time, the SO sends the addresses of its Web Services to the EOs and also to the monitor agent when they establish contracts (SLAs). The address of the UDDI is known by all the entities in GBF, since it is generally accepted that it is information shared by all the members of the federation of providers (see Subsection 3.1.1).

Initially, the first action performed by the SO when it receives a customer request, is to create a data structure called Service Order. This structure represents the service requested by the customer, and is used to manage the service information during the whole provisioning process. Each SO has a local database where it stores the structure and updates it in accordance with the service life cycle. Figure 3.5 shows a relational model diagram detailing the Service Order structure, where PK stands for Primary Key and FK stands for Foreign Key. A Service Order can have N SLAs signed by the parties involved in the service provisioning process and N Paths that can provide the service. A Path is composed of N Elements. The diagram shows the main information of each structure, which is described below.

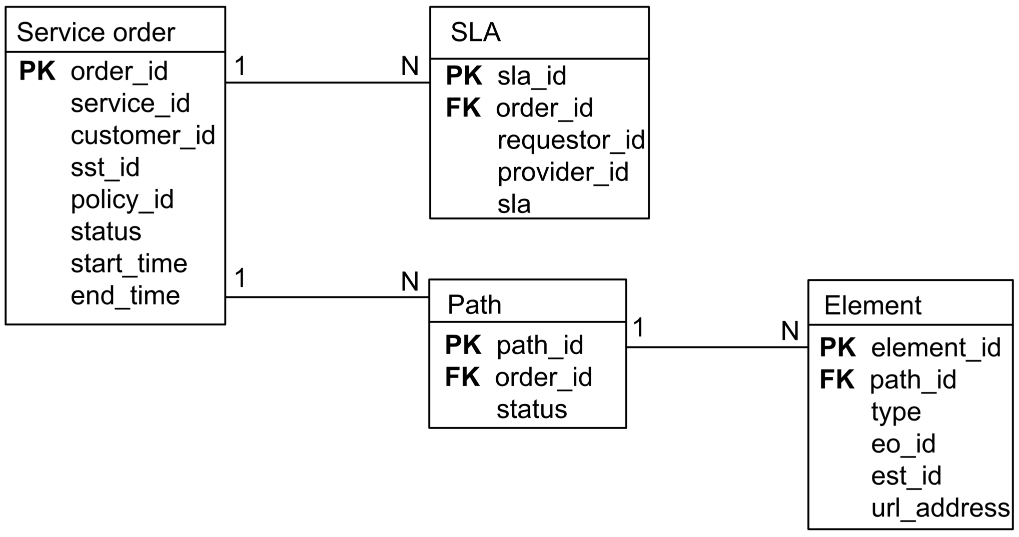


Figure 3.5: Service Order structure

Service Order

- *order_id*: Identifier of the Service Order - Primary Key (PK);
- *service_id*: Identifier of the requested service;

- *customer_id*: Identifier of the customer that requested the service;
- *sst_id*: Identifier of the SST associated with this service;
- *policy_id*: Identifier of the service policy that regulates the provisioning process;
- *status*: Current status of the service;
- *start_time*: Time required to start the service provisioning;
- *end_time*: Time required to finish the service provisioning;

SLA

- *sla_id*: Identifier of the SLA - Primary Key (PK);
- *order_id*: Identifier of the associated Service Order - Foreign Key (FK);
- *requestor_id*: Identifier of the customer that requested the service (if it is a SLA between a customer and SO) or identifier of the SO that requested the service element (if it is a SLA between the SO and EO);
- *provider_id*: Identifier of the SO that provides the service (if it is a SLA between a customer and SO) or identifier of the EO that provides the service element (if it is a SLA between the SO and EO); and
- *sla*: The SLA document.

Path

- *path_id*: Identifier of the path - Primary Key (PK);
- *order_id*: Identifier of the associated Service Order - Foreign Key (FK);
- *status*: Status of the path;

Element

- *element_id*: Identifier of the service element - Primary Key (PK);
- *path_id*: Identifier of the associated Path - Foreign Key (FK);
- *type*: Type of the service element;

- *eo_id*: Identifier of the EO that provides the service element;
- *est_id*: Identifier of the EST associated with this service element; and
- *url_address*: Address of the Web Service used to invoke the service element at the EO.

As can be noted from the information outlined above, the Service Order is the representation of a service request. It keeps the most important information about the service provisioning, such as the service elements that compose the service, and the contracts established between all the parties, as well as the policies that regulate this provisioning. The Service Order also maintains the state of the service request, as shown in the state diagram of Figure 3.6, which allows the GBF to manage and keep track of the service life cycle.

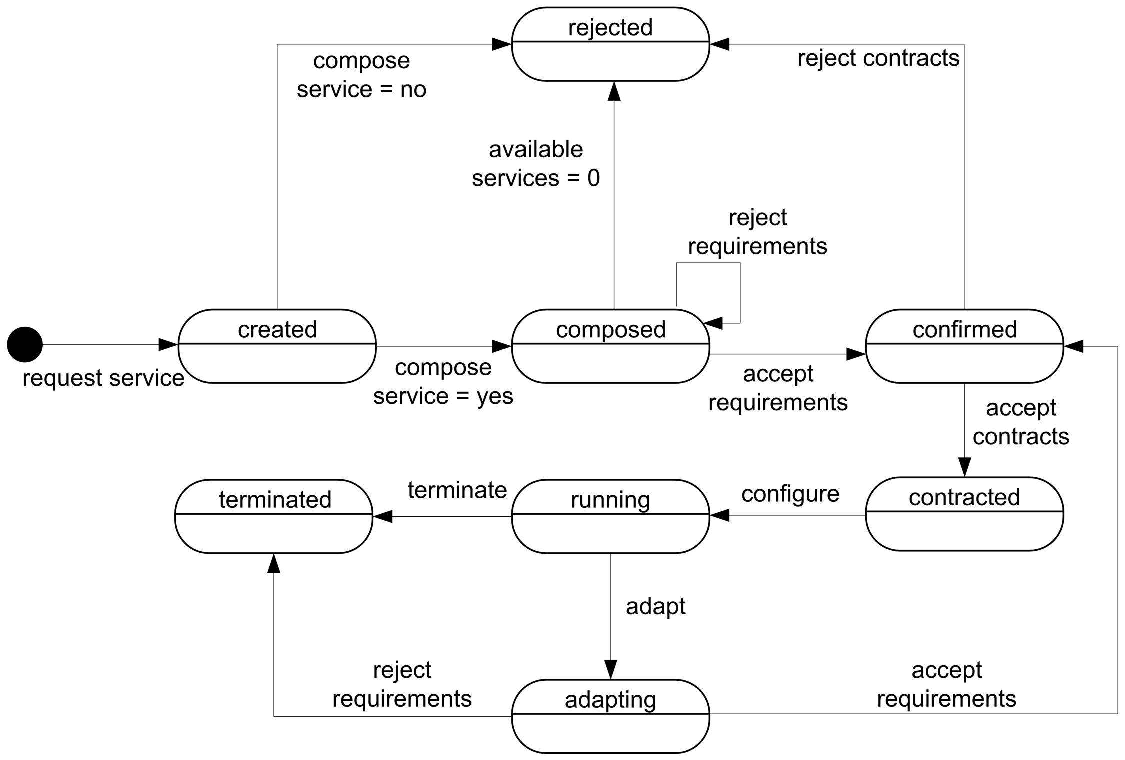


Figure 3.6: Service request state diagram

This state diagram illustrates how the main events that occur in GBF determine the service request behaviour. GBF is able to handle the different scenarios involving service provisioning from the time the service is requested until the service termination or rejection, which are the final states.

3.2 Summary

In this chapter, an overview of the GBF has been presented. It has been shown how the GBF handles inter-domain service requests performed by customers. GBF uses a SOA-based approach to support providers in creating composite services in an automatic and on-demand fashion. The aim of these composite services is to fulfill customer requirements, and they are formed of service elements offered by third providers. To allow these functionalities to operate, GBF relies on service templates to represent the services and on BL to support the information exchange.

Service templates are XML documents that contain the business and technical information about services and service elements. Providers publish service templates at the UDDI to make their offers available. They also use the service templates as the main data structure to exchange service and service element information with each other.

The BL is the communication channel that providers use to interact with each other. It supports the interaction between providers by means of Web Service messages, and this allows them to play well-defined business roles - both as the provider of services (SO) and provider of service elements (EO). This interaction also enables the providers to manage the service life cycle.

Chapter 4

QoS for Inter-Domain Services (QIDS)

Quality of Service (QoS) is a well-explored topic in computing systems and networks. QoS provisioning and management not only satisfy the requirements of customers for enhanced services, but also allow providers to optimize their resources. Providers expect that if they employ QoS mechanisms, such as DiffServ, IntServ or Multiprotocol Label Switching (MPLS), on their networks, their resources can be used more efficiently and at the same time increase the likelihood of customer satisfaction and market segmentation. Both circumstances can help to increase the revenue of providers since: (i) they can attend more customers if they have more available resources; and (ii) a contented customer will not change his/her provider and can act on their behalf as a free advertising agent.

Despite the recognized benefits of using QoS mechanisms, they are more widely employed in the providers' core networks. There is a lack of advanced services in multi-domain environments due to the complexity of managing these services [Rao 08]. In the Internet environment, each provider has its own policies, equipment and business strategies. This situation prevents the existence of a smooth interaction between providers, which makes it difficult for them to exchange the information needed to properly configure their equipment and manage their services and thus ensure a uniform service quality from one endpoint to another.

However, as the Internet evolves and new technologies/paradigms/methodologies emerge, the need to overcome the barrier of heterogeneity increases. For instance, in the Internet of Things [Spiess 09], the number of devices connected to the Internet will rise, as will the number of connections between the providers that offer services for these devices. Another example is the advent of Cloud Computing along with its three application scenarios [Vaquero 09], namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS), which can lead to situations where the providers of these services must interact intensively to satisfy a customer request. These are just some of the reasons that can stimulate the need for effective mechanisms to leverage the interaction between providers and enable them to manage inter-domain services with QoS guarantees. However, several issues must be tackled to achieve this goal, such as the mapping of service classes, service composition, service establishment and service adaptation.

In a composite end-to-end service, service elements (smaller services) must interact to accomplish the desired goal. Since in inter-domain environments, each provider defines its classes of services in accordance with internal policies and business objectives, it is necessary to employ a means to create an end-to-end path comprising service elements with equivalent quality of service levels. Standardizing classes of services is the usual approach to address this issue [Jacobs 05]. The problem is that if the classes of services that providers offer is regulated, their freedom to use any business strategy they want is restrained. Thus, some benefit can be derived from giving providers the flexibility to define service classes according to their own preferences.

Another factor to take into consideration when managing inter-domain services is how to compose an end-to-end service that fulfills the QoS requirements of the customer and, at the same time, complies with the business expectations of the providers [Burgstahler 03, Papazoglou 07b]. On the one hand, there seems to be a general consensus that providers no longer want to capitalize on connectivity services alone, but also to take advantage of end-user services. On the other hand, the demand by customers for more and more advanced services increases, as the infrastructure and transport technologies evolve to support these services. In view of this, simply using QoS connectivity parameters (e.g. delay, jitter and packet loss) to compose an end-to-end service may not fully comply with the customer expectations of the service or the providers' business strategies.

Once a service path (composed of service elements) has been found, the providers of these service elements must reach agreements and sign contracts, so the different parties can have guarantees from each other about the correct provisioning of the required

QoS levels. Moreover, owing to the heterogeneity of the providers' equipment, it is also necessary to create mechanisms that can translate the service requirements into appropriate resource configuration instructions, depending on the vendor equipment of each provider.

Finally, the adaptation of QoS also plays an important role in inter-domain QoS management. In real inter-domain provisioning scenarios, where different entities interact, there is a chance to interrupt the service when there are infrastructure malfunctions, contract violations, communication problems, etc. However, this is an unsatisfactory situation since it may cause customer frustration and eventually a loss of market share. Creating mechanisms that ensure that the service is carried out effectively, in spite of adverse circumstances, is thus an undertaking of great value.

The QoS for Inter-Domain Services (QIDS) model was developed and integrated in the Global Business Framework (GBF) to support the provisioning of QoS services in multi-domain environments. QIDS [Matos 10a] permits providers to define service classes in accordance with their own preferences [Matos 10b]. It also supports the composition of inter-domain services by assembling service elements and considering individual service parameters [Matos 10b]. Moreover, QIDS supports Service Level Agreement (SLA) establishments among providers and the automatic generation of configuration scripts that can be enforced in the equipment. Finally, it allows the adaptation of inter-domain services, so that they can guarantee that new QoS requirements are met [Matos 11b].

The remainder of this chapter is organized as follows: Section 4.1 provides an overview of the QIDS structure and how it is integrated in GBF. Section 4.2 shows the policies used by QIDS when performing its functions. Section 4.3 outlines the service classes specification and classification used in QIDS. Section 4.4 describes the inter-domain service composition process, while Section 4.5 examines the resource reservation, the SLA establishment and the generation of configuration scripts processes used in the inter-domain service establishment. Section 4.6 describes the QoS adaptation in inter-domain services, and finally, Section 4.7 summarizes the chapter.

4.1 Overview of QIDS

QIDS provides an automatic and on-demand solution to manage QoS-aware services in inter-domain environments. It follows Service Oriented Architecture (SOA) principles, in which a service is composed of service elements offered by different providers. By using dynamic mapping of service requirements into configuration scripts, QIDS allows providers to use any QoS model (DiffServ, IntServ, MPLS) they want inside their core networks. QIDS functionalities are supported by the integration of the QoS Manager component in the GBF architecture, as shown in Figure 4.1.

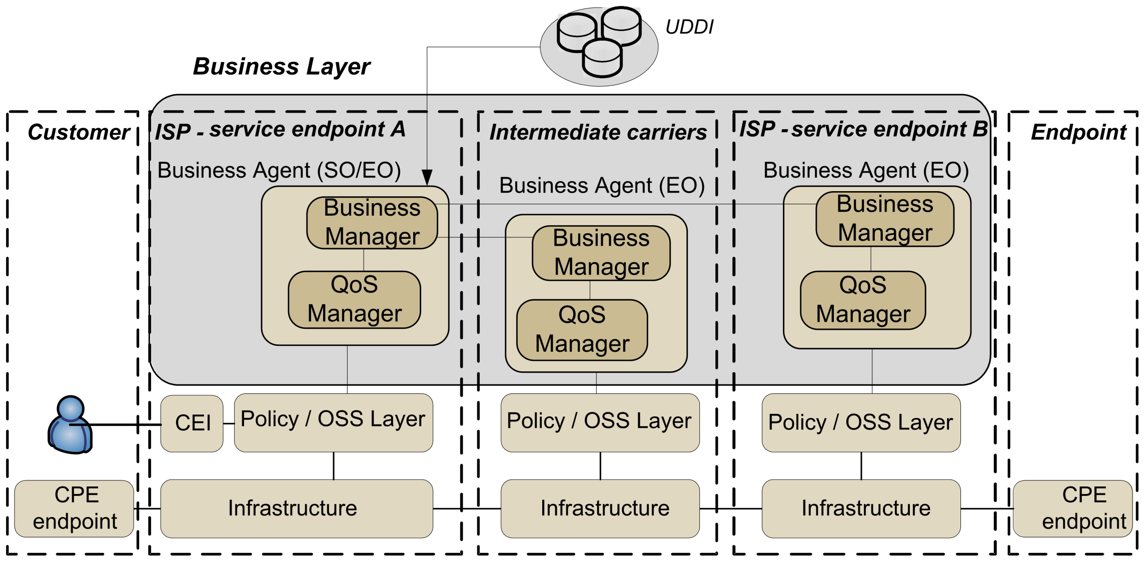


Figure 4.1: Integration of the QoS manager in the GBF architecture

In GBF, the QoS Manager component handles the QoS issues on behalf of the providers. This component interfaces with the Business Manager component and the Policy/Operational Support System (OSS) layer. Figure 4.2 shows an architectural view of the QoS Manager, which is composed of two modules: Policy and Configuration Handling and QoS Management. The Policy and Configuration Handling module is responsible for contacting the Policy/OSS layer in order to access the Policy Repository and the Resource Database of each provider. The Policy Repository contains the policies that GBF and QIDS use during the inter-domain service management process. The Resource Database is a simple database that contains information about the providers' resources.

Once the Business Manager finds service elements that can possibly be used to create an inter-domain service, it sends the Element Specification Template (EST) of each service element to the QoS Management module. The QoS Management module handles the QoS-related aspects of the inter-domain service management process. It is composed of three main mechanisms: Service Composition, QoS Adaptation and SLA Creation. The Service Composition mechanism creates possible end-to-end service

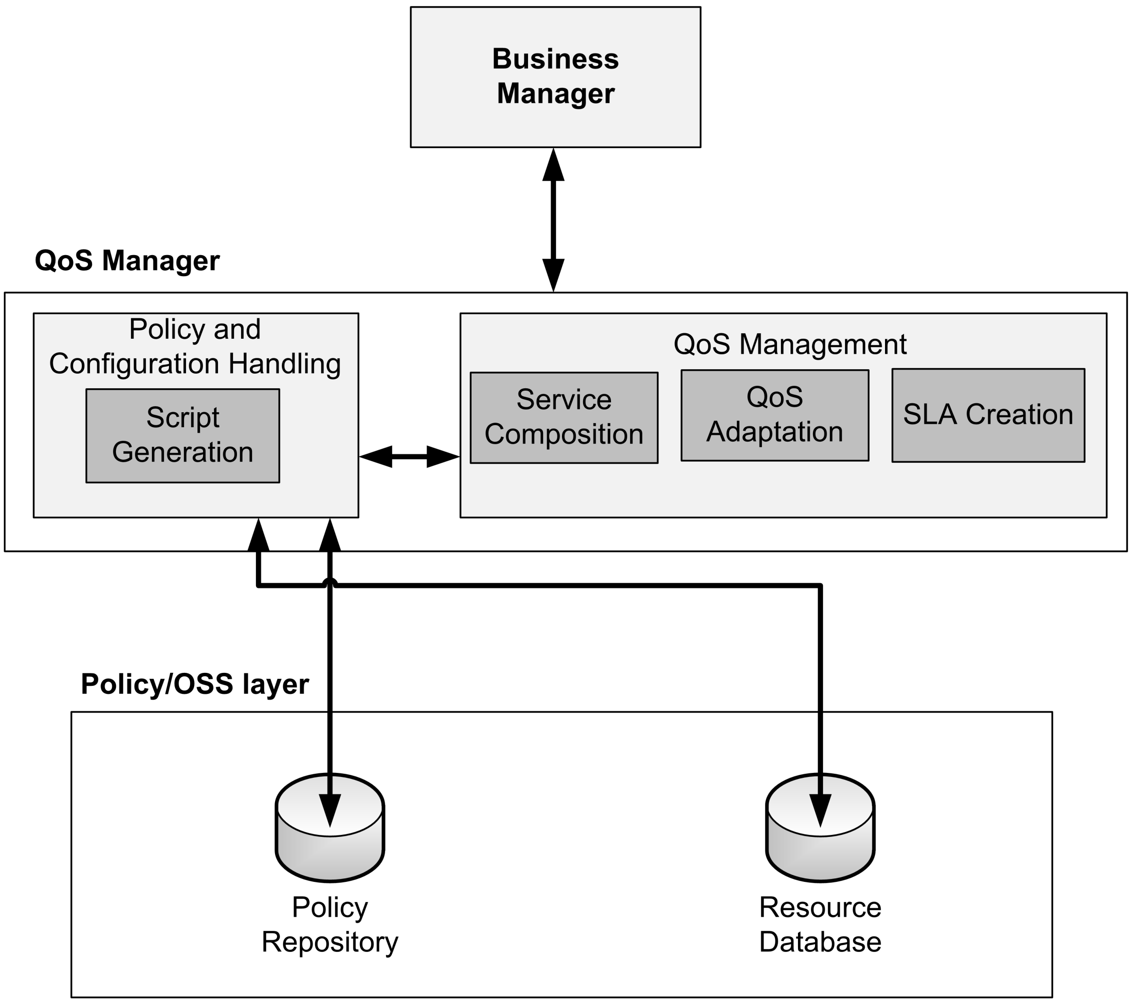


Figure 4.2: QoS Manager component

paths that can fulfill the customer requirements by comparing the QoS and service parameters of each EST. The SLA Creation mechanism supports providers in creating the SLAs that will be signed by the parties involved in the provisioning process. The QoS Adaptation mechanism allows providers to adapt the QoS of their services in case contract violations or abnormal events occur.

The Policy and Configuration Handling module also has a Script Generation mechanism that is responsible for creating the configuration script that is enforced in the provider's equipment, by mapping the service requirements into configuration instructions. The Policy and Configuration Handling module also contacts the Resource Database to find the available resources that can meet the service requirements, as well as the Policy Repository to obtain the policies used during the service composition and the service configuration processes.

The abovementioned mechanisms will be presented in more detail throughout this chapter.

4.2 Service Policies

Service policies are an important part of the proposed QoS model, since they are used as guidelines to perform crucial operations. For instance, policies are used to decide, among other factors, the following: which parameters have to be taken into account when composing an end-to-end path to provide a particular service level; and how to map out a service requirement into configuration instructions. For this reason, QIDS uses two main types of service policies: service composition policies and service configuration policies.

Service composition policies are used by QIDS during the composition of the end-to-end service path, which is triggered by a customer request. Every service offered by an Service Owner (SO) has a corresponding service composition policy, which specifies the different service levels (QoS) that are supported by this service. Each service level is characterized by a set of parameter values. Initially, the SO determines the service level that is suited to the customer requirements by comparing the requirements with the parameter values laid down in the policies.

The service composition policies also describe how to combine the different service element types (Access Element – AcE, Application Element – ApE and Connection Element – CoE) to create the service. For instance, an SO may define in its policy that a Premium class of its service should have a maximum of two CoEs to minimize the problem of packet loss. The composition policy also dictates how business issues affects the selection of the services. Table 4.1 shows fragments of a service composition policy, where each column presents a piece of information depicted in the policy.

The first column of Table 4.1 presents the piece of information regarding the type of service (*video*), its QoS level (*basic*) and its code in the context of each provider (*V001*). Furthermore, it also lists the number of each type of service elements that can be part of a basic video service. For instance, the policy states that the basic video service must have a maximum of five CoEs and a minimum of one CoE. The second column of Table 4.1 presents the parameter values that a basic video service has to support,

Table 4.1: Service composition policy

Service type and class	Parameters	Rules

service = video
level = basic
id = V001
elements
 ape
  max = 1
  min = 1
 end−ape
 ace
  max = 1
  min = 1
 end−ace
 coe
  max = 5
  min = 1
 end−coe
end−elements
	parameters
	 performance
	  delay
	   max = 150
	   unit = ms
	  end−delay
	  bandwidth
	   min = 500
	   unit = Kb/s
	  end−bandwidth
	 end−performance
	 service−specific
	  frame−rate
	   min = 20
	   unit =fps
	  end−frame−rate
	 end−service−specific
	end−parameters
		rules
		 rule
		  if
		   level = basic
		  then
		   final−price =< 200
		  end−if
		 end−rule
		 rule
		  if
		   eo = E001
		  and
		   eo C path
		  then
		   path−price = − %5
		  end−if
		 end−rule
		end−rules

along with the unit of each parameter. Finally, the third column of Table 4.1 shows the rules that providers can define in order to apply during the service composition process. The first rule states that if the SO is offering a basic service, then the final price of the service must not exceed 200$. This rule can limit the number of service elements used to compose the service, depending on how much they cost. The second rule states that if a specific Element Owner (EO) (*id = E001*) is present in a path, then the price of this path can be reduced by 5%. By using these rules, providers can create business strategies to differentiate their services and strength business relationships. For instance, the second rule can be applied to give priority to some EO that the SO already has a previous agreement with.

Other types of policies used by QIDS are the service configuration policies. These policies enable QIDS to map service requirements into configuration commands, which are enforced in the providers' equipment. For instance, by using these policies, QIDS can determine the group of commands that have to be executed on the router to allow a minimum bandwidth for a certain traffic. A service configuration policy is composed of actions, each of which is associated with configurations that are applied to carry it out. Each configuration has a set of mandatory parameters that are used in the instructions to perform the configuration. Table 4.2 shows extracts of a configuration policy, which describe the configurations required by a service to comply with the corresponding high-level instructions. The first column of Table 4.2 shows the configurations that are needed, along with their parameters, to connect to a neighbor provider. For instance, it is necessary for the interface, the IP address and the mask information to perform the *associate-interface* configuration. The second column shows the instructions and the parameter that are used to apply a QoS configuration. These high-level instructions are then mapped into configuration commands during the generation of the configuration scripts (see Subsection 4.5.3).

Table 4.2: Service configuration policy

Create route	Apply QoS

action
 connect−neighbor
end−action
configurations
 associate−interface
 create−bgp−route
end−configurations
instructions
 associate−interface
  interface = param1
  ip−address = param2
  mask = param3
 end−associate−interface
 create−bgp−route
  ip−address = param1
  as−number = param2
 end−create−bgp−route
end−instructions
	action
	 apply−qos−config
	end−action
	configurations
	 create−access−list
	 create−traffic−class
	 create−qos−policy
	end−configurations
	instructions
	 create−access−list
	  access−list = param1
	  interface = param2
	 end−create−access−list
	 create−traffic−class
	  traffic−class = param3
	 end−create−traffic−class
	 create−qos−policy
	  qos−policy = param4
	 end−create−qos−policy
	end−instructions

4.3 Service Classes

An important issue when dealing with QoS in inter-domain environments is the mapping of the service classes. To convey the service traffic between different providers in an appropriate way, it is necessary to find equivalent service classes along the entire end-to-end path. The usual approach in the literature is to standardize a set of service class definitions that all providers must use [Jacobs 05]. These definitions state the QoS parameter values (or range values) that each service level must support. For instance, one may define that a premium connection service in every provider must support a maximum jitter of 10 ms and a minimum bandwidth of 5 Mb/s. Despite the benefits derived from facilitating mapping mechanisms, the standardization of service classes has a drawback: it restrains the freedom of providers from following their business strategies. A provider that already has business strategies based on the service classes it offers, will have to adjust them according to the proposed standards. For instance, a provider may have a portfolio of services with well-defined service classes associated with competitive prices that are offered to a target customer base. If this provider adjusts its service classes to comply with a proposed standard, the price strategy of its services is very likely to change, which may affect the relationship with the customer base.

For this reason, QIDS allows a provider to define its service classes in accordance with its own preferences (the term service class is used in both the context of composite services and service elements throughout this thesis). In doing this, providers are able to maintain the business strategies that they have already implemented (or create new ones), and thus differentiate their services from those of their competitors. This autonomy leverages market competition, which eventually results in better and cheaper services. Hence, it can be assumed that every service must have a minimum set of default information that is used to advertise, search and compose these services. However, when considering the classes of the services, providers can define the parameter values on the basis of their strategies, which can result in a wide range of service class definitions. Table 4.3 shows possible QoS parameter values for two service classes of a CoE from two different EOs. In this case, each EO published two templates (one for each service class).

Table 4.3: QoS parameter values of CoE

Service class	QoS parameter	EO 1	EO 2
Basic	Jitter	10 ms	5 ms
	Bandwidth	512 Kb/s	1024 Kb/s
	Packet loss	5 %	3 %
Silver	Jitter	5 ms	3 ms
	Bandwidth	1024 Kb/s	2048 Kb/s
	Packet loss	3 %	1 %

In this example, given the fact that jitter, delay and bandwidth might be mandatory QoS parameters for CoEs, EOs should define the parameter values they think are most appropriate for each service class. In fact, they do not have to standardize the service class names as well, since the service composition process (shown in Section 4.4) takes into account the parameter values. This approach allows a more flexible and dynamic mapping mechanism, where EOs are able to create service classes independently from their neighbor domains. Figure 4.3 illustrates a service class mapping between three EOs that provide connection services (CoEs) from endpoint A to endpoint B. The small rectangles inside each EO represent service classes and the higher the number of rectangles inside an EO, the better is the quality of the service class. Each rectangle pattern represents a different level of service class quality; thus the rectangles with the same pattern inside different EOs have an equivalent service class quality. As can be seen, an end-to-end service path may encompass service classes independently from their level inside each EO. This allows EOs to define the service class granularity (number of service classes) they want, which increases their business strategy options.

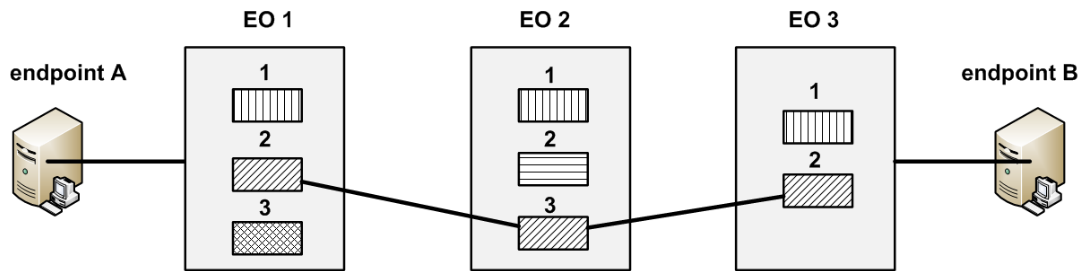


Figure 4.3: Service class mapping example

The service classes, along with the parameter values, are defined in the templates exchanged in the BL of the GBF.

4.3.1 Specification of Service Classes

Figure 4.4 presents a fragment of a template that shows how a service class is specified (a more detailed description of templates is given in Appendix A). Although this figure shows a template of a composite service (SST), the representation of parameters is the same for the templates of service elements (ESTs). The only difference is that CoEs and AcEs usually do not have service-specific parameters, while ApEs do not have performance parameters. Hence, depending on the template type, some parameters do not appear.

The QoS and service-specific parameters are declared in the *QoS* tag of the templates exchanged in the BL, while the business parameters are declared in the *Business* tag. The performance parameters (*PerformanceParameters* tag) are declared by using the *QuantitativeMaximum*, *QuantitativeMinimum* and the *Unit* tags, which define the maximum acceptable value, the minimum acceptable value and the unit of

```
<ServiceTemplate>
   ...
 <QoS>
  <Name>Silver</Name>
  <Parameters>
   <PerformanceParameters>
    <Jitter>
     <QuantitativeMaximum>30</QuantitativeMaximum>
     <QuantitativeMinimum>-1</QuantitativeMinimum>
     <Unit>ms</Unit>
    </Jitter>
    <Bandwidth>
     <QuantitativeMaximum>2000</QuantitativeMaximum>
     <QuantitativeMinimum>64</QuantitativeMinimum>
     <Unit>kbps</Unit>
    </Bandwidth>
   </PerformanceParameters>
   <ServiceParameters>
    <FrameRate>
     <Value>60</Value>
     <Unit>fps</Unit>
    </FrameRate>
    <Encoding>
     <Value>MPEG-1</Value>
     <Unit>unitless</Unit>
    </Encoding>
   </ServiceParameters>
  </Parameters>
 </QoS>
 <Business>
  <Availability>
   <ResponseTime>
    <Value>1</Value>
    <Unit>min</Unit>
   </ResponseTime>
  </Availability>
  <Financial>
   <Currency>$</Currency>
   <ActivationCharge>400</ActivationCharge>
   <InterruptionCharge>150</InterruptionCharge>
   <ChangeCharge>100</ChangeCharge>
  </Financial>
 </Business>
   ...
</ServiceTemplate>
```

Figure 4.4: Example of an SST showing the QoS parameters and the Business parameters

the parameter, respectively. The service-specific parameters (*ServiceParameters* tag) are declared by using the *Value* and the *Unit* tags. The *Value* tag defines the value of the parameter.

Performance parameters are related to connection services, and they usually include a wide range of values to be able to support a variety of services with different requirements [Chen 04, ITU-T 01, ETSI 10]. For this reason, these parameters have a range of acceptable values (minimum and maximum) to allow the creation of service classes to accommodate the different services. On the other hand, service-specific parameters do not have a range of acceptable values. Instead, they only have one acceptable value for each service class, since they are constrained by parameters with strict values. For instance, depending on the service class, a video streaming service can accept only MPEG-1, MPEG-2 or MPEG-4 as a value of the Encoding parameter.

The representation of the business parameters depends on the parameter that is declared. The example of Figure 4.4 shows two business parameters: Availability and Financial. The Availability parameter has the same representation as the service-specific parameters (*Value* and *Unit*), while the Financial parameter is declared by using the *Currency*, *ActivationCharge*, *InterruptionCharge* and *ChangeCharge* tags.

4.3.2 Classification of Parameters

Two distinct classification schemes of the parameters employed by QIDS to create the end-to-end service paths are used in this thesis. In the first scheme (Figure 4.5), the parameters are classified on the basis of QoS parameters (performance or service-specific) or business parameters. The QoS parameters were inspired by Jin and Nahrstedt [Jin 04] and Georgievski and Sharda [Georgievski 03], while the business parameters were defined in the context of this thesis:

- QoS parameters
  - Performance parameters: These parameters (also known as connectivity parameters) denote the performance of the network and are associated with connectivity and/or transport services. They are used as metrics to evaluate the effectiveness of the network when forwarding traffic. Jitter, bandwidth and delay are examples of performance parameters;

  – Service-specific parameters: These are parameters that are associated with specific services or applications. For instance, frame rate and encoding parameters for video streaming services, encryption parameters such as IP Security Protocol (IPSec) or Generic Routing Encapsulation (GRE) for Virtual Private Network (VPN) services, storage capacity parameter for backup services, and so on;

- Business parameters: These parameters are associated with the business aspects of the services. They are used by providers to improve their service offers, which means that they create business strategies that allow providers to differentiate their services from those of their competitors. Price, availability, warranties and penalties are examples of business parameters.

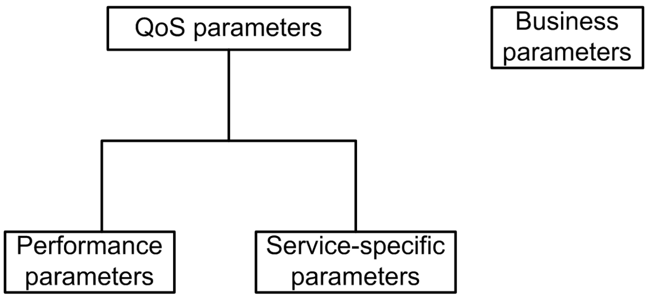


Figure 4.5: QoS parameters

In the second classification scheme, the parameters are categorized according to how their values can be combined to calculate the cost of the entire end-to-end service path. Let P be a specific parameter, *RV(P)* the value of P required by the service, *CV(P)* the value of P in an end-to-end service path composed of n service elements (E_1 to E_n), $V(PE_i)$ the value of P in each service element i and *A(EP)* the set of service elements E that contain the parameter P and pertain to the service path. The definition of each parameter along with its formula is as follows:

- Additive: A parameter is additive when the parameter value of the entire provisioning path is the sum of the parameter values of each service element along the path (e.g. delay, jitter, price) (Equation 4.1);

$$CV(P) = V(PE_1) + V(PE_2) + \cdots + V(PE_n) \tag{4.1}$$

- Multiplicative: A parameter is multiplicative when the parameter value of the entire provisioning path is the product of the parameter values of each service element along the path (e.g. packet loss) (Equation 4.2);

$$CV(P) = V(PE_1) * V(PE_2) * \cdots * V(PE_n) \tag{4.2}$$

- Concave: A parameter is concave when the parameter value of the entire provisioning path is limited by the minimum (or maximum) parameter value of all the service elements along the path (e.g. bandwidth) (Equation 4.3);

$$CV(P) = \begin{cases} min\{V(PE_1), V(PE_2), \ldots, V(PE_n)\} \\ max\{V(PE_1), V(PE_2), \ldots, V(PE_n)\} \end{cases} \tag{4.3}$$

- Independent: A parameter is independent when the parameter value does not need to be compared with the parameter values of the other service elements along the path. Only the service elements that have the parameter need to be checked and each parameter value is compared to the service requirement individually (e.g. most of the service-specific parameters). For instance, if a service requires a certain encoding value, it is not necessary to compare the service elements that have the encoding parameter with each other. Each of the encoding values are compared to the encoding value required by the service (Equation 4.4).

$$\forall E_i \in A(EP), \begin{cases} RV(P) = V(PE_i) \\ RV(P) \leq V(PE_i) \\ RV(P) \geq V(PE_i) \end{cases} \tag{4.4}$$

The first three categories are already defined by a well-known classification [Curado 05, Xiao 99] and are usually applied to performance parameters, while the fourth category was defined in the context of this work and is usually applied to service-specific parameters. Business parameters can be classified into any of the four categories.

4.4 Inter-Domain QoS-Aware Service Composition

Composing inter-domain services is an intricate process. It is necessary to combine the QoS features of each component to create an appropriate service path that satisfies the QoS customer requirements. Usually, the composition process only takes into account a set of QoS connectivity parameters (jitter, delay, packet loss, bandwidth, etc.) to create the service path (note that this is not a formal definition for connectivity parameters, but rather, a well-accepted characterization of these QoS parameters). However, providers have realized that they have to diversify their services and not simply rely on connectivity services, if they wish to leverage their revenue. When multi-domain environments are considered, this need becomes even more apparent, since there is an increase in competition. Thus, given that providers are likely to offer other services apart from connectivity [Bertin 09], it is necessary to consider the QoS parameters of each specific service: for instance, frame rate and encoding parameters for video streaming services, encryption parameters, such as IPSec or Generic Routing Encapsulation (GRE), for VPN services, storage capacity parameter for backup services, and so on. In addition, the use of service-specific parameters is likely to increase the degree of customer satisfaction [Ibarrola 10].

In such competitive scenarios (multi-domain negotiation), service composition must also take into account the business aspects [Pouyllau 09] of each service (price, availability, warranties, penalties, etc.), so the providers can take advantage of their strategies to achieve their business goals. To illustrate the above statement, it is possible to envisage a scenario where a customer must choose between two composite services. Both services satisfy the customer requirements; however, the customer might choose the service that offers the best warranties, regardless of whether or not it is the cheapest.

To tackle these issues, QIDS performs an automatic and on-demand inter-domain service composition that takes into consideration the QoS performance parameters, QoS service-specific parameters and business parameters to create end-to-end service paths.

4.4.1 Composition Mechanism

The inter-domain QoS-aware service composition is triggered when a customer requests a service provisioning in GBF. At this moment, the SO is responsible to handle this request by searching and combining service elements in order to provide a service that satisfies the customer requirements. Figure 4.6 shows a sequence diagram presenting the main operations performed in the QoS service configuration request.

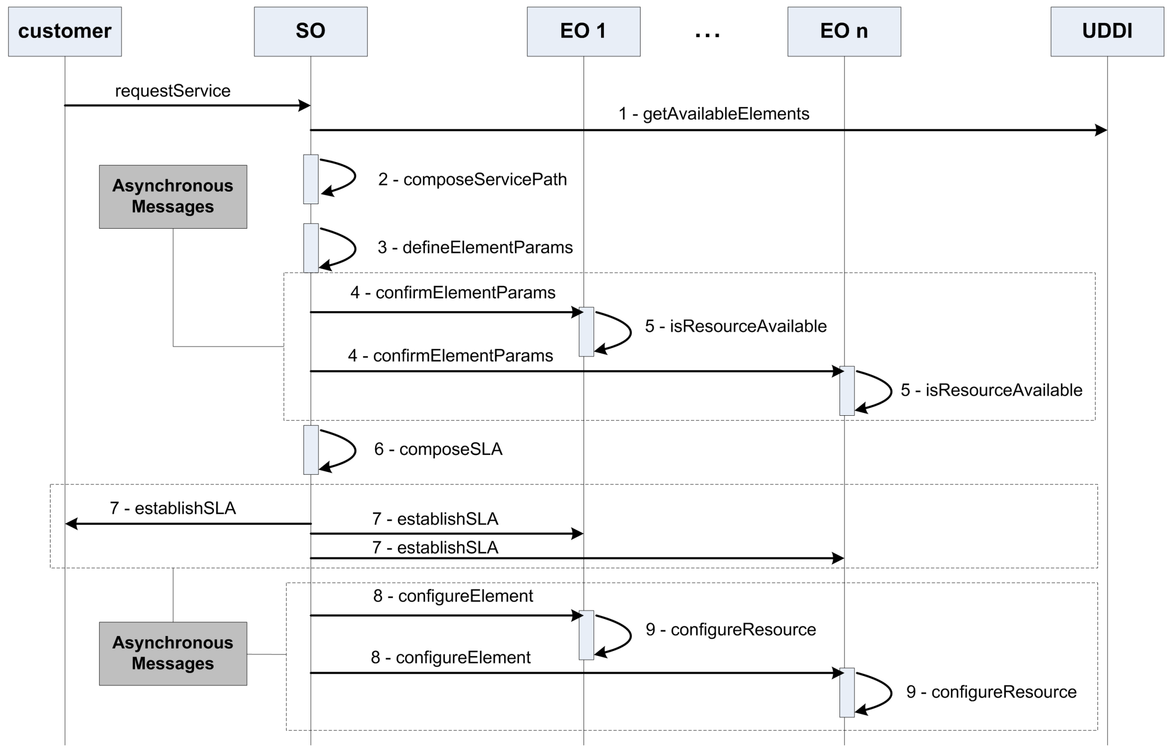


Figure 4.6: Sequence diagram for a service configuration request

Initially, the customer uses the Customer Entry Interface (CEI) to define the characteristics of the service he/she wants, such as the time period of the service and the desired service quality and some information required to provide the service, such as the endpoints' addresses. As mentioned above, the CEI may be either a simple web portal or a more complex OSS-hosted application accessed by the customer through a client API. If the customer is a skilled operator, he/she can fill in specific forms to outline the desired service characteristics. Alternatively, a more intelligent application could be used to spare the customer from having to enter into technical details. Whichever method is adopted, after the customer has made clear his/her service requirements, the CEI initiates the service configuration request by sending the *requestService* message to the SO.

The first action taken by the SO when it receives the *requestService* message is to create a Service Order structure that will comprise all the information related to the requested service. After that, it has to fit the customer requirements into one of its service composition policies. For instance, by analyzing the customer requirements of a VPN service, the SO can determine whether they can fit into a basic VPN service offered by itself. This policy dictates how the service must be composed and this is undertaken by, among other things, stating the maximum (or minimum) number of service element types that must be part of the service and the QoS parameter thresholds that the service must guarantee (e.g. the service composition policy can state that the VPN service must have at least two AcEs and one CoE and guarantee a minimum bandwidth of 5 Mb/s).

Afterwards, the SO searches through the UDDI for service elements that can be used to provide the service (*getAvailableElements*). Due to the limitations of UDDI, it is not possible to perform complex searches with it. In the case of this study, the search has been carried out by using the service element type criterion (AcE, CoE or ApE). Thus, the ESTs of every service element found at the UDDI are returned from the search. Once the SO obtains the ESTs, it starts the service composition process (*composeServicePath*) by comparing the service policy rules with the information from the ESTs so that a choice can be made of the service elements that guarantee the QoS parameter thresholds. These service elements are then used to create a possible end-to-end path to provide the service. The service composition is a three-phase process that will be detailed in the next subsection (Subsection 4.4.1.2).

When the SO finds a possible path, it builds partial configuration templates that define the parameter values that the service elements in the path must guarantee (*defineElementParams*). These partial templates contain the following: (i) the identification and address of the requestor SO; (ii) the global identification of the requested service, which is linked to the Service Order created in the SO (the *service_id* field in the Service Order); (iii) the identification of the EOs associated with each service element; (iv) the parameter thresholds that each service element must guarantee, which are in compliance with the the ESTs published at the UDDI and the service requirements; and (v) the service provisioning time, i.e., the time period stipulated for the service element to support these parameter thresholds.

Once the partial templates are built, the SO contacts each EO (*confirmElementParams*) that owns a service element to check if it can really deliver the service element quality parameters in accordance with the offer announced at the UDDI. The reason this confirmatory step is required, is that the update process of the service element offers, may not be on real time (it depends on the internal policies of each provider), and this may lead to non-synchronization between the offer and the current EO network capacity.

When an EO receives the *confirmElementParams* request, it carries out an operation to check the availability of its resources (*isResourceAvailable*). In QIDS, the operation just checks with a database to find out if there are resources that can guarantee the requirements and are available within the specified time period. If a resource is available, then it is associated with the requested service (*service_id*) and marked as booked for that time period, so that other requests cannot use the resource at the same time. The EO can establish a timeout for the SO to contract the booked resource, which occurs at the SLA establishment. When the timeout expires, the resource is released.

If all the EOs confirm that they have available resources to satisfy the request order, the SO composes the SLAs that will be established with each EO and with the customer (*composeSLA*). These SLAs will state the terms of the service (between the SO and the customer) and the service elements (between the SO and each EO) that must be followed to guarantee the correct service provisioning. The SO then sends the SLAs (*establishSLA*) to the customer and each EO. If the customer or any EO is unwilling to accept the SLA terms, they may refuse the SLA establishment or make a counter-proposal to the SO. Otherwise, the SO incorporates the configuration information necessary to create the end-to-end service provisioning path into the SLAs and sends them to the EOs (*configureElement*). The EOs then create and enforce the configuration scripts with the new parameter values into their resources (*configureResource*). At the end of this process the service is ready to be executed.

It is worth noting that in this process, as well as in the service adaptation process described in the next sections, the SO deals directly with each EO that it chooses to compose the service. EOs do not communicate with each other, which is a means of protecting sensitive business information. Each provider only knows about the service element it provides. Moreover, it may also prevent abusive behavior from a neighboring domain [Pouyllau 09], since an EO might request from a neighboring EO more resources than were necessary to provide a service.

In the next subsections, there is an explanation of how the service elements are connected and the three-phase service composition process.

4.4.1.1 Connection of Service Elements

The question of how the service elements connect to each other is a crucial aspect of the composition process. In the ESTs of each service element, there is a tag called *CarrierDomain*. This tag specifies the domain to which the service element belongs, and it is used to check whether this service element domain matches the service endpoint informed by the customer. For instance, during a video stream service request, the customer provides information about his/her endpoint IP address. The SO then inspects the *CarrierDomain* of each AcE to determine if its domain encompasses this IP address. If so, then the AcE is able to provide an access service to the customer. In the event of the customer requesting a different service and providing information of more than one endpoint address, such as VPN services, a similar process occurs. In this case, it is necessary to determine the AcEs that can provide access services for each endpoint.

Going back to the video stream example, the SO must find out the ApEs that can provide the video requested by the customer. When these ApEs are found, the SO must then create paths by connecting CoEs between these ApEs and the AcEs found previously. To do this, the SO uses the information provided in the *ReachableCarries* tag of the CoEs templates. This tag contains one or more *ReachableCarrier* tags, and each specifies a domain the CoE can connect to.

To create a connection path, the SO determines if a *ReachableCarrier* domain of a CoE matches the domain of an endpoint. If so, this CoE is connected to this endpoint. Next, the SO verifies if a *ReachableCarrier* domain of another CoE matches the domain of the same endpoint or the domain of a CoE previously connected to the path. If so, the new CoE is connected to the endpoint or to the CoE, respectively. This process continues until all the CoEs have been checked and the path reaches the second endpoint, which may result in several end-to-end service paths.

Although QIDS uses the domain as the information that can be used to check the connectivity between the service elements, the model supports the use of more specific information, such as subnet or IP addresses. There now follows a simple illustration of how the service elements are connected. Table 4.4 shows the domains the CoEs can reach. These domains represent the country abbreviations used in the Internet. Figure 4.7 illustrates how these service domains are connected by means of matching these domains.

Initially, the SO verifies the endpoints of the possible paths, i.e. the ApEs and AcEs. In this example, they are ApE1 and AcE1 (Figure 4.7a). Following this, the SO checks the reachable domains of CoE1 and finds out whether it can reach the *br* domain, and thus connect it to AcE1 (Figure 4.7b). When analyzing the reachable

Table 4.4: Domains and reachable domains of the service elements

Service element	Domain	Reachable domains		
ApE1	de	—	—	—
AcE1	br	—	—	—
CoE1	pt	br	ch	uk
CoE2	uk	it	br	es
CoE3	fr	es	uk	de

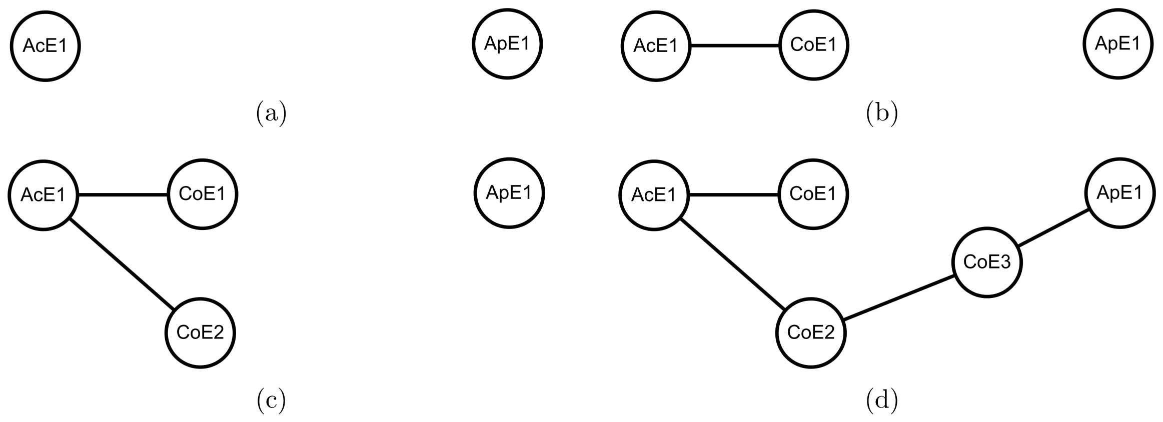


Figure 4.7: Example of connection between service elements

domains of CoE2, the SO determines whether this service element can also reach the *br* domain, thus also connecting it to AcE1 (Figure 4.7c). Finally, the SO checks the reachable domains of CoE3 by determining if it can reach the *uk* domain of CoE2 and the de domain of ApE1 (Figure 4.7d). As a result, the CoE3 is connected to CoE2 and ApE1, thus creating a possible end-to-end service path.

4.4.1.2 Three-Phase Process

The service composition process aims to create an end-to-end service path that satisfies the requirements of the service requested by the customer. To do so, the SO combines several service elements by taking into account the performance, service-specific and business parameters of each service element. Figure 4.8 illustrates this process, where the three key phases of the composition are highlighted: *Service elements filtering*, *Graph building* and *Paths sorting and filtering*.

Initially, the SO must fit the customer requirements into one of its service policies (*Policy matching*). As explained earlier, this is a simple operation, since the SO

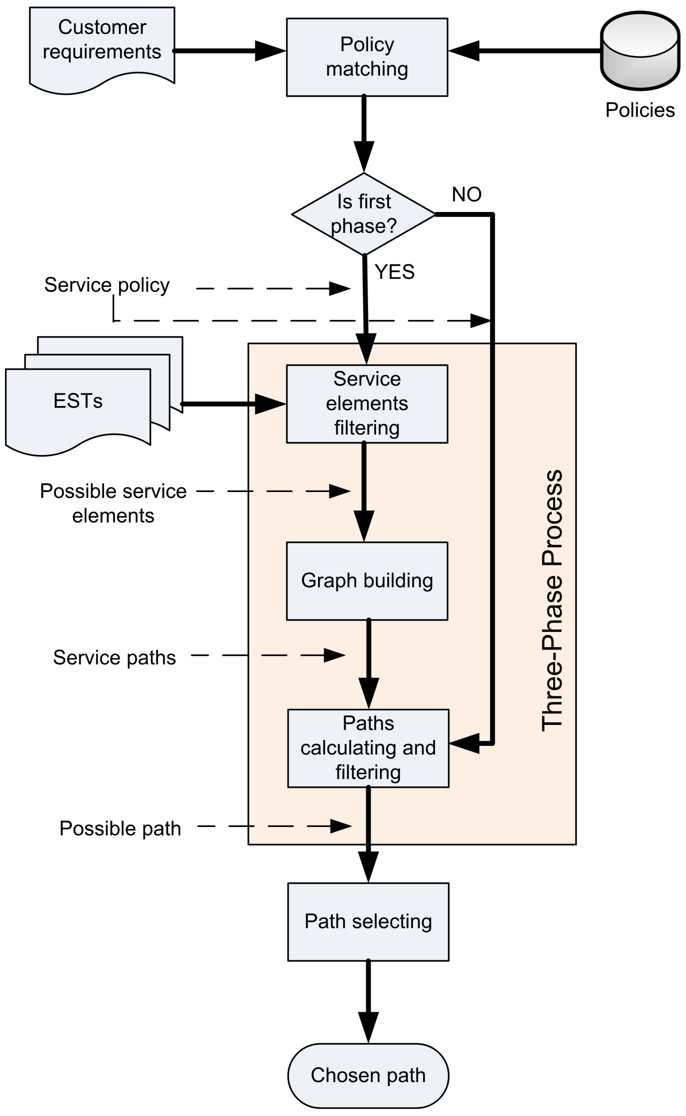


Figure 4.8: Service composition flowchart

determines the service level that is suited to the customer requirements by comparing these requirements with the parameter values laid down in the policies. By complying with the service policy rules and using the information from the ESTs, the SO starts the first phase of the composition (*Service elements filtering*), which is depicted in Algorithm 1. In this phase, the SO checks if the domains of the ApEs and AcEs found at the UDDI, can reach the endpoints informed by the customer (line 4). If any of the service elements cannot reach an endpoint, it is discarded (lines 6 – 8). The SO also compares the parameter threshold values from the service policy with the concave and independent parameters of each service element found at the UDDI. In the event that any parameter of a service element fails to comply with the corresponding threshold imposed by the service policy, the SO eliminates that service element (lines 9 – 15).

The output of this first phase consists of a list of service elements that comply with the parameters (concave and independent) from the service policy and can reach the endpoint domains specified by the customer (line 16).

Algorithm 1: Service elements filtering

```
   Input: seList, policyParamList, endpointList
   Output: seList
   // seList: List of service elements
   // policyParamList: List of service policy parameters
   // endpointList: List of endpoints
1  foreach (element ∈ seList) do
2      discard ← true;
3      foreach (endpoint ∈ endpointList) do
4          if (element = ApE or AcE) and ((ApE.domain = endpoint.domain) or
           (AcE.domain = endpoint.domain)) then
5              discard ← false;
6      if discard then
7          eliminateFromList(element);
8          continue;
9      seParamList ← element.params;
10     foreach (policyParam ∈ policyParamList) do
11         if paramPolicy.type = CONCAVE or INDEPENDENT then
12             foreach (seParam ∈ seParamList) do
13                 if (seParam.name = policyParam.name) and (seParam.value does
                   not satisfy policyParam.threshold) then
14                     eliminateFromList(element);
15                     continue;
16 return seList;
```

During the second phase of the service composition process (*Graph building*), depicted in Algorithm 2, the SO builds a graph representation of the connections between the service elements that were not discarded. To determine how they can be connected, the SO uses the reachable domain information contained in the templates of the CoEs and the domain information contained in the templates of the ApEs and AcEs (as explained in the previous subsection). In this algorithm, the domains of the ApEs, AcEs and CoEs are defined as nodes, and are included in the graph (lines 4 – 12). The algorithm also defines each domain that is reachable by every CoE, as a node (lines 13 – 14). A test is performed to find out if this node was already included in the graph. If it was not, the node is then included in the graph (lines 15 – 16). The association between the domain of a CoE and the domains that are reachable by this CoE creates the edges of the graph (line 17). A test to avoid the inclusion of duplicate edges in the graph is also performed (lines 18 – 19). If one of the reachable domains of a CoE matches the domain of an ApE (or AcE), this is proof that an edge exists between this CoE and the ApE (or AcE), since all the ApE and AcE domains were already included in the graph as nodes. The output of this phase is a graph that contains all of the possible paths between the endpoints (line 20).

Algorithm 2: Graph building

```
   Input: AcEList, ApEList, CoEList
   Output: G
   // AcEList, ApEList and CoEList:  List of service elements
   // G: Graph containing the paths between the endpoints
1  Vertex V, Vr;
2  Edge E;
3  Graph G;
4  foreach AcE ∈ AcEList do
5      V ← defineAsVertex(AcE.domain);
6      G ← graphAddVertex(V);
7  foreach ApE ∈ ApEList do
8      V ← defineAsVertex(ApE.domain);
9      G ← graphAddVertex(V);
10 foreach CoE ∈ CoEList do
11     V ← defineAsVertex(CoE.domain);
12     G ← graphAddVertex(V);
13     foreach reachDomain of CoE do
14         Vr ← defineAsVertex(reachDomain);
15         if $Vr \notin G$ then
16             G ← graphAddVertex(Vr);
17         E ← defineAsEdge(CoE.domain, reachDomain);
18         if $E \notin G$ then
19             G ← graphAddEdge(E);
20 return G;
```

During the last phase (*Paths calculating and filtering*), it is necessary to calculate the cost of each path. To do this, the SO uses the rules regarding the QoS parameter categories (additive, multiplicative, concave and independent) to compare the parameter values of every service element in each path. In this phase, another filtering process must be carried out so that only those paths that can guarantee the service requirements are selected. Algorithm 3 depicts this phase.

In this last phase, the SO calculates the k shortest paths using p as the path cost, where k is the number of paths that will be checked and p is any parameter, usually

Algorithm 3: Paths calculating and filtering

```
Input: G, policyParamList
// G: Graph containing the paths between the endpoints
// policyParamList: List of parameters defined in the service policy
1  Graph G;
2  BellmanFord BlF;
3  pathList ← BlF.calculateKPaths(G, k, p);
4  sortPaths(pathList, p);
5  foreach path ∈ pathList do
6      satisfy ← true;
7      foreach policyParam ∈ policyParamList do
8          if policyParam.type = ADDITIVE or MULTIPLICATIVE then
9              pathParam ← path.getPathParam(policyParam.name);
10             if pathParam does not satisfies policyParam then
11                 satisfy ← false;
12                 break; // Check next path
13     if satisfy then
14         a path was found;
15         break;
16 if not satisfy then
17     no path was found;
```

additive or multiplicative (e.g. price) (line 3). Both k and p are defined by SO internal policies. An implementation of the Bellman-Ford algorithm was used to calculate the shortest paths. Once the paths have been calculated, the SO then sorts these k paths according to p (line 4). After the paths are sorted, the SO picks the first path in the sequence and compares the additive and the multiplicative parameter values of the path with the service policy requirements. If this path satisfies all the requirements, it is selected for the customer as a feasible service provisioning path. Otherwise, the SO repeats this process with the next path in the sequence until no path remains (lines 5 – 15). It is worth remembering that both the concave and independent parameters of the service elements were already compared in the first phase, which means that a graph has been produced where all the paths fulfill the requirements of these parameters. Following, an example is given to illustrate this three-phase process.

Suppose that a customer requested a video streaming service and the SO found out that the customer requirements match a basic video streaming service offered by it. Table 4.5 displays the service policy for the basic video streaming, showing the performance parameter values (e.g. maximum delay, minimum bandwidth), the service-specific parameter values (e.g. encoding, frame rate) and the business parameter values (e.g. maximum number of hops, maximum price the customer is willing to pay) the service must meet. In its search at the UDDI, the SO finds eight available service elements. Table 4.6 shows the parameter values offered by each service element. Depending on each parameter, the values in Table 4.6 represent the maximum, the minimum, or a single value offered by the service element. By comparing the values in both tables, it is possible to see that ApE2 and CoE3 do not satisfy the frame rate and the bandwidth requirements, respectively. Thus, these service elements will be removed from the subsequent phases of the service composition.

Table 4.5: SO service policy (basic video streaming)

Parameter	Type	Value
Delay (ms)	Additive	< 150
Jitter (ms)	Additive	< 100
Bandwidth (Kb/s)	Concave	>= 500
Encoding	Independent	MPEG-4
Frame rate (fps)	Independent	>=20
Audio channels	Independent	>= 2
Number of hops[a]	Additive	<=3
Price ($)	Additive	<= 200

[a] In this thesis, number of hops is considered as the number of intermediary domains (CoEs)

Table 4.6: Service element parameter values

Parameter	AcE1	AcE2	CoE1	CoE2	CoE3	CoE4	ApE1	ApE2
Max. Delay (ms)	40	40	30	40	30	30		
Max. Jitter (ms)	40	40	20	35	30	30		
Min. Bandwidth (Kb/s)	512	512	1000	1000	384	512		
Encoding							MPEG-4	MPEG-4
Min. Frame rate (fps)							30	15
Min. Audio channels							4	2
Price	50	35	40	35	25	30	40	30

During the second phase, the SO uses the service elements that were not discarded to build the graph. Figure 4.9 shows an example of a graph resulting from the connections between the service elements. As can be seen from the figure, there are three paths from ApE1 (EO that hosts the video server) to the AcEs (EOs that can provide access service to the customer).

Once all the paths have been discovered, it is necessary to calculate the cost of each path. To do this, the SO uses the QoS parameter categories (additive, multiplicative, concave and independent) to compare the parameter values of the service elements

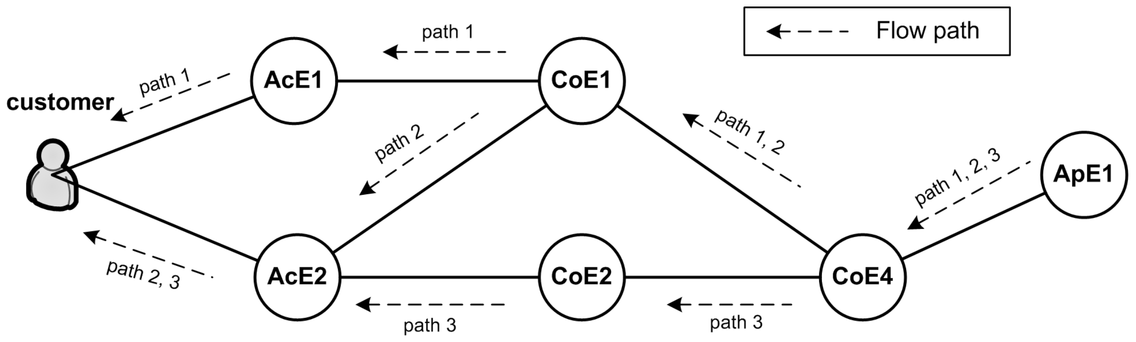


Figure 4.9: Graph representation of the service element connections

(shown in Table 4.6) that compose each path. For instance, the maximum delay of a path is the sum of the maximum delays of each service element in the path. Table 4.7 summarizes the paths and their costs as calculated by the SO.

Table 4.7: Service path costs

Parameter	AcE1–CoE1–CoE4–ApE1	AcE2–CoE1–CoE4–ApE1	AcE2–CoE2–CoE4–ApE1
Delay (ms)	100	100	110
Jitter (ms)	90	90	105
Bandwidth (Kb/s)	512	512	512
Encoding	MPEG-4	MPEG-4	MPEG-4
Frame rate (fps)	20	20	20
Audio channels	2	2	2
Price	160	145	140

In the last phase, the SO has to choose one path to present to the customer. Assuming that the SO sorts the paths by price, it is evident that, although the third path (*AcE2-CoE2-CoE4-ApE1*) is the cheapest, it exceeds the maximum jitter defined by the SO service policy (jitter < 100 ms). The SO then checks the second cheapest path (*AcE2-CoE1-CoE4-ApE1*) to find out whether it meets all the service requirements. If so, the SO offers this path to the customer as a feasible path to provide the service (*Path selecting*). If the customer accepts it, the service provisioning process continues.

It should be stressed that the purpose of this service composition approach is not to find the optimal path for the service provisioning. Rather, it aims to find a feasible path that can satisfy both the customer service requirements and the providers' business expectations by using the QoS performance parameters, the QoS service-specific parameters and the business parameters advertised by each service element. Furthermore, as can be seen from Figure 4.8, the service policy plays an important role in almost every step of the process, thus allowing an independent and dynamic mechanism to compose services.

4.5 Inter-Domain Service Establishment

As already discussed, the process of inter-domain service composition permits providers to create and select end-to-end paths that meet the customers requirements and the providers' business expectations. However, simply finding a suitable end-to-end path is not enough to guarantee the correct provisioning of the inter-domain service. Other factors must be considered to support the provisioning of inter-domain services with QoS guarantees.

For this reason, QIDS supports the automatic creation and establishment of SLAs between the parties involved in the provisioning process, as a means of formalizing the role of each party. Moreover, it employs a reservation in advance scheme to guarantee that resources are reserved for a requested service. Finally, QIDS also enables the automatic creation of configuration scripts, which are enforced in the providers' equipment.

4.5.1 Resource Reservation

A reservation in advance approach is used by the EOs to reserve the resources at the confirmation stage of the service provisioning [Wolf 97]. In this approach, the resource is not reserved at the moment the service starts, but rather, it can be reserved in a time interval before the execution start-time. This time interval can vary from a few seconds to as long as several months. However long it may be, at the confirmation step, the SO sends the following information to each EO: the parameter values that the service element of an EO must guarantee; the identification of the service that corresponds with the current service request (*service_id* of the service order); the time period during the service element must be provided; and the IP address, the Autonomous System (AS) number and the domain of the neighboring EOs to which this service element must connect. This connection information is gathered during the service composition process when the SO builds the connection graph.

In each EO there is a local database that stores the information about its resources. Whenever a confirmation request arrives at the EO, it checks in its database to find out if there is any available resource that meets the criteria sent by the SO. Figure 4.10 displays the structure of this database and the main information used to verify the service availability, where PK stands for Primary Key and FK stands for Foreign Key. Routers are the only resources represented in this structure; however, it can be easily adapted to accommodate other equipment, such as switches, firewalls, servers, etc. A real router can have N interfaces (physical and logical) and each interface is associated with a neighboring connection, which represents the neighbor EO that the interface can connect to. Moreover, an interface is associated with N configurations, where each configuration represents a real configuration that must be enforced in the router. The description of each field in this structure is given below.

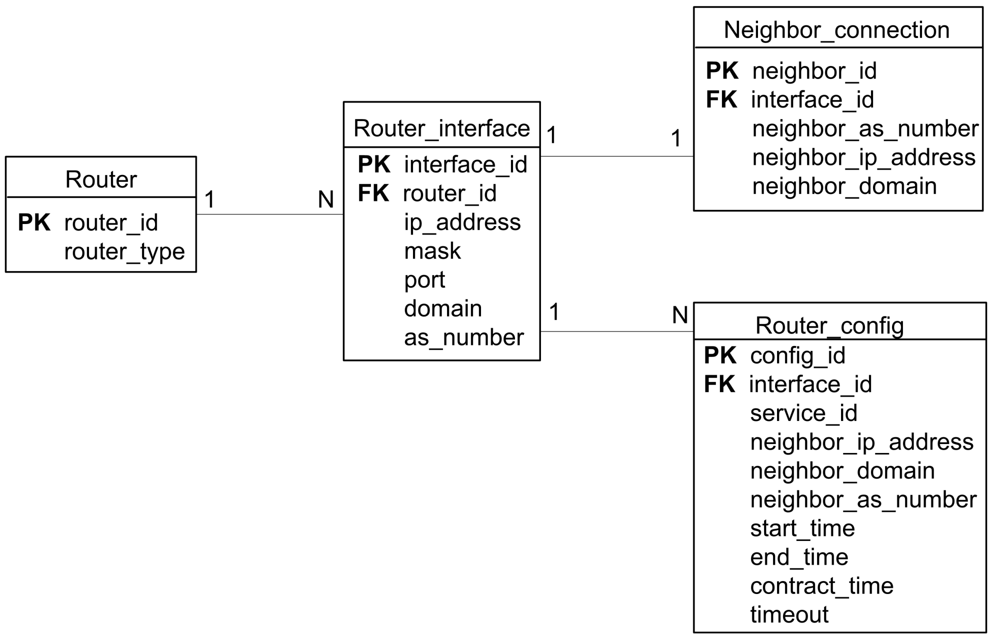


Figure 4.10: Resource database

Router

- *router_id*: Identifier of the router - Primary Key (PK);
- *router_type*: Type of the router;

Router_interface

- *interface_id*: Identifier of the interface - Primary Key (PK);

- *router_id*: Identifier of the associated router - Foreign Key (FK);
- *ip_address*: IP address of the interface;
- *mask*: Address mask of the interface;
- *port*: Port of the interface;
- *domain*: Domain to which this interface belongs;
- *as_number*: Number of the autonomous system to which this interface belongs;

Neighbor_connection

- *neighbor_id*: Identifier of the configuration - Primary Key (PK);
- *interface_id*: Identifier of the associated interface - Foreign Key (FK);
- *neighbor_as_number*: The AS number of the equipment to which the service element must be connected;
- *neighbor_ip_address*: The IP address of the equipment to which the service element must be connected;
- *neighbor_domain*: The domain of the equipment to which the service element must be connected;

Router_config

- *config_id*: Identifier of the configuration - Primary Key (PK);
- *interface_id*: Identifier of the associated interface - Foreign Key (FK);
- *service_id*: Identifier of the service associated with this request;
- *neighbor_ip_address*: The IP address of the equipment to which the service element must be connected;
- *neighbor_domain*: The domain of the equipment to which the service element must be connected;
- *neighbor_as_number*: The AS number of the equipment to which the service element must be connected;

- *start_time*: The time the service element provisioning starts;
- *end_time*: The time the service element provisioning ends;
- *contract_time*: The time the contract was established;
- *timeout*: The time that EO has to wait for the SO to contract the resource.

Algorithm 4 depicts the resource in advance approach used in each EO. All the input data of the algorithm consist of information sent by the SO to each EO. To determine the availability of resources, the EO checks if any of its router interfaces (Ri) can connect to a neighboring EO, by checking the IP address, the AS number and the domain of the neighboring EO (line 3). If the EO finds an interface that can connect to this neighboring EO, it verifies if this interface has already been reserved during the time period of the service provisioning (line 4). This is done by checking the start time and the end time of the configurations associated with the interface. If the EO does not find any configuration in the requested time period, it reserves the equipment by creating a configuration, so that other requests cannot use the resource at the same time (lines 5 – 6). The EO can also establish a timeout for the SO to contract the reserved resource (line 7), which occurs at the SLA establishment. If the timeout expires, the resource is released. In the event of there not being any available resources, the EO replies to the SO and terminates its execution (lines 9 – 10). The SO then starts up another composition process without using that EO.

4.5.2 SLA Creation and Establishment

To facilitate the information exchange and the automatic data handling, the SLAs are represented as XML documents. They define the QoS parameter thresholds the service must respect, along with the penalties that are incurred when these thresholds are violated. Moreover, the SLAs also include other information such as the identification of the provider and the requestor of the service, monitoring methods, reporting schedules, and financial rates, and so on. To create a SLA, the SO uses information from its own policies and from the ESTs. For instance, the SO uses the parameter values which the EOs have already confirmed they can guarantee as the thresholds which the service (or the service element) must respect. Regarding the financial rates, the charge for a service activation may be the sum of the activation charges of each service element

Algorithm 4: Resource reservation

```
   Input: paramList, neighborIP, neighborAS, neighborDomain, serviceID, start-time,
          end-time
   // paramList: Service element parameter values
   // neighborIP, neighborAS, neighborDomain, serviceID, start-time,
      end-time: Information from the neighbor EO sent by the SO
1  RouterInterface Ri;
2  Free ← false;
3  Ri ← findResourceConnectingToEO(neighborIP, neighborAS, neighborDomain);
4  Free ← Ri.isFree(start-time, end-time);
5  if Free then
6      configurationID ← Ri.createReservation(neighborIP, neighborAS,
       neighborDomain, serviceID, start-time, end-time, paramList);
7      createTimeoutThread(configurationID);
8  else
9      replyToSO(no resource available);
10     terminate();
```

plus a provider's percentage gain as stipulated in its policies. Figure 4.11 shows a small subset of an SLA signed between a SO and an EO.

An important part of the SLA is the *Violation* clause. This clause stipulates what actions must be taken when some violation occurs at the service provisioning. For instance, the *Violation* tag of the example in Figure 4.11 states that a service adaptation must be performed when the service element does not guarantee the minimum frame rate that was agreed on.

After the SLAs are created, the SO sends them to the customer and to every EO that is part of the end-to-end service path (Figure 4.12), thus forming a centralized negotiation model [Asgari 05]. If the customer or any EO does not agree with the SLA terms, they may refuse the SLA establishment and make a counter-proposal to the SO. The SO can then change some SLA terms and send them back again to the parties in order to reach the agreements. In this negotiation model, the SO negotiates directly with each EO and the customer. An EO does not have any knowledge of the agreements reached between the SO and other EOs, which helps to protect the confidentiality of the providers' business information.

It should be noted that this thesis does not intend to recommend what set of information the SLAs should comprise or establish an SLA negotiation model between providers. Rather, the aim is to show that QIDS can support the creation and

```
<SLA>
 <SLAId>10301</SLAId>
 <Business>
  <Service>
   <ServiceId>1043</ServiceId>
   <Type>ApplicationElement</Type>
   <Description>Video streaming</Description>
  </Service>
  <Parties>
   <RequestorId>SO_001</RequestorId>
   <ProviderId>EO_009</ProviderId>
  </Parties>
  <Financial>
   <Currency>$</Currency>
   <ActivationCharge>100</ActivationCharge>
   <InterruptionCharge>50</InterruptionCharge>
   <AdaptationCharge>30</AdaptationCharge>
  </Financial>
  <Violations>
   <Violation>
    <ViolationCode>V_001</ViolationCode>
    <ViolationParameter>Frame rate</ViolationParameter>
    <ViolationDescription>Minimum frame rate not guaranteed
    </ViolationDescription>
    <ViolationAction>Service adaptation</ViolationAction>
   </Violation>
  </Violations>
 </Business>
 <SLS>
  <ServiceParameters>
   <FrameRate>
    <Value>24</Value>
    <Unit>fps</Unit>
   </FrameRate>
   <Encoding>
    <Value>MPEG-4</Value>
    <Unit>Unitless</Unit>
   </Encoding>
  </ServiceParameters>
 </SLS>
</SLA>
```

Figure 4.11: Example of an SST showing the QoS parameters and the Business parameters

establishment of SLAs by dynamically using information from policies and ESTs to build these SLAs and create an infrastructure in which providers can exchange messages and thus allow proposals and counter-proposals to be made.

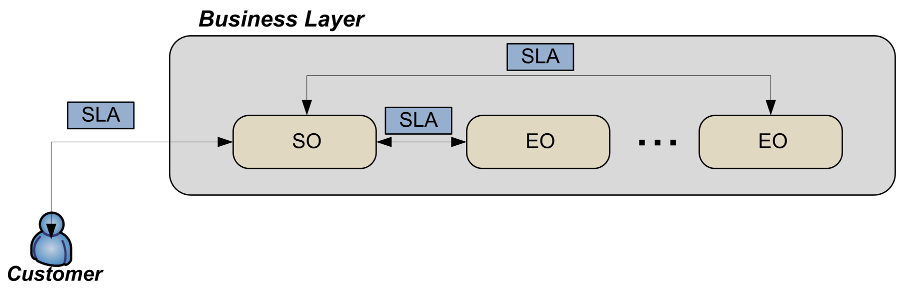


Figure 4.12: SLA establishment example

4.5.3 Generation of Configuration Scripts

In order to configure the service elements, the SO must incorporate the configuration information necessary to create the end-to-end service provisioning path into the SLAs and send them to the EOs. If the EO does not receive the configuration information before the timeout, the resource is released. Otherwise, the EO will create a configuration script for the resource that has been previously reserved. In the latter case, the EO may also update its service element offer at the UDDI, since its free resource capacity has been reduced. The frequency of the update process is dictated by the internal policies of each EO.

The generation of the configuration script (illustrated in Figure 4.13) is a process that involves the combination of information from several sources. First of all, the EO gets the service element identification from the SLA so it can obtain the corresponding EST. The EST then identifies the appropriate configuration policy, which defines high level instructions and the parameters for these instructions. Depending on the requirements received from the SO and the type of the service element involved, these high level instructions are combined in a specific order, so that they can specify how the service element must be properly configured. Once these instructions have been located, the EO then uses the equipment information from the resource database and vendor-specific commands to map out these instructions into final configuration commands.

Algorithm 5 depicts in more detail the generation of the configuration script. After the service element identification, the EST, the configuration policy and the SO requirements have been obtained (lines 1 – 4), the algorithm starts the process of mapping every requirement into high-level instructions (lines 5 – 6). All the requirement types that can be requested by the SO are previously known by all of the

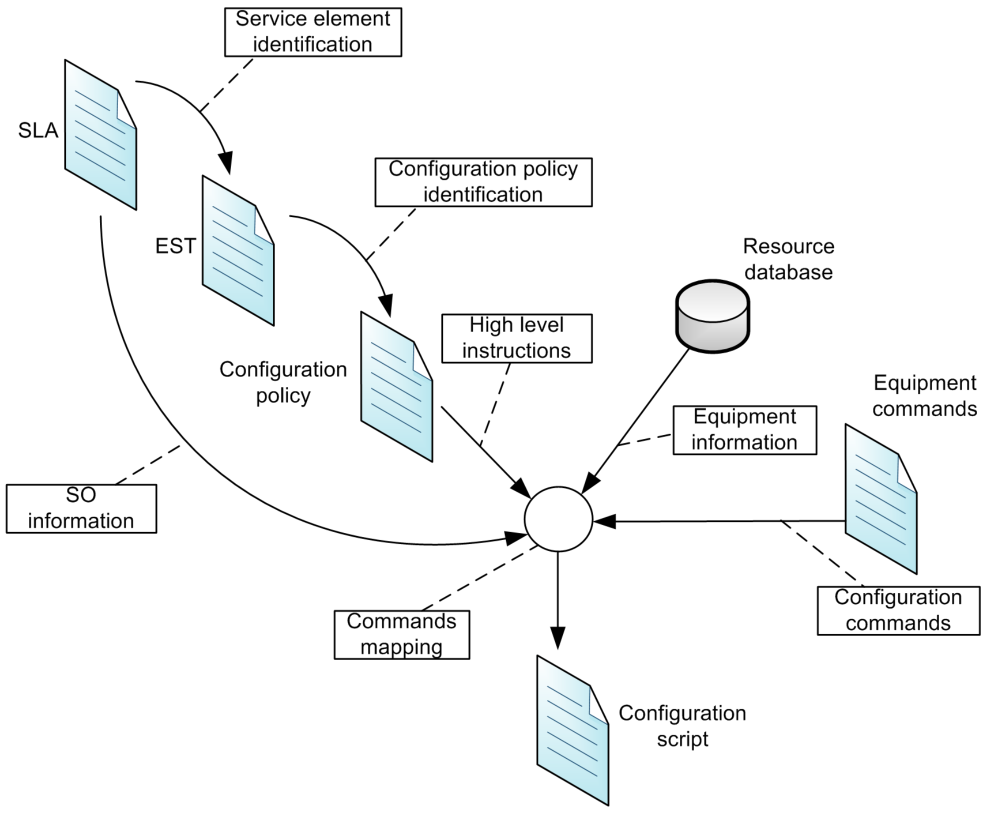


Figure 4.13: Configuration script generation

EOs, since they are part of the federation of providers (see Subsection 3.1.1). Thus, there are associations in the configuration policy between every requirement type and high-level instructions defined by each EO. These private associations allow providers to define their high-level instructions according to their own preferences. For each high-level instruction, there is also a list of associated parameters that are appended to these instructions (line 7).

After all high-level instructions and their associated parameters are defined, the algorithm maps every high level instruction into a configuration command specific from their vendor equipment (lines 9 – 10). This intermediary step (high level instructions) between requirements and configuration commands gives providers a vendor technology flexibility, since they only need to change the mapping between the instructions and the commands in the case they change their equipment technology. Regarding the configuration commands, there are basically three types: (i) independent – commands that do not depend on external values. For instance, the Cisco command to enter the configuration mode of a router (i.e. *config terminal*); (ii) EO dependent – commands that depend on local information, such as the IP addresses and port numbers of

EO's routers; and (iii) SO dependent – commands that depend on remote information received from the SO, such as the IP addresses and AS numbers of neighbor routers. The EO and SO dependent commands have a fixed part and a variable part. The variable part has arguments that receive values from the EO or the SO. Thus, when the commands are defined, the EO substitutes each parameter value, received from the SO or obtained from the EO itself, in the appropriate command argument (line 11) and populates a list of commands (line 12), which represents the configuration script. Finally, the algorithm connects to the appropriate resource (line 13) and executes each configuration command (lines 14 – 15). At the end of this process, the algorithm closes the connection to the resource.

Algorithm 5: Generation of configuration script

```
   Input: SLA, Router, Interface
   // SLA: Service level agreement agreed between SO and EO
   // Router and Interface:  Router and interface of the router that will
      be configured
 1 ElementID ← SLA.obtainElementID();
 2 EST ← obtainEST(ElementID);
 3 ConfigPolicy ← obtainConfigurationPolicy(EST);
 4 RequirementList ← SLA.obtainRequirements();
 5 foreach (Requirement Ri ∈ RequirementList) do
 6     HLinstruction ← ConfigPolicy.mapRequirementIntoInstruction(Ri);
 7     ParameterList ← obtainAssociateParams(HLinstruction, Ri);
 8     addToInstructionList(HLInstruction, ParameterList);
 9 foreach (HighLevelInstruction HLi and ParameterList PLi ∈ InstructionList) do
10     Commands ← mapInstructionIntoCommand(HLi);
11     defineCommandParameters(Commands, PLi);
12     addToCommandList(Commands);
13 Connection ← connectResource(Router, Interface);
14 foreach (Command Ci ∈ CommandList) do
15     Connection.executeCommand(Ci);
16 Connection.close();
```

A simple example of how the mapping process occurs is shown in Table 4.8. In this example, the SO requests a CoE to connect to an equipment from a neighboring EO by giving its IP address and AS number. This request corresponds to the high level instruction to configure a Border Gateway Protocol (BGP) route by using IP address *10.1.2.1* and AS number *60* as parameters. Finally, this high level instruction is mapped into the configuration commands shown in the third column of Table 4.8, by using the IP address and AS number values.

The process of creating the configuration script described above is automatic and

Table 4.8: Mapping requirements into configuration commands

SO requirement	High level instruction and parameters	Configuration commands
• Connect the CoE to IP address 10.1.2.1 and AS number 60	• Create BGP route ◇ IP address = 10.1.2.1 ◇ AS number = 60	• *router bgp 1* • *neighbor 10.1.2.1 remote-as 60* • *neighbor 10.1.2.1 activate* • *neighbor 10.1.2.1 as-override*

on-demand, and is carried out in accordance with the policies of each EO. This autonomy has three main advantages:

- Security: Other providers do not influence the configuration of a provider's equipment. The only external information the EO uses when configuring its equipment is the service parameter values previously agreed on with the SO and the addresses of the neighbor equipment;
- QoS model autonomy: the EOs are free to use the QoS model (DiffServ, IntServ, MPLS) configuration they prefer inside their core networks. The only obligation is to deliver the traffic with the QoS requirements (as stipulated in the SLAs)o their connecting domains on the path;
- Vendor technology autonomy: Each EO configures its resources according to its vendor equipment technology (e.g. programming languages, operating systems, etc.).

4.6 QoS Adaptation in Inter-Domain Services

In real inter-domain provisioning scenarios, there is a chance that the service will be interrupted as a result of infrastructure malfunctions, contract violations, communication problems, etc. However, this is an unsatisfactory situation since it may cause customer frustration and eventually a loss of market share. Creating mechanisms that assure the service execution is maintained, in spite of adverse circumstances, is a valuable undertaking. By keeping a high level of customer satisfaction, providers can increase their incomes, since a contented customer does not change his/her provider and can act as a free advertising agent.

The first step towards providing a satisfactory solution to this problem is to reconfigure the service, by substituting one or more components that are part of the service. However, reconfiguration is a time-consuming process, which may require considerable restructuring on the part of providers [Lin 09]. In such situations, it may be necessary to replace all the providers in the service provisioning path, which can generate as much overhead as a new service provisioning request. Thus, QoS adaptation is preferable to service reconfiguration and it is an important feature that must be included in approaches to inter-domain service management. Additionally, since business requirements have become an important factor in advanced service provisioning [Rubio-Loyola 10], the adaptation process should also be flexible enough to respond to these business requirements and not just to technical issues.

In view of this, QIDS employs a QoS adaptation approach to inter-domain service management scenarios. This approach employs a weak adaptation [Salehie 09], in which providers can adapt the QoS of a service by reconfiguring one or more service elements in the provisioning path, so the service can meet new QoS requirements. It allows providers to determine the new service parameters, renegotiate these parameters with other providers along the provisioning path, update their contracts and enforce the changes in the equipment configuration in an automatic and on-demand fashion. Moreover, by adopting this approach, it is possible not only to adapt the QoS of the services to address technical problems but also to meet business-related requests. For instance, owing to financial pressures, a customer may request a downgrade of his/her service or a provider may request an adaptation in a VoIP service to give priority to a video service. Another advantage is that if a provider configuration in the provisioning path already complies with the new requirements, this provider will not be affected by the adaptation thus decreasing the processing time of the whole adaptation process.

4.6.1 Adaptation Mechanism

The QoS adaptation process allows providers to perform modifications in inter-domain QoS-aware services that are being provisioned. These modifications aim to maintain the service features as agreed during the composition process (and ratified by the SLAs) or to satisfy a request from one of the players of the provisioning process. This process has some similarities to service establishment. The main difference is that the former neither needs to search for available service elements nor to compose the service path, since the service elements that compose the service remain the same. If the SO detects that the same path is not able to fulfill the adaptation request, it has to search for new service elements and calculate a new service path, which is a reconfiguration process.

After all the players of the provisioning process (customer, SO and EOs) have reached an agreement and the service is being executed, it is necessary to use a monitoring mechanism to check if the service complies with the agreed requirements, although it is beyond the scope of this research study to describe this. Thus, the monitoring rests on the assumption that the SO contracts a trusted third party entity from the federation of providers which monitors the service (Figure 4.14), which is a well-accepted solution [Djarallah 09, Kim 08, Jacobs 05]. In this case, Monitor Agents (MAs), which are located in the edge devices of the providers, monitor the traffic between the service elements, by using the information embedded in an SLA that has been signed by both the SO and the monitoring entity. Since the monitoring entity is from the federation of providers, and consequently, has a trust relationship with the providers, it is acceptable for them to allow the MAs to reside in their edge devices. Furthermore, the MA is aware (by means of the signed SLA) of the parameter thresholds that it must check for a specific service and is responsible for sending alert messages to the SO to inform about the service status. These alert messages are sent in a time interval that is specified by the SLA. Moreover, the MA also sends alert messages to notify the SO about any service violations that may eventually occur, which can cause a service adaptation.

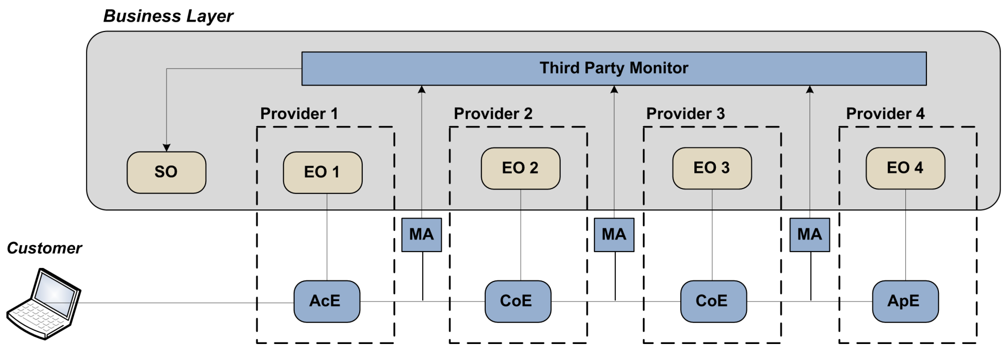


Figure 4.14: Service monitoring

The service adaptation can be triggered by three different situations:

- Violation messages from MAs: Third party agents contracted to monitor the service can send messages to the SO if they detect SLA violations;

- Requests from customers: A customer may request adaptations of the service in accordance with his/her needs. For instance, a service level downgrade may be requested as a result of financial constraints;

- Requests from EOs that are providing service elements: A EO may request an adaptation due to technical problems (e.g. heavy traffic, equipment malfunctions, etc.) or the intensity of business demands.

To better understand the process triggered by the situations mentioned above, Figure 4.15 presents the sequence diagram with the activities that are performed during the service adaptation.

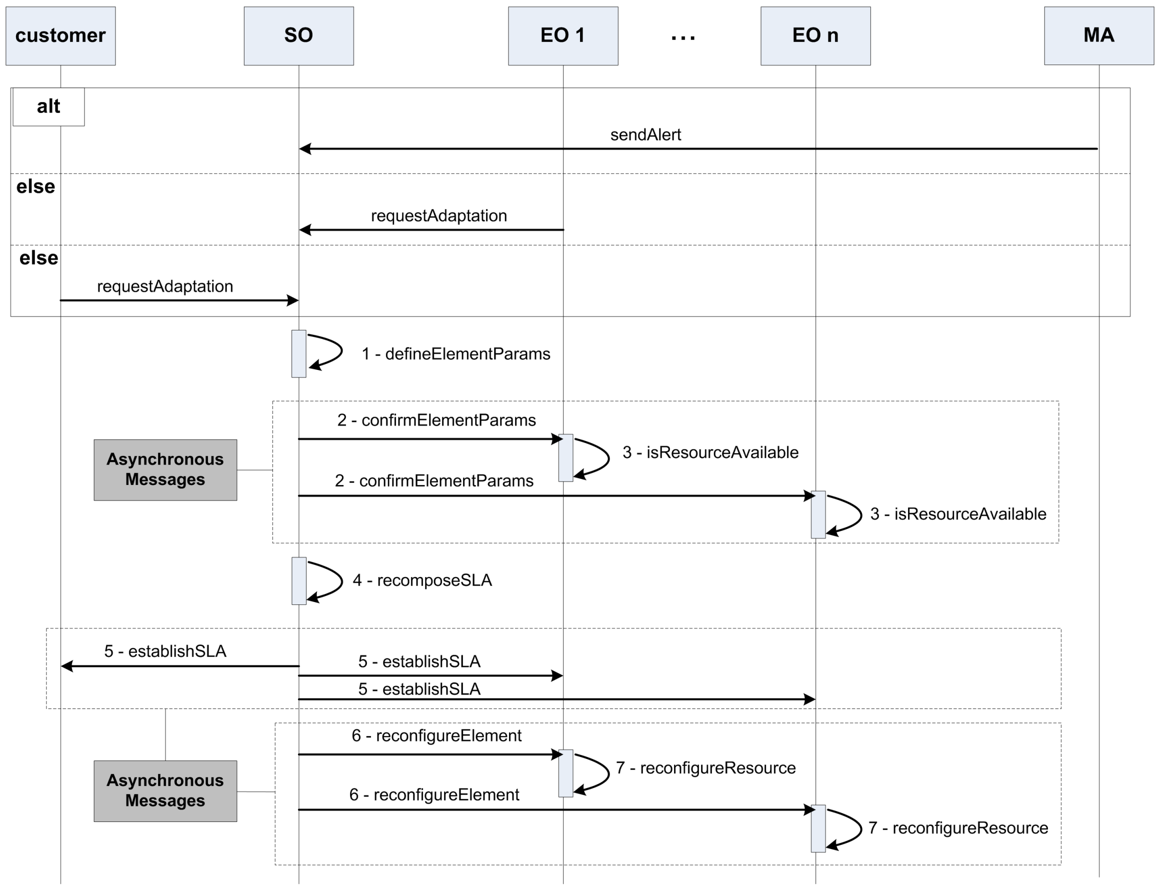


Figure 4.15: Service adaptation request sequence diagram

By using the SLA identification included in the adaptation request (sent by the customer or the EO) or in the alert message (sent by the MA), the SO is able to identify the service that must be adapted. After the SO has identified the service, it must define the new parameters of the service elements that compose the service (*defineElementParams*). For instance, a customer may request a QoS level upgrading in his/her video streaming service from Silver to Premium class. The SO then searches in its service policies for the new parameter values that the Premium video service must guarantee (e.g. bandwidth of 10 Mb/s instead of 5 Mb/s).

Once the SO has determined the new parameter values, it sends confirmation messages to the EOs (*confirmElementParams*) to state whether they can guarantee the new values or not, since these new values may be different from those previously contracted. It should be pointed out that the SO may not send confirmation messages to some EOs if the new parameter values still fit in with the current configuration of the service elements of these EOs. In view of the example above of video streaming upgrading, if the CoEs are already configured to guarantee a maximum bandwidth of 10 Mb/s, there is no need to adapt these CoEs.

After the EOs have received the confirmation message from the SO, they check to decide if they can guarantee the new parameter values (*isResourceAvailable*). If so, they reserve their resources and send an affirmative response to the confirmation message. If all the EOs respond affirmatively, the SO recomposes the SLAs (*recomposeSLA*) which were previously signed during the composition process. In this stage, all the agreements between the SO and the EOs that are affected by the service adaptation are reformulated, as well as, the agreement between the SO and the customer. Obviously, the penalties associated with the adaptation process, as defined in the SLAs, such as financial compensations, are imposed. However, although billing issues are an important factor that must be handled in inter-domain provisioning, they are beyond the scope of this work.

When the new SLAs have been finalized, the SO sends them to the customer and to all the EOs that were affected by the adaptation request (*establishSLA*). At this moment, all the players that have received the new SLAs (customer and adapted EOs) must decide whether they agree with the SLA terms or not. If they do, the SO sends a reconfiguration request (*reconfigureElement*) to the EOs. The EOs then create and enforce a configuration script with the new parameter values into their resources (*reconfigureResource*). At the end of this process, the adapted service is ready to continue its execution.

Algorithm 6 depicts the adaptation process performed by QIDS. In the first nine lines of the algorithm, there is a sequence of commands executed to obtain information about the new requirements the service must support. These commands are as follows:

Line 1 – SO obtains the SLA identification from the adaptation request message; Line 2 – SO obtains the SLA; Line 3 – SO obtains the service identification specified in the SLA; Line 4 – SO obtains the service order instance associated with the service identification; Line 5 – SO obtains the type of service; Line 6 – SO obtains the service policy of that type of service; Line 7 – SO obtains the requirements of the adaptation; Line 8 – SO obtains the new parameter values that the EOs have to support; Line 9 – SO obtains the list of service elements that compose the service.

Algorithm 6: QoS adaptation

```
   Input: AdaptationRequest
   // AdaptationRequest:  A request that triggers an adaptation process
 1 SLA_ID ← AdaptationRequest.getSLA_ID();
 2 SLA ← obtainSLA(SLA_ID);
 3 ServiceID ← SLA.getServiceID();
 4 ServiceOrder ← obtainServiceOrder(ServiceID);
 5 ServiceType ← ServiceOrder.getServiceType;
 6 ServicePolicy ← obtainServicePolicy(ServiceType);
 7 RequirementList ← AdaptationRequest.getRequirements();
 8 NewParameterList ← ServicePolicy.defineElementParameters(RequirementList);
 9 ServiceElementList ← ServiceOrder.getServiceElements;
10 foreach (ServiceElement SEi ∈ ServiceElementList) do
11     ParameterList ← SEi.getParameters();
12     Assure ← compare(ParameterList, NewParameterList);
13     if (not Assure) then
14         EO ← obtainEO(SEi);
15         confirmElementParameters(EO, NewParameterList);
16         addToAdaptedEOList(EO);
17 foreach (ElementOwner EOi ∈ AdaptedList) do
18     EO_SLA ← obtainSLA(ServiceOrder, EOi);
19     NewSLA ← recomposeSLA(EO_SLA);
20     addToNewSLAList(NewSLA);
21 foreach (ElementOwner EOi ∈ AdaptedList and NewSLA SLAi ∈ NewSLAList) do
22     establishSLA(EOi, SLAi);
23 foreach (ElementOwner EOi ∈ AdaptedList) do
24     reconfigureElement(EOi, ServiceOrderID);
```

When the list of service elements is found, the SO compares the parameter values of each of them with the new parameter values requested by the adaptation (lines 10 – 12). If the service element does not support the new requirements, the SO sends a confirmation message to the EO that provides the service element and adds this EO to a list of adapted EOs (lines 13 – 16). If all the EOs respond to the confirmation message affirmatively, the SO recomposes the SLAs with regard to each EO and adds these recomposed SLAs in a new SLA list (lines 17 – 20). Finally, the SO establishes these recomposed SLAs and requests the EOs to reconfigure their service elements (lines 21 – 24).

Figure 4.16 summarizes the adaptation process by showing an example of a service upgrade request. Suppose that a customer requests the upgrading of his/her video streaming service from Silver to Premium class. Each service class has its own set of parameter values (for the sake of simplicity, only four parameters are considered here). There are four service elements that interact to provide the service: AcE1, CoE1, CoE2, and ApE1. Each pattern in the small rectangles represents the current configuration of the service element. As can be seen in the figure, AcE1 does not support the new jitter and bandwidth values, and ApE1 does not support the new frame rate value; hence they need to be adapted. On the other hand, both the CoE1 and CoE2 support the Premium service requirements, which prevents them from being affected by the adaptation process.

The set of operations referred to above, enables providers to adapt QoS-aware services in an automatic and dynamic fashion. This feature enhances the management of inter-domain services, since providers do not need to manually interfere in the service provisioning every time an adaptation request is performed, and thus considerably reduces the operational times. Moreover, depending on the adaptation that is required, only a subset of the EOs that compose the service need to adapt their service elements, which also decreases the operational times. QIDS also handles adaptation requests related to business issues and not only when technical problems arise.

4.7 Summary

This chapter includes a description of the QoS for Inter-Domain Services (QIDS) model, which was integrated with GBF, and shows how it can enhance the inter-domain service management by handling several processes in an automatic, on-demand and dynamic fashion. Initially, QIDS allows a provider to find an end-to-end service path that can meet the performance and service-specific requirements of the customer and, at the same time, is in accordance with the business expectations of the providers. This end-to-end service path is composed of service elements from other providers. To compose this path, QIDS analyses and compares the parameter values of the service

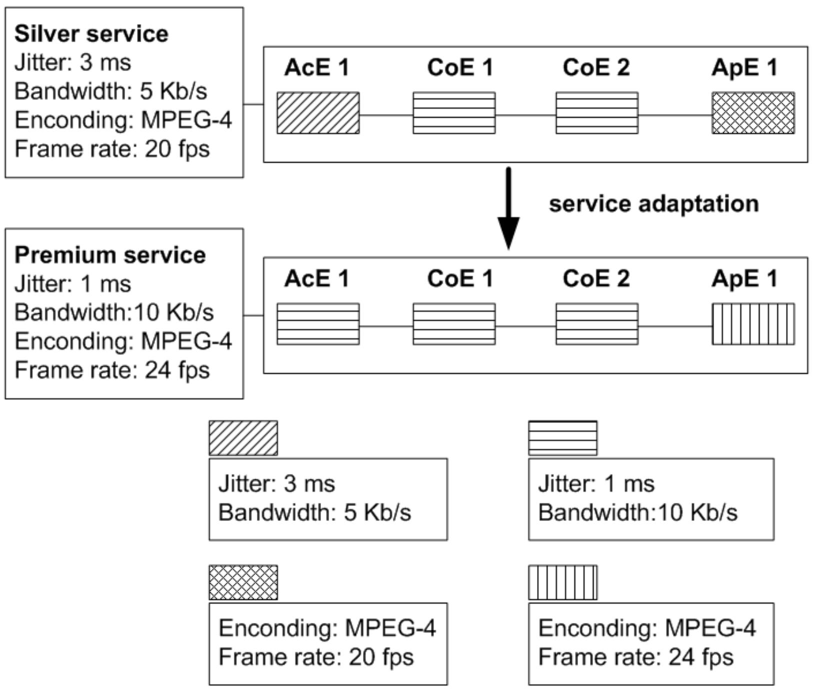


Figure 4.16: Service adaptation representation (service upgrade)

elements with each other on an individual basis, thus creating suitable end-to-end service paths that meet all these criteria.

QIDS adds flexibility to the provisioning process by allowing providers to use their own service policies to manage the inter-domain service. Moreover, it also enables providers to define the service classes according to these policies. This autonomy ensures that providers do not need to adjust their business strategies simply to comply with a standardization of service classes.

In addition to the discovery of the end-to-end path, other factors need to be taken into account to guarantee the correct provisioning of the inter-domain service, such as resource reservation, SLA establishment, and resource configuration. For this reason, QIDS employs a reservation in advance scheme to guarantee that the resources are reserved for the requested service. It also permits the creation and establishment of SLAs between the parties involved in the provisioning process. Moreover, it supports the generation of configuration scripts to be enforced in the reserved resources. This generation conforms to the providers' own policies, thus guaranteeing that the providers can use the QoS model they want inside their core networks.

Finally, QIDS also implements a QoS adaptation approach that allows providers to adapt the QoS of services that are being provisioned. As a result, if a service violates an

SLA agreement, it can be adapted in order to guarantee its continuity. This approach allows adaptation requests based on business issues to be handled as well.

Chapter 5

Prototype Development and Evaluation Results

In this chapter, there is an examination of the implementation issues arising from a prototype developed to validate QoS for Inter-Domain Services (QIDS). This prototype implements the QIDS functions described in the previous chapter. Additionally, there are some observations about the Global Business Framework (GBF) implementation, which supports the QIDS functions. Following this, the study focuses on a use case implemented in the prototype, which consists of establishing inter-domain Virtual Private Networks (VPNs) for video streaming services. Evaluation tests were undertaken in testbed scenarios and the results are discussed and analyzed.

This chapter is structured as follows: Section 5.1 describes the use case and the testbed environment used for the experiments. Section 5.2 discusses the QIDS prototype and the tools used to support its implementation and execution. Section 5.3 details the tests carried out in the prototype and analyses the results of the evaluation study. Finally, Section 5.4 contains a summary of the conclusions drawn from the evaluation undertaken in this chapter.

5.1 Use Case Scenario and Testbed Environment

The use case scenario chosen to validate QIDS involved establishing an inter-domain Border Gateway Protocol (BGP)/Multiprotocol Label Switching (MPLS) VPN

[Rosen 06] to support video streaming with QoS requirements. This scenario was mainly chosen for three reasons:

The use case scenario chosen to validate QIDS involved establishing an inter-domain Border Gateway Protocol (BGP)/Multiprotocol Label Switching (MPLS) VPN [Rosen 06] to support video streaming with QoS requirements. This scenario was mainly chosen for three reasons: (i) Inter-domain VPNs are services that are often offered by providers to their customers; (ii) The establishment of a VPN is an awkward task that involves human intervention and manual configuration; this leads to long-term VPNs, which makes it difficult to offer applications that require more granular, "immediately available" and short-term VPNs; and (iii) VPNs are a convenient means of supporting QoS in inter-domain scenarios, since they are able to separate the transmitted traffic.

In BGP/MPLS VPNs, BGP is used to distribute VPN routing information across the providers' networks and MPLS is used to forward VPN traffic from one VPN endpoint to another. Figure 5.1 illustrates a BGP/MPLS VPN architecture and shows its main elements. According to RFC 4026 [Andersson 05] and RFC 4364 [Rosen 06], a BGP/MPLS VPN architecture has four types of routers: – Customer Edge (CE) is a router that resides at the edge of the customer site; – Provider Edge (PE) is a router that resides at the edge of the provider network and connects directly to CE routers; – Provider router (P) resides within the provider's network and does not connect to CE routers and; – Autonomous System Border Router (ASBR) resides at the edge of the provider's network and connects to other ASBR routers. ASBRs are border routers that connect different provider networks.

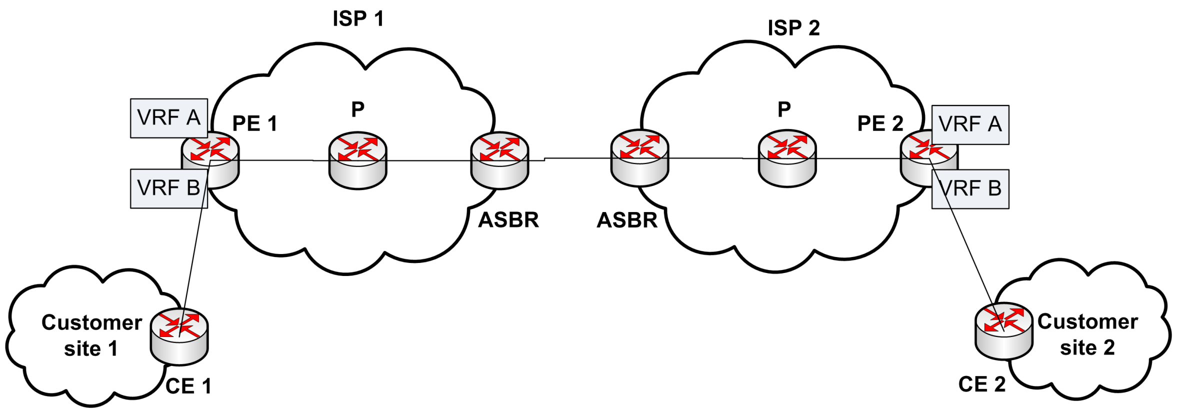


Figure 5.1: BGP/MPLS VPN architecture

As well as the regular IP routing table, each PE may have zero or more Virtual Routing and Forwarding (VRF) tables. Each VPN is associated with a VRF, which maintains the customer's routing information. Thus, the traffic for each VPN is kept separated from the rest. In BGP/MPLS VPN scenarios, when a customer wants to connect to another customer, a VPN must be established between the PEs that are connected to these customer sites. Considering the topology example in Figure 5.1, a customer from site 1 may request a connection to a customer in site 2. To effect this, CE1 announces its route to PE1 by informing a regular IPV4 address. PE1 then transforms this IPV4 address into a VPN-IPV4 address, prepends a MPLS label and associates the traffic from CE1 with a VRF (for instance, VRF A). The VPN-IPV4 address is created by adding a route distinguisher (RD) to the IPV4 address. The MPLS label is used to forward the VPN traffic inside the providers' networks. Through using the labels, the P routers do not have to examine the IP address or headers of the transmitted packets. They just use the labels to forward the packets. When PE2 receives an advertised route, it also associates the route with VRF A, transforms the VPN-IPV4 address back into a IPV4 address and advertises this address to CE1, thus completing the route establishment.

Following the principles of BGP/MPLS VPNs, the basic testbed topology used to test QIDS is presented in Figure 5.2. The topology consists of two transit domains (CoEs) and two endpoint domains (AcE and ApE) and each domain has an identification (bt.uk, ft.fr, pt.pt and dt.de). In fact, this topology was generated by making use of the example discussed in Subsection 4.4.1.2. It illustrates a real scenario, since it was verified that most of the inter-domain traffic (80% – 90%) exchanged by an ISP only travels a few Autonomous System (AS) hops away (two to four hops) [Quoitin 03, Pujol 05]. The testbed was deployed in six machines. One machine hosts the Universal Description, Discovery, and Integration (UDDI), another machine hosts the Service Owner (SO) while the remaining machines host the Element Owners (EOs) (each machine hosts one EO). QIDS and GBF are deployed in the Tomcat/Axis instances of each machine that hosts the SO and the EOs. The machines that host the EOs have also had the Dynamips/Dynagen installed to emulate the routers that support the BGP/MPLS VPN.

It is necessary to configure VRF tables in each PE router to establish a BGP/MPLS VPN. By using a VRF it is possible to assign a unique value for a customer's VPN. This means that the users at a specific VPN cannot inspect packet traffics outside this VPN. To configure a BGP/MPLS VPN, the following information is required:

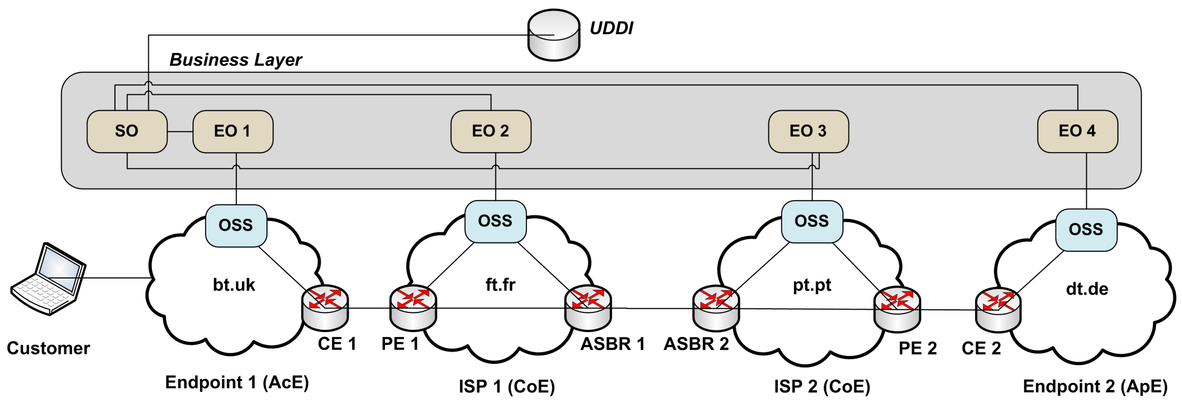


Figure 5.2: Inter-domain topology

- VRF name: The unique name that represents a VRF. Every PE router in the VPN must use the same VRF name;
- Route Distinguisher (RD): A 16-bit or 32-bit number that helps to identify the VPN. Every PE router in the VPN must use the same RD;
- Interface or sub-interface IP address and Autonomous System (AS) number of each endpoint: Used to forward the traffic to the correct interface (or sub-interface) address on PE-CE connections; and
- Interface or sub-interface IP address and AS number of each ASBR router: Used to forward the traffic to the correct interface (or sub-interface) address on ASBR-ASBR connections.

The VRF name and RD are automatically generated by the SO, since they must be the same for every PE in the VPN. The interface (or sub-interface) IP address and AS number of CE, PE and ASBR routers are acquired from the templates of each service element. It is worth mentioning that, since these IP addresses are from edge routers (public addresses), they are not sensitive information that providers would be reluctant to share.

The configuration of the BGP/MPLS VPN is performed only on CoEs. Figure 5.3 shows a fragment of the template that the SO sends to each CoE (in this case, the ft.fr domain of the topology). In addition to the VRF name, RD, interface IP address and AS number of each edge router, it also contains a unique identification for the service, the identification of the service element, the neighbor domain, and the alert URL, which is used to call the alert Web Service on the SO. This information is necessary in case the EO needs to send alert messages to the SO, due to some service element problem.

```
<ElementParameters>
   <ServiceId>280</ServiceId>                                      <!--   Service identification                      -->
   <ElementSOId>119</ElementSOId>                                  <!--   Element identification                      -->
   <VRF>vpnred223</VRF>                                            <!--   VRF name                                    -->
   <RouterDistinguisher>100:100223</RouterDistinguisher>           <!--   32-bit RD number                            -->
   <CE-ASNumber>65200</CE-ASNumber>                                <!--   Neighbor endpoint AS number                 -->
   <CE-Domain>bt.uk</CE-Domain>                                    <!--   Neighbor endpoint domain                    -->
   <CE-IP>10.1.2.1</CE-IP>                                         <!--   Neighbor endpoint IP address                -->
   <ASBR-ASNumber>2</ASBR-ASNumber>                                <!--   AS number of the neighbor transit domain    -->
   <ASBR-Domain>pt.pt</ASBR-Domain>                                <!--   Domain of the neighbor transit domain       -->
   <ASBR-IP>10.4.4.10</ASBR-IP>                                    <!--   IP address of the neighbor transit domain   -->
   <AlertURL>urn:alertservice;alert_service;AlertServiceService;   <!--                                               -->
                  receiveAlert;http://10.3.1.81:8080/axis/services/ <!--  Alert URL (alert Web Service address         -->
                  alert_service?wsdl                                <!--                 in the SO)                    -->
   </AlertURL>
</ElementParameters>
```

Figure 5.3: Information on the service element configuration

When the EO receives the configuration request from the SO, it creates the configuration script and connects to Dynamips through Telnet to enforce the configuration. Table 5.1 presents the commands executed by the EO to configure the BGP/MPLS VPN on a PE router (using IOS Cisco software syntax) in accordance with the information provided in Figure 5.3.

The generated script also contains the required QoS configuration. In the sample scenario, it is assumed that providers will use a simple DiffServ-based configuration. Basically, to forward the traffic in compliance with the QoS requirements, an access list is created to perform packet filtering and then the traffic from a specific source (interface or sub-interface) is associated with this access list. After this, a traffic class is created and associated with the traffic filtered by the access list. A QoS policy is created, which can associate a traffic class with one or more QoS actions. Finally, the QoS policy is associated with the interface (or sub-interface). Table 5.2 shows the QoS configuration enforced on a PE router. This configuration guarantees that all traffic that comes from the CE address is forwarded with a bandwidth of 512 Kb/s.

The configurations required for AcE2 and ApE1 are much simpler. In the ApE1, the EO must send the video traffic with the appropriate QoS characteristics to the interface (or sub-interface) connected to PE2, while at the AcE2, the EO must redirect the traffic received from PE1 to the interface (or sub-interface), connected to the customer that supports the QoS characteristics. At the end of the configuration of the service elements, the service can be provisioned.

Table 5.1: BGP/MPLS VPN configuration

Cisco IOS command	Description
ip vrf vpnred223	Enables the VRF configuration, defining a VPN instance with a VRF name (vpnred223)
rd 100:100223	Creates a routing and forwarding table for the specified VRF
route-target both 100:100223	Creates lists of import and export route-target extended communities for the specified VRF
interface Serial1/1.120 multipoint	Enter the sub-interface configuration mode
ip vrf forwarding vpnred223	Associates the VRF instance with the sub-interface
ip address 10.10.2.2 255.255.255.0	The sub-interface IP address is reconfigured. (Due to the VRF association, the sub-interface loses its IP address)
router bgp 1	Enter the BGP configuration mode
address-family ipv4 vrf vpnred223	Enter the address family configuration mode to configure routing sessions between PE and CE
neighbor 10.1.2.1 remote-as 65200	Adds an entry to the BGP table
neighbor 10.1.2.1 activate	Enables exchange of information with a BGP neighboring router
neighbor 10.1.2.1 as-override	This command is used to identify the site where a route originated, preventing routing loops between routers within the VPN

5.2 Prototype development

To validate GBF and QIDS and carry out the use case described in the previous section, a prototype was implemented using Java 1.5. Communication between the SO and EOs was performed by means of Web Service technologies. Apache Tomcat 5.5.26 [Tomcat] and Apache Axis 1.4 [AXIS] were used to support the implementation and execution of these Web Services. Tomcat was used as the application server that hosts the Web Services, while Axis was used as an implementation of SOAP, which

Table 5.2: QoS configuration

Cisco IOS command	Description
ip access-list extended vpnacl1	Creates an access list to perform packet filtering
permit ip 10.1.2.1 0.0.0.0 any	Associates the traffic from source 10.1.2.1 with the access list
class-map match-any vpnbasic	Creates traffic class
match access-group name vpnacl1	Associates the access list with the created class
policy-map QoS	Creates a QoS policy
class vpnbasic	Associates the traffic class with the created QoS policy
priority 512	Guarantees the specified bandwidth (in Kb/s) for the traffic class
interface Serial1/1.110 multipoint	Enters the sub-interface configuration mode
service-policy output QoS	Associates the QoS policy with the sub-interface

allows the development of the Web Services. Apache jUDDI 2.0rc5 [JUDDI] was used as the implementation of the UDDI and UDDI4J [UDDI4J] as the API to access the UDDI. The relational database was created in MySQL 5.0 [MySQL]. The JDOM API [JDOM] was used to support the handling of the XML documents, while the WSDL4J API [WSDL] was used to support the creation, representation and manipulation of the WSDL documents. The JGraphT [JGraphT] was the API used to manipulate the graph that was built in the service composition process.

The Business Layer (BL) of GBF was implemented as an abstract layer, where entities from different domains make Web Service calls to each other (see Figure 3.3 in Chapter 3). Although security concerns are of importance to multi-domain systems, this issue is beyond the scope of this thesis. Nonetheless, it is possible to devise mechanisms that can guarantee the security of the messages exchanged in the BL, such as the Web Services Security (WS-Security) developed by OASIS [OASIS], which ensures the integrity and confidentiality of SOAP messages.

A basic Policy/OSS Layer was developed. Together with the Policy Repository and the Resource Database, this layer contains simple Java classes to support the interface with the BL and the Infrastructure Layer. In real usage scenarios, it is likely that providers will use OSS applications that they have already installed, such as applications to handle connection admission control, billing and monitoring issues.

To test the prototype, the Infrastructure Layer was built by using the Dynamips router emulator [Dynamips] to emulate Cisco 7200 routers. The use of Dynamips makes it possible to create several topologies with interconnected autonomous systems and populate them with Cisco 7200 routers. The Dynagen front-end [Dynagen] was used to access Dynamips. It provides a command-line interface to interact with the routers. To configure the routers, the service requirements are translated into Internetwork Operating System (IOS) commands [IOS].

Figure 5.4 shows the prototype architecture, presenting the communication technology that the components/tools of the three layers use to interact with each other. Simple Java classes were created to act as the Monitoring Agent (MA) and the customer. The MA sends alert messages to the SO in compliance with what is defined in the SLAs, while the customer sends requisition messages to the SO through the Policy/OSS layer. The communication of the components with the UDDI is performed by means of SOAP over standard HTTP requests. Although communication between Business Managers, and between the Business Manager and the MA, is also conducted through SOAP over HTTP, this is encapsulated in the Web Service calls. Simple Java calls are used to allow the QoS Manager to interact with the Business Manager and the Policy/OSS Layer. Nevertheless, these calls can be easily adapted to Remote Method Invocation (RMI) calls if these components can be found in different machines. Telnet sessions are used by Dynamips/Dynagen to accept connections from the Policy/OSS layer. Through these Telnet connections it is possible to enforce the configuration scripts in the equipment. In the Infrastructure layer, the emulated routers exchange route information with each other by means of BGP.

In order to run the prototype, every provider must have a Tomcat and an Axis instance. The entity (machine, server, provider) that hosts the service directory must have a Tomcat instance as well, and a UDDI implementation. Three data types of the UDDI were used to represent the service and service element offers. The *businessEntity* data type represents the provider of the service (or service element). This structure contains information about the provider itself, such as the name, contact information, address and a list of services offered. A service or service element offer is mapped into

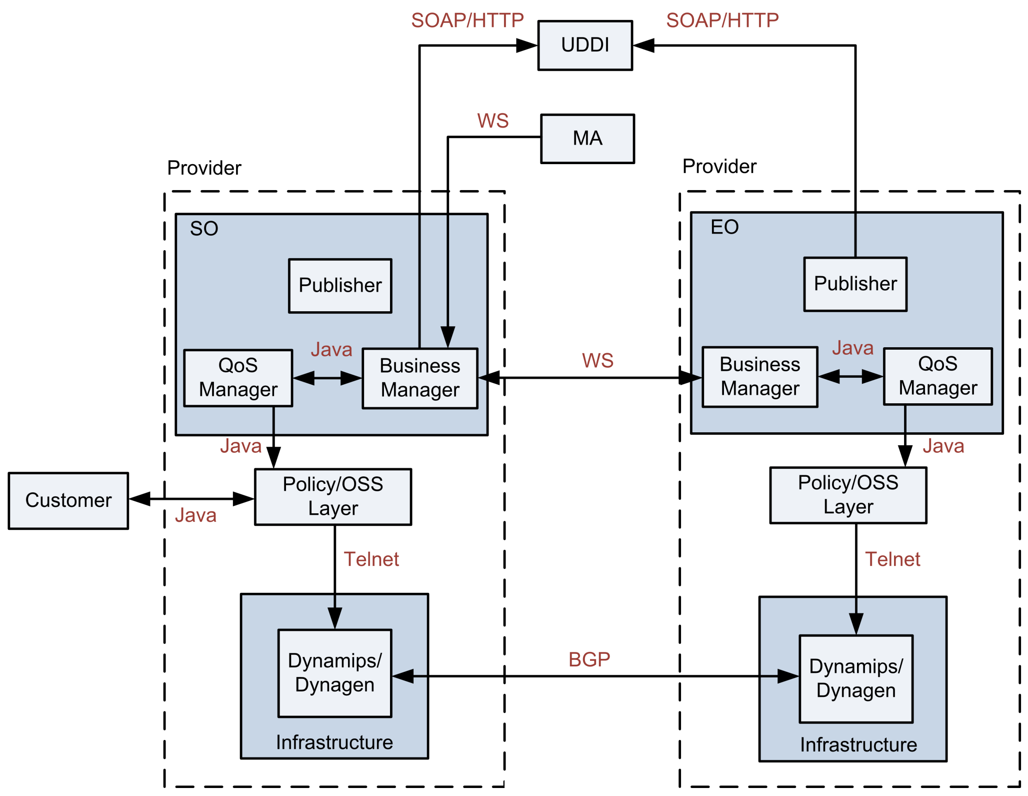


Figure 5.4: Prototype architecture

a *businessService* structure, which represents a logical service. It contains information such as the name and description of the service. The *businessService* can contain one or more *bindingTemplate* structures. A *bindingTemplate* represents the technical information of the service, such as the access point (e.g. URL used to call the service) and the Web Service Definition Language (WSDL) of the service. In the prototype implementation, the templates are stored in the *bindingTemplate* structure. The Publisher component uses the interfaces provided by the UDDI4J to publish, update or withdraw service and service element offers at the UDDI. More information about UDDI data structure can be found in the UDDI API Specification [UDDI 02].

For every service (or service element) offer published in the UDDI, there must be a corresponding Web Service deployed in the Axis of the provider that published the offer. When a provider (SO or EO) needs a service, its Business Manager component obtains the offers from the UDDI, along with the template of the service or service element and the WSDL. By using the information contained in the WSDL, the provider is able to call the corresponding Web Service at the provider where the service or service element is deployed.

During the service composition, an implementation of the BellmanFord algorithm (provided by the JGraphT API) was used to calculate the shortest paths in the graph.

5.3 Evaluation tests

This section examines the tests that were carried out to validate QIDS. These tests were undertaken by using the topology referred to above and aim to verify the overhead in the Business Layer (BL) caused by the message exchange between providers and the performance of the model when composing, establishing and adapting inter-domain services.

These experiments were divided into three groups. The objective of the first group of experiments is to assess the overhead in the BL. Since the BL is used as the communication channel by all the parties (SO, EO, MA and UDDI) involved in the provisioning process, several messages are exchanged during the service life-cycle. The amount of data exchanged between a pair of entities is measured during the main operations that are carried out to manage the services. These operations comprise service establishment, service adaptation, service monitoring, and service shutdown.

The second set of tests verifies the efficiency of QIDS in handling service establishment and service adaptation requests. Initially, they determine the times QIDS takes to handle several service establishment and service adaptation requests. An assessment is also made of the percentage of times each operation takes when a customer makes a request for establishment and adaptation. Afterwards, further tests are carried out by adding more CoEs in the testbed topology, thus allowing scenarios with more transit domains to be simulated. Finally, the efficiency of QIDS in handling service adaptation was evaluated by: (i) measuring the times the SO takes to handle an adaptation as the number of EOs that have to be adapted increases; and (ii) measuring the times an EO takes to handle an adaptation when the number of requesting SOs increases.

In the third group of experiments, the service composition of QIDS is evaluated by increasing the number of service elements that have to be compared. This test seeks to analyse the impact that more service elements (more service options) have on the scalability of the approach used to compose an end-to-end service. A final test is performed to compare the service composition approach used by QIDS, (where performance, service-specific and business parameters are taken into consideration), with a simpler approach that only takes one parameter into account.

5.3.1 Business Layer Overhead

During the inter-domain service management, several messages are exchanged in the BL of the GBF, as shown in Figure 3.3. These messages allow the entities of the management process (SO, EO, MA and UDDI) to communicate with each other. As mentioned earlier, all of the messages exchanged in BL are SOAP messages. Since SOAP messages are based on XML documents, they are larger to transmit and more time-consuming to process than binary-based messages used by other communication protocols, such as RMI and Common Object Request Broker Architecture (CORBA). However, SOAP messages are more extensible and offer greater interoperability, which is essential to facilitate interactions between entities from different domains.

Table 5.3 shows the overhead in BL associated with the main operations performed during the inter-domain service management (Service establishment, service adaptation, service monitoring and service shutdown). The messages that were exchanged to manage the offers at the UDDI (publish, update and withdraw) were not taken into account. These messages are not usually triggered during inter-domain service management operations, and thus do not have a significant impact on the BL overhead. The values shown in Table 5.3 represent the average time of 100 requests. The traffic between the machines (representing different domains) was analysed with the aid of the Wireshark tool (previously known as Ethereal), which is a network protocol analyzer [Wireshark].

Table 5.3: Business Layer overhead – in Packets (Bytes)

Interacting entities	Service establishment	Service adaptation	Service monitoring	Service shutdown
SO / UDDI	304 (93840)	—	—	—
MA / SO	—	—	3 (1873)	—
SO / EO 1	118 (59374)	118 (58994)	—	3 (1902)
SO / EO 2	118 (59289)	118 (59572)	—	3 (1902)
SO / EO 3	118 (59107)	118 (59163)	—	3 (1902)
SO / EO 4	118 (59267)	118 (59247)	—	3 (1902)

During the establishment of one VPN, eight service elements were published in the UDDI. When the SO contacts the UDDI to search for available service elements, it obtains at the same time the eight Element Specification Templates (ESTs) and the eight WSDLs of the Web Services that correspond to each service element, which results in almost 94 KB. Further messages are exchanged between the SO and EOs in order to configure the service, - more precisely, three messages are sent by the SO to each EO, as can be seen in the sequence diagram of Figure 4.6. These messages result in about 60 KB for each SO-EO pair due to the exchange of XML documents, such as the SLAs and configuration templates. In the service adaptation, the overhead in BL caused by the exchange of messages between the SO and each EO is approximately the same as that of the service establishment. The difference is that not every service element may be affected by this adaptation, thus the overhead of about 60 KB for each SO-EO pair, only occurs if the service element of this EO has to be adapted. Another difference is that there is no overhead caused by the search for service elements in the UDDI. During the service monitoring, the MA sends alert messages to the SO on a time-period basis. One message is less than 2 KB, since it only contains the information needed to identify the service that is being monitored and the alert code, which specifies the service status. Finally, the service shutdown operation induces an overhead of only about 3 KB for each SO-EO pair, since just a simple message is sent informing the identification of the service that has to be terminated.

As can be noted from the values in Table 5.3, a higher overhead in BL occurs during the service establishment and the service adaptation due to the exchange of XML documents. A possible solution that can be used to decrease this overhead would be to reduce the name of the tags in the XML documents and/or employ some form of mapping mechanism to translate tag codes into tag names. However, this causes extra processing on the part of the providers. Moreover, this approach also conflicts with XML nature, which allows an easy representation of information.

5.3.2 Service Establishment and Service Adaptation

Tests with an increasing number of simultaneous requests were carried out to determine the efficiency of QIDS in handling more than one request. Each test was conducted ten times, and the values obtained were averaged.

Table 5.4 shows the times needed to handle 1, 10, 50 and 100 simultaneous establishment requests. The times given seem to be very satisfactory, especially when this is compared with the current approach, in which providers use human-based interactions to exchange information and establish contracts, and configure their equipment in a manual fashion (which may take hours). Moreover, from the customer's standpoint, given the fact that he/she requested the QoS-aware service through a web portal, 3.3 seconds is a fairly low waiting time. In most cases, this is faster than the time one has to wait for the web page to load.

Table 5.4: Service establishment requisition times

Request	Times (s)
1	3.3
10	12.3
50	45.8
100	82.7

It should be underlined that the times given here are concerned with QIDS performance in handling requests for service establishment (from the customers request to the resource configuration). In the case of BGP/MPLS VPNs, the total establishment time is much longer (about 2.5 minutes) due to the time the BGP convergence process takes to advertise the new routes. However, if pre-established routes are taken into account, the total establishment time of the QoS-aware services is similar to the times given here, since there is no need for the BGP to advertise the routes. Furthermore, in a real provider scenario with real equipment and carrier-level servers, the results are expected to be much better.

The service adaptation request use case is quite similar to the service establishment request use case. Since the adaptation only requires a modification of some service element or service requirements, the EOs that compose the service remain the same. Thus, there is no need to search for service elements or to compose the service path. Taking into consideration the SO service policies in Table 4.5 and the service elements parameter values in Table 4.6, if a customer requests a slight improvement in the video quality (e.g. frame rate = 30 fps), it can be seen that the service path can still provide the new service requirement. Table 5.5 shows the times needed to handle 1, 10, 50 and 100 simultaneous adaptation requests.

The values given in Table 5.5 show a slight decrease in the times needed to perform the service adaptation when compared to those of the service establishment. As stated earlier, this difference occurs because the SO does not have to either search for available service elements or to compose the service path. In view of the fact that the service path

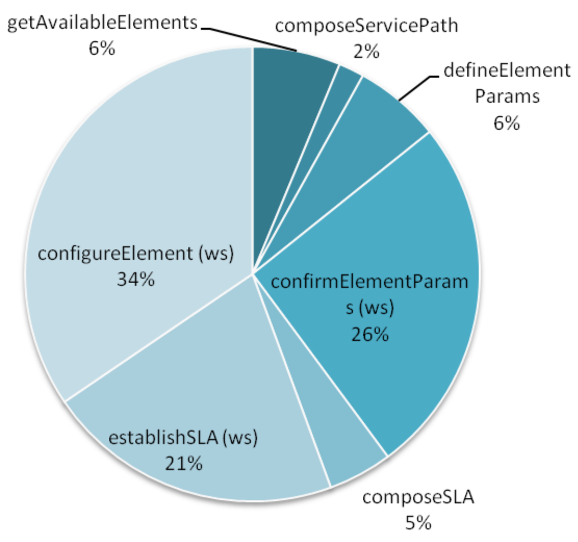
Table 5.5: Service adaptation requisition times

Request	Times (s)
1	2.5
10	8.8
50	35.1
100	66.7

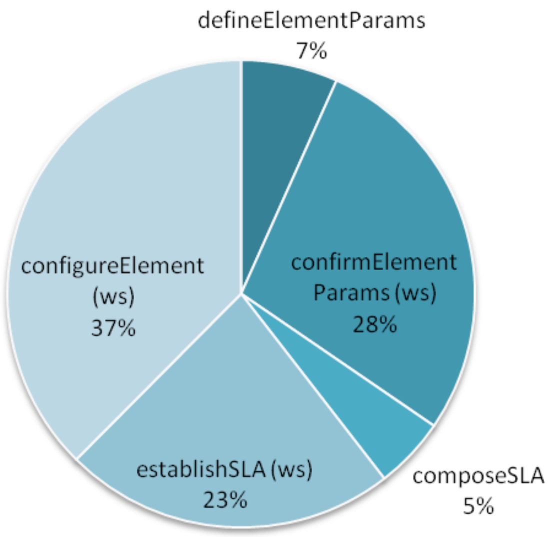
is not affected by the adaptation process, which excludes BGP route advertisements, the required adaptations to the service can be performed in a short period of time.

Figure 5.5 shows the amount of time each major operation takes when a customer makes a request for establishment (Figure 5.5a) and adaptation (Figure 5.5b) of the video streaming service with QoS guarantees. It is apparent from the chart that the overhead of operations executed locally in the SO is much lower than the overhead of operations that contain remote calls, which are performed by the means of Web Services (*confirmElementParams*, *establishSLA*, *configureElement*). The Web Service execution times for the establishment and adaptation of services are about 81% and 88%, respectively. Remote calls perform serialization and deserialization of objects (or data structures) in order to transmit them across the network, which increases the total execution time. In addition, Web Service calls are bandwidth and time expensive, because they are based on SOAP messages, which are textual in nature. However, this drawback does not outweigh the benefit derived from permitting providers to freely design their services by using interface definitions. This is an important advantage, since it allows a loose coupling of services, which facilitates the service composition process (a crucial prerequisite in inter-domain provisioning). Additionally, when comparing the use of these Web Service calls with the manual approach used by providers today, the advantage is obvious: an automated interaction between the providers.

Tests to handle service establishment requests were carried out to evaluate the efficiency of the model. In these tests, the number of intermediary domains (CoEs) that compose the service path increases; thus more CoEs were added to the topology presented in Figure 5.2. Figure 5.6 shows the time needed to handle service establishment requests in scenarios where two, three, four and five CoEs compose the service path. 500 requests were performed for each scenario, and each request was triggered immediately after the previous request had been handled by the QIDS. Each point on the curves represents the average time of 50 successive requests (X axis).



(a) Service establishment



(b) Service adaptation

Figure 5.5: Percentage of operation times

This test aims to verify the performance of the SO when several successive requests are performed.

As can be seen from the graph, the times needed to handle a service establishment request do not vary significantly when more CoEs are added to the configuration topology. The longer times at the beginning of each curve on the graph are due to the initialization of the Java classes. The mean time to handle a service request in the scenario with two CoEs is 3376 ms, while the mean time to handle a service request in the scenario with five CoEs is 3432 ms. If more CoEs are added to the service path, it is expected that the time to handle a service request will increase, since there are more domains to negotiate, establish contracts and configure resources. Nevertheless, since the SO communicates directly with each EO and these interactions are performed by parallel threads, the difference between the mean times is very small. The increase in the time is less than 2%, which shows the efficiency of the QoS model in handling more complex scenarios (paths with more intermediary domains).

Two more tests were carried out to determine the efficiency of QIDS in handling service adaptation requests. The first test seeks to find out the time the SO takes to handle adaptation requests when the number of EOs that are adapted increases. Figure 5.7 shows the times measured in four different scenarios, which are represented by the four curves. The number of EOs affected by the adaptation varies according to the scenarios (from one EO to four EOs). 2,000 successive adaptation requests were

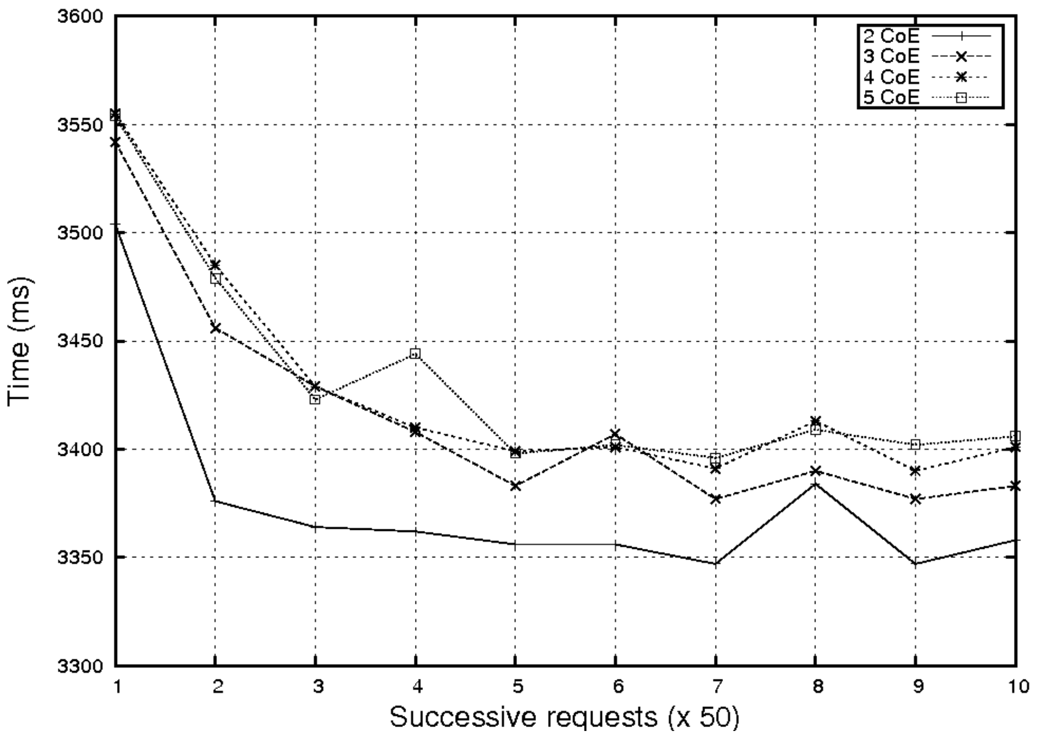


Figure 5.6: Mean time for handling service establishment requests

performed in each scenario. After each 200 successive requests has been made, the average time to handle these requests is measured and plotted through each point in the X axis.

The chart shows that as the number of EOs increases, the time to adapt the QoS of the service increases as well. This is an expected behavior, since if there are more service elements to adapt, then more SLA re-composition and establishment, and reconfiguration of equipment are necessary. However, this increase is not proportional to the increase in the number of EOs. The average times to handle an adaptation request when one, two, three and four EOs are affected are 2285 ms, 2406 ms, 2477 ms and 2540 ms, respectively. These times show the good scalability of the approach, since the difference between the times to adapt one EO and four EOs is only 255 ms, which represents an increase of only about 11%, while the number of EOs increases by a factor of four. This result becomes more significant when considering the fact that most of the inter-domain traffic crosses only two to four hops. In other words, it is unlikely that many more domains will be added to the service path. Moreover, adapting a service in less than three seconds is an extremely low execution time when compared to the time it takes when manual interference is required.

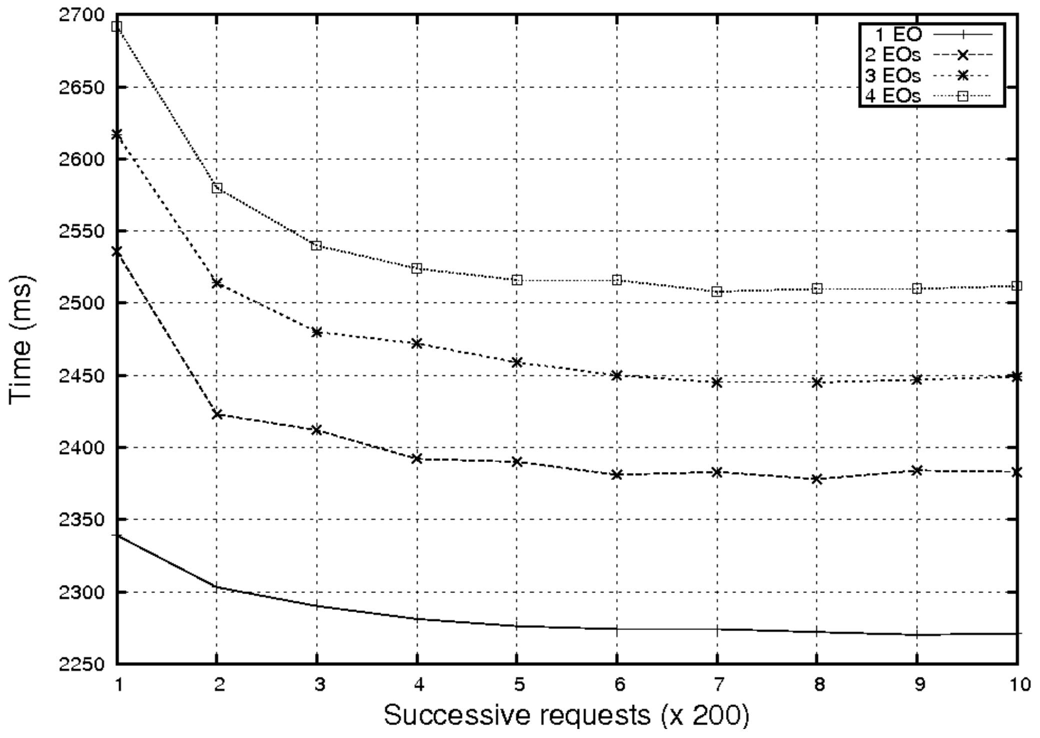


Figure 5.7: Service adaptation times (with an increasing number of EOs)

The second performed test seeks to determine the time an EO takes to handle an adaptation when the number of requesting SOs increases. Four different scenarios were used in this test, with one SO, two SOs, three SOs and four SOs. In each scenario, 2,000 successive adaptation requests were generated by each SO. In the scenarios that had more than one SO, the requests from different SOs were simultaneous. Figure 5.8 shows the results, where each curve represents a scenario. After each 200 successive requests, the average time to handle these requests is measured and plotted along each point in the X axis.

The results in Figure 5.8 show that as the number of SOs performing requests increases, the time an EO takes to handle a request also increases. This is also an anticipated behavior, since if there are more requests, then there are more resources that need to be configured and more time will be spent on communicating with different SOs. Similar to what was observed in the previous test, the increase in the time an EO takes to handle an adaptation request is not proportional to the increase in the number of SOs. The difference between the average times to handle an adaptation when one SO is requesting (964 ms) and when four SOs are requesting (1059 ms) is only 94 ms (an increase of 10%, approximately). This result shows the efficiency resulting from this approach. Moreover, the time taken by the EO to adapt its service

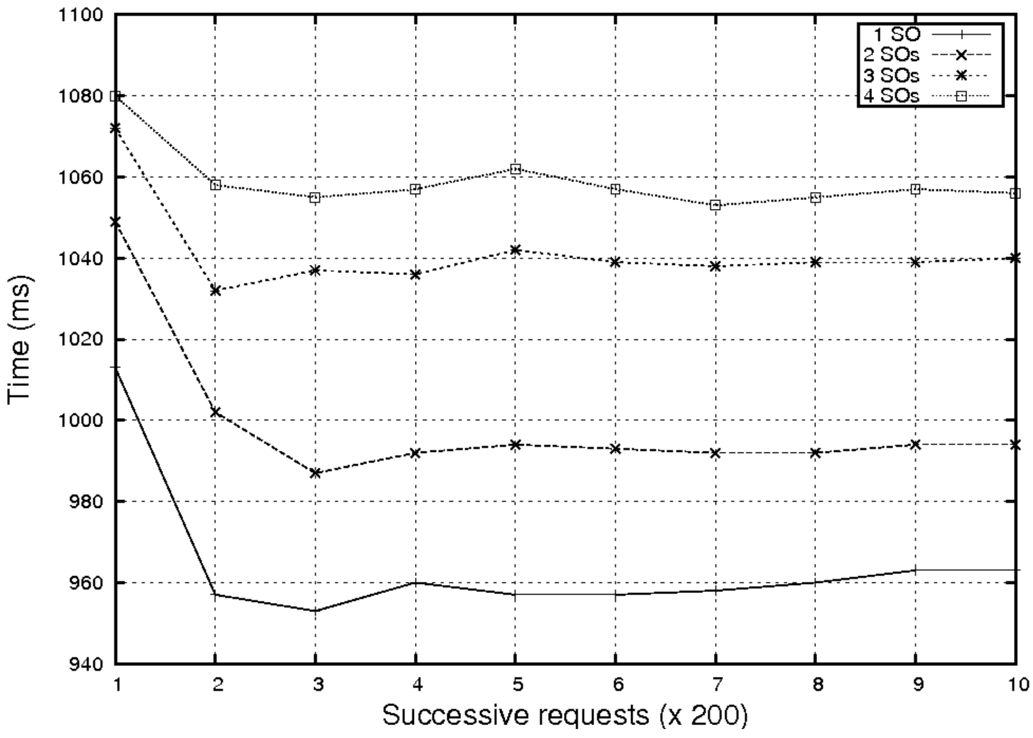


Figure 5.8: Service adaptation times (with an increasing number of SOs)

element, which includes equipment configuration, is extremely low (about 1 second). It is worth pointing out that the times measured in this test only encompass the processing times that occur inside the EOs. For this reason, the times needed for communication between the SOs and EOs have not been computed.

The results of the tests carried out to verify the performance of an EO in handling the service establishment requested by SOs are quite similar to those of the tests shown in the last graph. This is due to the fact that the actions executed by an EO during a service establishment are the same as the actions executed during a service adaptation. Thus, to avoid redundancy, these tests were not included.

It should be stressed that these results are influenced by the use of Dynamips (router emulator). Dynamips consumes a considerable amount of RAM memory for each machine where it is installed. Moreover, according to the manufacturer of Dynamips, it is possible to achieve a performance of about one Kb/s, while in old 7200 Cisco routers 100 Kb/s can be achieved. Thus, in real scenarios with real routers, these results are likely to be much better.

5.3.3 Service Composition

The next experiment assesses the efficiency of QIDS in handling service composition requests when more service elements are compared in the composition process. This test aims to verify the scalability of the composition process. Considering that an EO offers a small set of different service element options of the same type, then there are more EOs to offer the increasing number of service elements. The graph displayed in Figure 5.9 shows the times needed to perform the service composition when the number of service elements increases. 10,000 composition requests were performed, where a new request is triggered immediately after the previous one. Each point in the graph represents the average time of 100 successive requisitions. At the beginning of each curve in the graph, a longer period of time is needed to compose the service, which is explained by the time it takes to load the Java classes in the memory. In this test, it is assumed that there are four scenarios in which the SO has to consider 10, 40, 70 and 100 service elements in order to compose the graph. When the graph is built, the SO calculates the five shortest paths (using the price parameter as the path cost) that can be used as possible end-to-end service paths. The quantity of each service element type is distributed according to the percentages outlined in Table 5.6. The parameter values of the service elements were created by means of a procedure that generates random values for each type of parameter.

Table 5.6: Service element type quantities in each scenario

Service element type	Percentage
AcE	10 %
ApE	20 %
CoE	70 %

As can be seen from Figure 5.9, the time of the service composition process increases as more service elements are taken into account during the composition. The difference between the mean times of the composition when using 10 service elements and using 100 service elements, is about 75 ms. This difference shows the good scalability of the approach, since while the number of service elements increases by a factor of 10, the composition time increases by a factor of only 3 (approximately). Moreover, this difference does not seem to be significant when considering the advantage of having more service element offers (more service options).

Another test was undertaken in the prototype to compare the times needed to establish an inter-domain service between two scenarios: the first scenario employs the

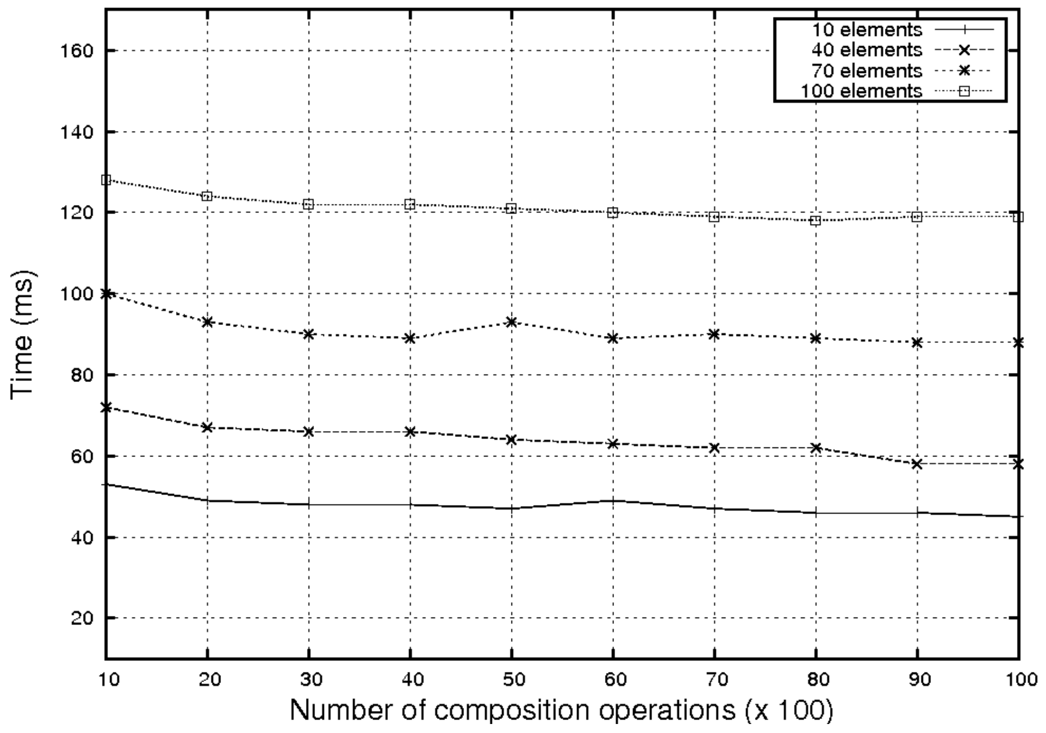


Figure 5.9: Service composition execution times

service composition approach used by QIDS, where the service elements are filtered by comparing the performance, the service-specific and the business parameters; the second scenario uses a simpler service composition approach to establish the same service (the SO composes the graph by using all of the available service elements and chooses the path with the best cost taking into account only one parameter - e.g. price). Table 5.7 provides the comparison between the times and also the relation between the time required to compose and to establish a service in each scenario. 100 establishment requests were performed in both scenarios and the results were averaged.

First of all, as can be seen from the table, the time to compose a service is a small fraction of the total time needed to establish this same service. In other words, the composition process does not significantly affect the time it takes to establish a service. Having said this, it can be seen that there is a slight increase in the time needed to establish an end-to-end service when using the QIDS approach. However, the time it takes to compose the service still remains a small fraction of the total establishment time. Moreover, this increase in the time is outweighed by the advantage of composing an end-to-end path that meets the customer service requirements and achieves the provider's business goals.

Table 5.7: Service composition approach times

	Simpler approach	QIDS approach
Service establishment	3321 ms	3370 ms
Service composition	34 ms	55 ms

5.4 Summary

This chapter has provided an analysis of the prototype implemented to validate QIDS, the use case scenario along with the testbed environment and the evaluation tests that have been carried out.

The implemented prototype encompasses all of the QIDS functionalities, as well as the GBF infrastructure used to allow communication between entities from different domains. A router emulator was used to emulate the equipment configured during the service establishment and the service adaptation processes.

The establishment of inter-domain BGP/MPLS VPNs was the use case chosen to validate the model. The establishment of inter-domain VPNs is a cumbersome task, since it requires human intervention to negotiate and establish the service requirements and manual configuration of equipment. As a result, these types of VPNs are only contracted on a long-term basis. This situation makes the inter-domain VPN an appropriate application to test the automatic, on-demand and dynamic approach of QIDS.

The experiments undertaken in the prototype demonstrated that more than 80% of the total execution time required for the establishment and the adaptation of inter-domain services, is caused by the exchange of SOAP messages (Web Service calls). Despite this large overhead, the use of Web Services allows providers to freely design their services by using interface definitions. Moreover, it also provides more extensibility and flexibility since it involves the use of XML documents, which are the basis of SOAP messages. All these beneficial factors are important when handling with interactions between entities from different domains that have different policies and equipment.

With regard to the performance tests, the results demonstrated that there was a good efficiency in handling service establishment requests. Despite the increase in the number of intermediary domains (two to five), the time to establish an inter-domain service increases by less than 2%. This is a satisfactory result, since most of the inter-domain traffic travels only two to four hops. In other words, it is unlikely that a service provisioning path will comprise more than five domains.

Similar behavior was observed in the experiments carried out to handle service adaptation requests, where QIDS showed a good rate of efficiency on both the SO and EO sides. When more EOs have to be adapted, the average time SO takes to adapt a service only increases by about 11%. On the EO side, when more SOs make a request for adaptation, the average time an EO takes to handle a request increases by only about 10%.

Since the prototype developed in this work uses the Dynamips router emulator, the results are influenced by the performance of this tool. While the old 7200 Cisco routers achieve a performance of 100 Kb/s, Dynamips can only achieve about one Kb/s. Moreover, Dynamips also consumes a considerable amount of RAM memory for each machine where it is installed. In view of this, tests in real scenarios with real equipment are expected to be much better.

In the tests conducted to evaluate the service composition times, it was verified that QIDS behaves well when the number of service elements increases, thus attesting the good scalability of the composition process. While the number of service elements increases by a factor of 10, the average time to perform the service composition process increases by a factor of only 3. Moreover, this increase in the time is outweighed when considering the advantage of having more service options (more service element offers).

Chapter 6

Conclusions and Future Work

This thesis addresses the problem of providing QoS-aware services in multi-provider environments by proposing a QoS model that supports providers in composing, establishing and adapting inter-domain services with QoS guarantees. The concluding chapter outlines the main conclusions of this work and some still unresolved problems that will be the subject of further studies. Section 6.1 provides a summary of the thesis. Section 6.2 describes the main contributions of this work, while Section 6.3 highlights some issues that still need to be addressed in future work.

6.1 Synthesis of the Thesis

The main contribution of this thesis was the proposal of a QoS model called QoS for Inter-Domain Services (QIDS), which can support providers in composing, establishing, and adapting QoS-aware services in inter-domain environments, in an automatic, dynamic and on-demand fashion. QIDS employs mechanisms that enhance the interaction between providers, thus allowing them to achieve a more advanced cooperation to provide inter-domain QoS-aware services. By supporting these value-added services, QIDS is able to benefit both the customer and provider: the customer is not restricted to his/her access provider service portfolio; and providers can increase their market share by offering added-value services to customers that are outside their domains. Moreover, providers can exchange their current human-based interactions and manual configurations for an automated, on-demand and dynamic process. QIDS was deployed in the context of the Global Business Framework (GBF), which is a framework that allows providers to offer and manage inter-domain services by using a Business Layer (BL).

The GBF was developed in collaboration with Alexandre Veloso de Matos, who is a PhD student at the Department of Informatics Engineering (University of Coimbra). GBF, which was discussed in Chapter 3, is a framework that uses a business layer (BL) as a communication channel and adopts an SOA-based approach, where a composite service is created by combining one or more smaller services. Providers use the BL to interact with each other, so that they can publish, search and combine service parts to offer to customers.

QIDS is a QoS model that has been developed in the context of GBF and supports the provisioning of QoS-aware services in multi-domain environments. It was discussed in Chapter 4, where its main functionalities are described. QIDS allows providers to define service classes in accordance with their own preferences. It also supports the composition of inter-domain services by assembling service elements. QIDS analyses the QoS performance parameters, the service-specific parameters and the business parameters of each service element when composing a service. Moreover, QIDS uses a booking scheme to support resource reservation, facilities SLA establishments among providers and allows automatic generation of configuration scripts to be enforced in the underlying equipment. Finally, it enables inter-domain services to be adapted so that they can meet different QoS requirements. Since QIDS does not interfere with the providers' core networks, they can use any QoS model they want inside their networks (e.g.DiffServ, IntServ, MPLS).

Experimental tests were conducted in a prototype that was implemented to validate and evaluate the QIDS contributions. The BGP/MPLS VPN establishment was the use case employed in the tests. Chapter 5 examines the results of these experiments, and shows that the approaches adopted in QIDS fulfill the proposed objectives: automatic, dynamic and on-demand composition, establishment and adaptation of inter-domain QoS-aware services.

6.2 Contributions

This section describes the main contributions made by this thesis.

A business layer-based framework to support inter-domain service provisioning

The first contribution, (which is a collaborative study carried out with Alexandre Veloso de Matos), is the development of a business layer-based framework (GBF) that enables providers to publish, search and compose services. This BL is used as a channel through which providers can communicate with each other by exchanging messages. By means of this framework, providers can use service templates to specify their service characteristics and make business agreements.

Flexible representation of QoS class definitions for inter-domain services

QIDS allows providers to represent the QoS classes of their services in a simple and flexible way by using service templates, which are inspired by some works, such as the TEQUILA project. There is no need to employ a set of standard service classes, since the mapping process is performed by determining the individual values of each parameter of the service element classes. This allows providers to maintain and/or create business strategies that can differentiate their service elements from their competitors and achieve an autonomy which leverages market competition and eventually results in better and cheaper services.

An automatic and dynamic mechanism for composing inter-domain QoS-aware services

QIDS provides a dynamic and automated means of solving the problem of selecting and composing services to meet QoS requirements in inter-domain environments. Algorithms to compose these services are presented. In these algorithms, QIDS does not only take into consideration traditional performance (connectivity) parameters, such as bandwidth and delay, but also service-specific parameters and business parameters. On the basis of local policies, QIDS compares these parameters with the service requirements and is thus able to create the suitable paths needed to provide the service. The adoption of this approach means that the service elements can be combined, regardless of the class of service of each part, since the parameters are compared individually in the composition process.

An automatic and dynamic mechanism for establishing and adapting inter-domain QoS-aware services

QIDS employs algorithms that allow providers to request reservation in advance of resources and generate configuration scripts. The reservation aims to ensure that the resources will meet the service requirements. The mechanism for generating configuration scripts enables high-level service requirements to be translated into configuration commands that can be enforced in the underlying equipment. Since providers translate these requirements in accordance with their local policies, they can use any QoS strategy they want inside their core networks. The creation and establishment of SLA contracts are also supported by QIDS. These SLAs that are agreed between the providers give guarantees that the service offered will comply with specified requirements; otherwise, corrective measures will be taken.

The QoS adaptation of inter-domain services is also supported by QIDS. The QoS adaptation process determines the new QoS parameter values that the service has to meet. These new parameter values are then sent to the providers affected by the adaptation. The providers then reconfigure their resources so that they can continue the service provisioning in response to the new requirements. This is one advantage of this approach, since only the providers who are affected need to reconfigure their equipment. If a provider's equipment configuration already supports the new requirements, there is no need to contact this provider.

Implementation of the QoS model for inter-domain services

A prototype was implemented to validate the QIDS mechanisms. This prototype can perform composition, establishment and adaptation of inter-domain QoS-aware services. It was integrated in the GBF and used the BL as a communication channel and Web Services as the communication technology. Tests carried out in a testbed running of the prototype demonstrated that QIDS has a good efficiency when handling service composition, service establishment and service adaptation requests.

6.3 Future Work

Plans for future work include the study and incorporation of prediction mechanisms, by means of which providers would be able to predict if and when a SLA violation will occur, thus allowing them to take corrective steps before the service deteriorates.

A study of mechanisms to deal with billing issues is also envisaged. This arises from the need to create the means by which providers can charge each other for services and impose financial penalties when SLA violations occur.

This thesis rests on the assumption that a federation of providers will be formed, which means that all providers agree to a standard representation of templates' information, such as employing the same representation for parameter values. However, a more extensive study of the means to represent this information needs to be carried out. For instance, ontology-based semantic annotations could be used as means to create flexible, and yet standardized representations.

In addition, there is also a need for an analysis and application of UDDI extensions to enhance the service discovery functionality, by adopting QoS criteria during the search for service elements in the UDDI. Instead of obtaining all the available service elements, it should be possible to undertake preliminary filtering of the service elements in the UDDI.

Bibliography

[Ahmed 05] T Ahmed, R Boutaba & A Mehaoua. *A measurement-based approach for dynamic QoS adaptation in DiffServ networks.* Computer Communications, vol. 28, no. 18, pages 2020–2033, 2005.

[Andersson 05] L Andersson & T Madsen. *Provider Provisioned Virtual Private Network (VPN) Terminology.* RFC 4026, March 2005.

[Asgari 05] A Asgari, M Boucadair, R Egan, P Morand, D Griffin, J Griem, P Georgatsos, J Spencer, G Pavlou & M Howarth. *Inter-Provider QoS Peering for IP Service Offering Across Multiple Domains.* In 2nd Int. Workshop on Next Generation Networking Middleware (NGNM05), IFIP Networking Conference, Waterloo, Canada, May 2005.

[Atkinson 07] C. Atkinson, P. Bostan, O. Hummel & D. Stoll. *A Practical Approach to Web Service Discovery and Retrieval.* In Web Services, 2007. ICWS 2007. IEEE International Conference on, pages 241–248, 2007.

[AXIS] AXIS. *WebServices - Axis.* http://axis.apache.org/axis/. Last accessed on January 2011.

[Beben 06] A. Beben. *EQ-BGP: an efficient inter-domain QoS routing protocol.* In Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on, volume 2, page 5 pp., 2006.

[Bertin 07] E. Bertin, I. Fodil & N. Crespi. *A business view for NGN service usage.* In Broadband Convergence Networks, 2007. BcN '07. 2nd IEEE/IFIP International Workshop on, pages 1–5, Munich, Germany, April 2007.

[Bertin 09] E. Bertin & N. Crespi. *Service business processes for the next generation of services: a required step to achieve service convergence.* Annals of Telecommunications, vol. 64, no. 3, pages 187–196, April 2009.

[Blake 98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang & W. Weiss. *An Architecture for Differentiated Services.* RFC 2475, December 1998.

[Blum 09a] N. Blum, T. Magedanz & F. Schreiner. *Management of SOA based NGN service exposure, service discovery and service composition.* In Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on, pages 430–437, 2009.

[Blum 09b] N. Blum, T. Magedanz, F. Schreiner & S. Wahle. *Service Oriented Testbed Infrastructures: a Cross-Layer Approach for NGNs.* Mobile Networks and Applications, vol. 15, pages 413–424, August 2009.

[Borcoci 06] E. Borcoci, A. Asgari, M. Andrade, P. Bretillon & T. Ahmed. *An Integrated Approach for End-to-End Quality of Service Offering in Multi-domain Heterogeneous Environments.* In Proceedings of the Eumob 2006, European Symposium on Mobile Media Delivery, in the scope of MobiMedia 2006 2nd International Mobile Multimedia Communications Conference (formerly MSAN), Alghero, Italy, September 2006.

[Bouras 05] C. Bouras & A. Sevasti. *Service level agreements for DiffServ-based services' provisioning.* Journal of Network and Computer Applications, vol. 28, no. 4, pages 285–302, 2005.

[Braden 94] R. Braden, D. Clark & S. Shenker. *Integrated Services in the Internet Architecture: an Overview.* RFC 1633, June 1994.

[Braden 97] R. Braden, L. Zhang, S. Berson, S. Herzog & S. Jamin. *Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification.* RFC 2205, September 1997.

[Burgstahler 03] L. Burgstahler, K. Dolzer, C. Hauser, J. Jhnert, S. Junghans, C. Macin & W. Payer. *Beyond technology: the missing pieces for QoS success.* In ACM SIGCOMM workshop on Revisiting IP QoS:

What have we learned, why do we care?, pages 121–130, Karlsruhe, Germany, 2003. ACM.

[Callen 06] C.R. Callen & J.S. Reeve. *Using Open Source to realise an NGOSS Proof of concept.* In Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP, pages 1–12, Vancouver, Canada, April 2006.

[Caruso 06] F. Caruso, D. Milham & S. Orobec. *Emerging industry standard for managing next generation transport networks: TMF MTOSI.* In Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP, pages 1–15, Vancouver, Canada, April 2006.

[Chen 04] Y. Chen, T. Farley & N. Ye. *QoS Requirements of Network Applications on the Internet.* Information Knowledge Systems Management, vol. 4, no. 1, pages 55–76, 2004.

[Cortese 03] G. Cortese, R. Fiutem, P. Cremonese, S. D'antonio, M. Esposito, S.P. Romano & A. Diaconescu. *Cadenus: creation and deployment of end-user services in premium IP networks.* Communications Magazine, IEEE, vol. 41, no. 1, pages 54–60, 2003.

[Cruvinel 08] L. Cruvinel, T. Vazao, F. Silva & A. Fonseca. *Dynamic QoS Adaptation for Multimedia Traffic.* In Computer Communications and Networks, 2008. ICCCN '08. Proceedings of 17th International Conference on, pages 1–7, 2008.

[Curado 05] M. Curado. *Quality of Service Routing for Class-Based Networks.* PhD Thesis, University of Coimbra, October 2005.

[Djarallah 09] N. B. Djarallah & H. Pouyllau. *Algorithms for SLA composition to provide inter-domain services.* In Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management, pages 460–467, New York, NY, USA, 2009. Institute of Electrical and Electronics Engineers Inc., The.

[Dobson 06] G. Dobson & A. Sanchez-Macian. *Towards Unified QoS/SLA Ontologies.* In Services Computing Workshops, 2006. SCW '06. IEEE, pages 169–174, 2006.

[Dugeon 07] O. Dugeon, D. Morris, E. Monteiro, W. Burakowski & M. Diaz. *End to End Quality of Service over Heterogeneous Networks EuQoS*. In Network Control and Engineering for QoS, Security and Mobility, IV, volume 229 of *IFIP International Federation for Information Processing*, pages 87–101. 2007.

[Dynagen] Dynagen. *Dynagen*. http://dynagen.org/. Last accessed on July 2010.

[Dynamips] Dynamips. *Dynamips Cisco 7200 Simulator*. http://www.ipflow.utc.fr/index.php/Cisco_7200_Simulator. Last accessed on July 2010.

[Engel 03] T. Engel, H. Granzer, B.F. Koch, M. Winter, P. Sampatakos, I.S. Venieris, H. Hussmann, F. Ricciato & S. Salsano. *AQUILA: adaptive resource control for QoS using an IP-based layered architecture*. Communications Magazine, IEEE, vol. 41, no. 1, pages 46–53, 2003.

[ETSI 08] ETSI. *TISPAN ETSI ES 282 001: Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); NGN Functional Architecture*, March 2008.

[ETSI 10] ETSI. *ETSI Technical Specification 122 105 (V9.1.0): Services and service capabilities*, October 2010.

[Freitas 10] R. B. Freitas, L. B. de Paula, E. Madeira & F. L. Verdi. *Using virtual topologies to manage inter-domain QoS in next-generation networks*. International Journal of Network Management, vol. 20, no. 3, pages 111–128, May 2010.

[Georgievski 03] M. Georgievski & N. Sharda. *A Taxonomy of QoS Parameters and Applications for Multimedia Communications*. In International Conference on Internet and Multimedia Systems and Applications, IMSA2003, pages 13–15, Kauai, Hawaii, USA, August 2003.

[Giladi 09] R. Giladi & E. Menachi. *ETNA's Service Layer Architecture for Automatic Provisioning of Inter-Domain Ethernet Transport Services*. In GLOBECOM Workshops, 2009 IEEE, Honolulu, Hawai, December 2009.

[Giordano 03] S. Giordano, S. Salsano, S. Van den Berghe, G. Ventre & D. Giannakopoulos. *Advanced QoS provisioning in IP networks: the European premium IP projects.* Communications Magazine, IEEE, vol. 41, no. 1, pages 30–36, 2003.

[Goestl 06] H. Goestl. *Using NGOSS Principles in todays OSS/BSS Projects NGOSS meets IMS/SDP.* In Telecommunications Network Strategy and Planning Symposium, 2006. NETWORKS 2006. 12th International, pages 1–8, New Delhi, India, November 2006.

[Gu 06] X. Gu & K. Nahrstedt. *Distributed multimedia service composition with statistical QoS assurances.* IEEE TRANSACTIONS ON MULTIMEDIA, vol. 8, pages 141—151, 2006.

[Gutierrez 08] P. Gutierrez, I. Miloucheva, D. Wagner, C. Niephaus, A. Flizikowski, N. Van Wambeke, F. Armando, C. Chassot & S.P. Romano. *NETQOS Policy Management Architecture for Flexible QoS Provisioning in Future Internet.* In Next Generation Mobile Applications, Services and Technologies, 2008. NGMAST '08. The Second International Conference on, pages 53–58, Cardiff, Wales, UK, September 2008.

[Hadjiantonis 07] A. M. Hadjiantonis, M. Charalambides & G. Pavlou. *An adaptive service management framework for wireless networks.* Vehicular Technology Magazine, IEEE, vol. 2, no. 3, pages 6–13, 2007.

[Hancok 05] R. Hancok, G. Karagiannis, J. Loughney & S. Van den Bosch. *Next Steps in Signaling (NSIS): Framework.* RFC 4080, June 2005.

[Howarth 05] M. P. Howarth, P. Flegkas, G. Pavlou, W. Ning, P. Trimintzios, D. Griffin, J. Griem, M. Boucadair, P. Morand, A. Asgari & P. Georgatsos. *Provisioning for interdomain quality of service: the MESCAL approach.* Communications Magazine, IEEE, vol. 43, no. 6, pages 129–137, 2005.

[Ibarrola 10] E. Ibarrola, F. Liberal, A. Ferro & J. Xiao. *Quality of service management for ISPs: A model and implementation methodology based on the ITU-T recommendation E.802 framework.* Communications Magazine, IEEE, vol. 48, no. 2, pages 146–153, 2010.

[IMS 07] IMS. *3GPP TS 23.228: IP Multimedia Subsystem (IMS); Stage 2. Release 7*, March 2007.

[IOS] IOS. *Cisco IOS Configuration Fundamentals Command Reference.* `http://www.cisco.com/en/US/docs/ios/12\_2t/fun/command/reference/fftabout.html`. Last accessed on August 2010.

[IPsphere 07a] Forum IPsphere. *The Business of IP: Delivering on the Promise of Convergence. Version 1.1*, June 2007.

[IPsphere 07b] Forum IPsphere. *IPsphere Framework Technical Specification (Release 1)*, June 2007.

[IPsphere 08] Forum IPsphere. *Interworking Session Services and Resource Management (SSRM) with IPsphere*, February 2008.

[Ismail 10] A. Ismail, J. Yan & J. Shen. *An offer generation approach to SLA negotiation support in service oriented computing.* Service Oriented Computing and Applications, vol. 4, no. 4, pages 277–289, November 2010.

[ITU-T 01] ITU-T. *ITU-T Recommendation G.1010: End-user multimedia QoS categories*, November 2001.

[ITU-T 04] ITU-T. *ITU-T Recommendation Y.2011: General principles and general reference model for Next Generation Networks*, October 2004.

[ITU-T 05a] ITU-T. *Next Generation Network Global Standards Initiative (NGN-GSI) Release I*, 2005.

[ITU-T 05b] ITU-T. *NGN FG Proceedings Part II*, 2005.

[Jacobs 05] P. Jacobs & B. Davie. *Technical Challenges in the Delivery of Interprovider QoS.* IEEE Communications Magazine, vol. 43, no. 6, pages 112–118, June 2005.

[JDOM] JDOM. *JDOM.* http://www.jdom.org/. Last accessed on November 2009.

[JGraphT] JGraphT. *JGraphT.* http://www.jgrapht.org/. Last accessed on November 2009.

[Jin 04] J. Jin & K. Nahrstedt. *QoS specification languages for distributed multimedia applications: a survey and taxonomy.* IEEE Multimedia, vol. 11, no. 3, pages 74– 87, September 2004.

[JUDDI] JUDDI. *JUDDI.* http://juddi.apache.org/. Last accessed on January 2010.

[Kim 08] D. Kim. *An architecture for internet inter-domain interconnections and bandwidth trading towards effective NGN deployment.* Annals of Telecommunications, pages 607–619, October 2008.

[Kovacikova 07] T. Kovacikova & P. Segec. *NGN Standards Activities in ETSI.* In Sixth International Conference on Networking, 2007. ICN '07., page 76, Sainte-Luce, Martinique, April 2007.

[Lavinal 09] E. Lavinal, N. Simoni, M. Song & B. Mathieu. *A next-generation service overlay architecture.* Annals of Telecommunications, vol. 64, pages 175–268, April 2009.

[Lin 09] K. Lin, J. Zhang & Y. Zhai. *An Efficient Approach for Service Process Reconfiguration in SOA with End-to-End QoS Constraints.* In Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing, pages 146–153. IEEE Computer Society, 2009.

[Liu 05] J. Liu, N. Gu, Y. Zong, Z. Ding, S. Zhang & Q. Zhang. *Service Registration and Discovery in a Domain-Oriented UDDI Registry.* In Computer and Information Technology, 2005. CIT 2005. The Fifth International Conference on, pages 276–283, 2005.

[Masip-Bruin 07] X. Masip-Bruin, M. Yannuzzi, R. Serral-Gracia, J. Domingo-Pascual, J. Enriquez-Gabeiras, M. A. Callejo, M. Diaz, F. Racaru, G. Stea, E. Mingozzi, A. Beben, W. Burakowski, E. Monteiro & L. Cordeiro. *The EuQoS system: a solution for QoS routing in heterogeneous networks [Quality of Service based Routing Algorithms for Heterogeneous Networks].* IEEE Communications Magazine, vol. 45, no. 2, pages 96–103, February 2007.

[Matos 08] A. Matos, F. Matos, P. Simões & E. Monteiro. *A Framework for the establishment of inter-domain, on-demand VPNs.* In

Network Operations and Management Symposium, 2008. NOMS 2008. IEEE, pages 232–239, Salvador, Brazil, April 2008.

[Matos 10a] F. Matos, A. Matos, P. Simões & E. Monteiro. *A QoS Model for Business Layers.* In The 24th International Conference on Information Networking (ICOIN 2010), Busan, Korea, January 2010.

[Matos 10b] F. Matos, A. Matos, P. Simões & E. Monteiro. *A Service Composition Approach for Inter-Domain Provisioning.* In 6th International Conference on Network and Services Management (CNSM), Niagara Fall, Canada, October 2010.

[Matos 11a] F. Matos, A. Matos, P. Simões & E. Monteiro. *Provisioning of Inter-Domain QoS-Aware Services.* Submitted to the Journal of Network and Systems Management (JNSM), November 2011.

[Matos 11b] F. Matos, A. Matos, P. Simões & E. Monteiro. *QoS Adaptation in Inter-Domain Services.* In IFIP/IEEE International Symposium on Integrated Network Management (IM), Dublin, Ireland, May 2011.

[Mingozzi 09] E. Mingozzi, G. Stea, M.A. Callejo-Rodrguez, J. Enrquez-Gabeiras, G. Garca-de-Blas, F.J. Ramn-Salquero, W. Burakowski, A. Beben, J. Sliwinski, H. Tarasiuk, O. Dugeon, M. Diaz, L. Baresse & E. Monteiro. *EuQoS: End-to-End Quality of Service over Heterogeneous Networks.* Computer Communications, vol. 32, no. 12, pages 1355–1370, July 2009.

[Mu 09] M. Mu, E. Cerqueira, F. Boavida & A. Mauthe. *Quality of Experience management framework for real-time multimedia applications.* Int. J. Internet Protoc. Technol., vol. 4, no. 1, pages 54–64, 2009.

[Mykoniati 03] E. Mykoniati, C. Charalampous, P. Georgatsos, T. Damilatis, D. Goderis, P. Trimintzios, G. Pavlou & D. Griffin. *Admission control for providing QoS in DiffServ IP networks: the TEQUILA approach.* Communications Magazine, IEEE, vol. 41, no. 1, pages 38–44, 2003.

[MySQL] MySQL. *MySQL.* http://www.mysql.com/. Last accessed on July 2009.

[OASIS] OASIS. *OASIS Web Services Security (WSS) TC.* `http://www.oasis-open.org/committees/tc\_home.php?wg\_abbrev=wss.` Last accessed on July 2010.

[OASIS 04] OASIS. *OASIS UDDI Specification TC.* http://uddi.org/pubs/uddi_v3.htm. Last accessed on August 2009, October 2004.

[OASIS 06] OASIS. *Reference Model for Service Oriented Architecture 1.0*, October 2006.

[Obreja 08] S. G. Obreja & E. Borcoci. *Overlay Topology Based Inter-domain Qos Paths Building.* In Proceedings of the 2008 Fourth Advanced International Conference on Telecommunications, pages 64–70. IEEE Computer Society, 2008.

[Oh 08] M. Oh, J. Baik, S. Kang & H. Choi. *An Efficient Approach for QoS-Aware Service Selection Based on a Tree-Based Algorithm.* In Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on, pages 605–610, Portlan, USA, May 2008.

[Papazoglou 07a] M. P. Papazoglou & W. J. Heuvel. *Service oriented architectures: approaches, technologies and research issues.* In The VLDB Journal, volume 16, pages 389–415. Springer-Verlag New York, Inc., 2007.

[Papazoglou 07b] M. P. Papazoglou, P. Traverso, S. Dustdar & F. Leymann. *Service-Oriented Computing: State of the Art and Research Challenges.* Computer, vol. 40, no. 11, pages 38–45, 2007.

[Pouyllau 09] H. Pouyllau, R. Douville, N. Djarallah & N. Sauze. *Economic and technical propositions for inter-domain services.* Bell Labs Technical Journal, vol. 14, no. 1, pages 185–202, 2009.

[Pouyllau 10] H. Pouyllau & R. Douville. *End-to-end QoS negotiation in network federations.* In Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP, pages 173–176, 2010.

[Pujol 05] J. Pujol, S. Schmid, L. Eggert, M. Brunner & J. Quittek. *Scalability analysis of the TurfNet naming and routing architecture.* In

Proceedings of the 1st ACM workshop on Dynamic interconnection of networks, DIN '05, page 2832, New York, NY, USA, 2005. ACM. ACM ID: 1080787.

[Quoitin 03] B. Quoitin, C. Pelsser, L. Swinnen, O. Bonaventure & S. Uhlig. *Interdomain traffic engineering with BGP.* Communications Magazine, IEEE, vol. 41, no. 5, pages 122–128, 2003.

[Rajan 06] G. Rajan, A.J.E. Armengol, P. Nooren & A. Foglar. *Policy Based QoS Architecture in MUSE.* In INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, pages 1–6, Barcelona, Spain, April 2006.

[Rao 08] S. Rao, S. Khavtasi, C. Chassot, N. Van Wambeke, F. Armando, S. P. Romano & T. Castaldi. *QoS Management in the Future Internet.* In World Congress on Science, Engineering and Technology (WCSET 2008), pages 549–555, Paris, France, July 2008.

[Rosen 01] E. Rosen, A. Viswanathan & R. Callon. *Multiprotocol Label Switching Architecture.* RFC 3031, January 2001.

[Rosen 06] E. Rosen & Y. Rekhter. *BGP/MPLS IP Virtual Private Networks (VPNs).* RFC 4364, February 2006.

[Rosenberg 02] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley & E. Schooler. *SIP: Session Initiation Protocol.* RFC 3261, June 2002.

[Rosenberg 09] F. Rosenberg, P. Leitner, A. Michlmayr, P. Celikovic & S. Dustdar. *Towards Composition as a Service - A Quality of Service Driven Approach.* pages 1733–1740. IEEE, March 2009.

[Royon 07] Y. Royon & S. Frenot. *Multiservice home gateways: business model, execution environment, management infrastructure.* Communications Magazine, IEEE, vol. 45, no. 10, pages 122–128, 2007.

[Rubio-Loyola 10] J. Rubio-Loyola, M. Charalambides, I. Aib, J. Serrat, G. Pavlou & R. Boutaba. *Business-driven management of differentiated services.*

In Network Operations and Management Symposium (NOMS), 2010 IEEE, pages 240–247, 2010.

[Salehie 09] M. Salehie & L. Tahvildari. *Self-adaptive software: Landscape and research challenges.* ACM Transactions on Autonomous and Adaptive Systems, vol. 4, no. 2, pages 1–42, May 2009.

[Schmidt 05] M. T. Schmidt, B. Hutchison, P. Lambros & R. Phippen. *The enterprise service bus: making service-oriented architecture real.* IBM Systems Journal, vol. 44, page 781797, October 2005. ACM ID: 1126978.

[Schulzrinne 03] H. Schulzrinne, S. Casner, R. Frederick & V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications.* RFC 3550, July 2003.

[Siekkinen 07] M. Siekkinen, V. Goebel, T. Plagemann, K.-A. Skevik, M. Banfield & I. Brusic. *Beyond the Future Internet–Requirements of Autonomic Networking Architectures to Address Long Term Future Networking Challenges.* In Future Trends of Distributed Computing Systems, 2007. FTDCS '07. 11th IEEE International Workshop on, pages 89–98, Sedona, Arizona, USA, March 2007.

[Skorin-Kapov 07] L. Skorin-Kapov, M. Mosmondor, O. Dobrijevic & M. Matijasevic. *Application-Level QoS Negotiation and Signaling for Advanced Multimedia Services in the IMS.* Communications Magazine, IEEE, vol. 45, no. 7, pages 108–116, 2007.

[Spiess 09] P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. Souza & V. Trifa. *SOA-Based Integration of the Internet of Things in Enterprise Services.* In Proceedings of the 2009 IEEE International Conference on Web Services, pages 968–975. IEEE Computer Society, 2009.

[Strassner 04] J. Strassner. Policy-based network management: solutions for the next generation. Morgan Kaufmann, 2004.

[Szwabe 06] A. Szwabe, A. Schorr, F. Hauck & A. Kassler. *Dynamic Multimedia Stream Adaptation and Rate Control for Heterogeneous Networks.* In 15th International Packet Video Workshop, Hangzhou, China, April 2006.

[TEQUILA 00] Consortium TEQUILA. *Tequila - D1.1: Functional Architecture Definition and Top Level Design*, 2000.

[TMF 02] TMF. *NGOSS ArchitectureTechnology Neutral Specification*, May 2002.

[TMF 09] TMF. *TMF061: Service Delivery Framework Reference Architecture; Version 1.0*, July 2009.

[Tomcat] Tomcat. *Apache Tomcat*. http://tomcat.apache.org/. Last accessed on July 2009.

[UDDI 02] Spec TC UDDI. *UDDI Version 2.04 API Specification*, July 2002.

[UDDI4J] UDDI4J. *UDDI4J*. http://uddi4j.sourceforge.net/. Last accessed on July 2009.

[UMTS 09] UMTS. *3GPP TS 23.107: Quality of Service (QoS) concept and architecture; Release 9*, December 2009.

[Vaquero 09] L. M. Vaquero, L. Rodero-Merino, J. Caceres & M. Lindner. *A break in the clouds: towards a cloud definition*. ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, pages 50–55, 2009.

[W3C 04] W3C. *Web Services Architecture*. http://www.w3.org/TR/ws-arch/. Last accessed on August 2009, February 2004.

[Wang 07] N. Wang, D. Griffin, J. Spencer, J. Griem, J.R. Sanchez, M. Boucadair, E. Mykomati, B. Quoitm, M. Howarth, G. Pavlou, A.J. Elizondo, M.L.G. Osma & P. Georgatsos. *A Framework for Lightweight QoS Provisioning: Network Planes and Parallel Internets*. In Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on, pages 797–800, Munich, Germany, May 2007.

[Wireshark] Wireshark. *Wireshark: Network Protocol Analyzer*. http://www.ethereal.com/. Last accessed on August 2009.

[Wolf 97] L. C. Wolf & R. Steinmetz. *Concepts for Resource Reservation in Advance.* Multimedia Tools and Applications, vol. 4, no. 3, pages 255–278, 1997.

[WSDL] WSDL. *Web Services Description Language for Java.* http://sourceforge.net/projects/wsdl4j/. Last accessed on August 2009.

[Xiao 99] Xipeng Xiao & L.M. Ni. *Internet QoS: a big picture.* Network, IEEE, vol. 13, no. 2, pages 8–18, 1999.

[Xiao 05] J. Xiao & R. Boutaba. *QoS-aware service composition and adaptation in autonomic communication.* IEEE Journal on Selected Areas in Communications, vol. 23, no. 12, pages 2344– 2360, December 2005.

[Zulkernine 11] F. H. Zulkernine & P. Martin. *An Adaptive and Intelligent SLA Negotiation System for Web Services.* IEEE Transactions on Services Computing, vol. 4, no. 1, pages 31–43, January 2011.

Appendix A

Service Templates

Service templates are the main structure used by providers to exchange information in GBF. They contain administrative, business and technical information about services and service elements. They also are used by providers to make their offers available at the UDDI. To guarantee portability and extensibility of services and service elements information, and facilitate the data manipulation, the service templates were designed as XML documents.

Figure A.1 presents the root element of the XSD. Rectangles with plus signs are complex elements that contain child elements. For the sake of organization, if a figure contains a complex element that is not expanded, this element will be described in subsequent figures. Dashed rectangles represent non-mandatory elements. Following, detailed descriptions of the entire XSD structure are presented

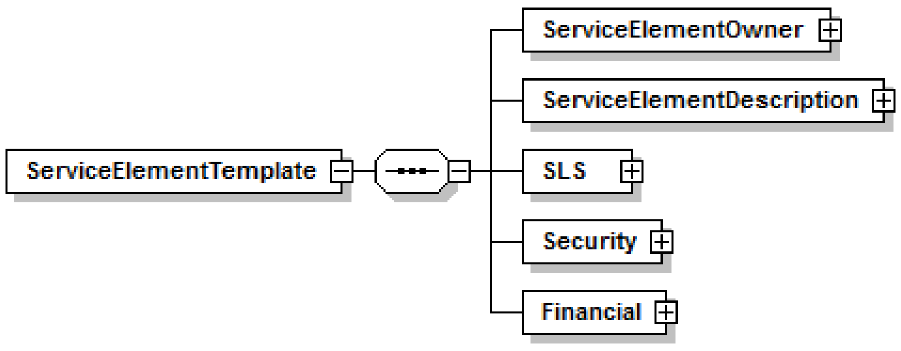


Figure A.1: XSD – Service Element Template

Service Element Owner

The *ServiceElementOwner* element depicted in Figure A.2 presents the identification and contact information of the EO that provides the service element. It contains the following sub-elements:

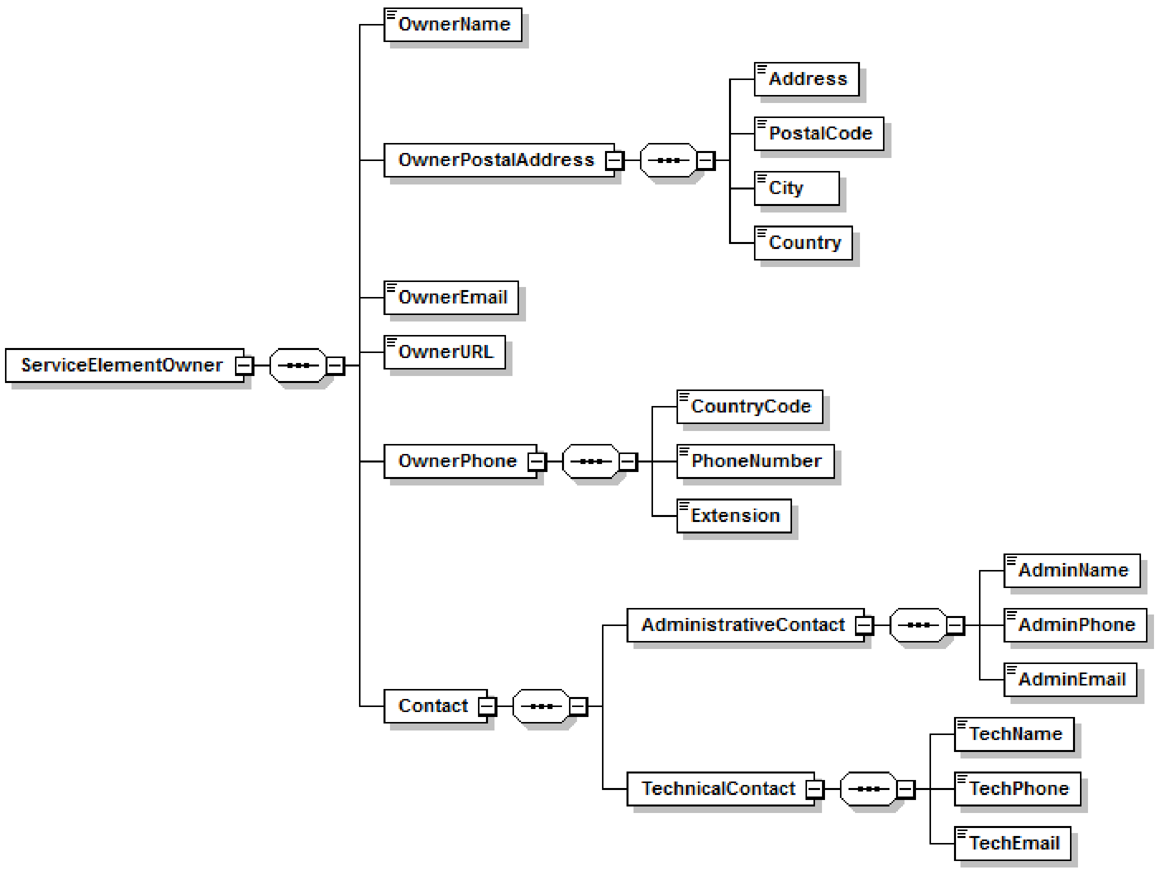


Figure A.2: XSD – Service Element Owner

- *OwnerName*: The EO name;
- *OwnerPostalAddress*: The physical address information of the EO;
- *OwnerEmail*: The EO e-email;
- *OwnerURL*: The EO web site;
- *OwnerPhone*: The EO contact phone number; and
- *Contact*: Name, phone number and email of the persons to contact at the EO to deal with administrative and technical issues.

Service Element Description

Figure A.3 presents general information about the service element. The *ServiceElementDescription* element contains the following sub-elements:

- *ElementName*: The name of the service element;
- *ElementType*: The type of the service element;
- *ElementClass*: Specifies the class (level of quality) of the service element; and
- *PublicationDate*: The publication date of the service element at the UDDI.

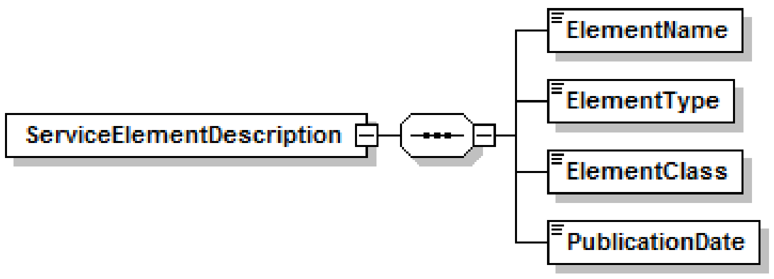


Figure A.3: XSD – Service Element Description

SLS

The *SLS* element (Figure A.4) contains the technical information about the service element. It contains the following sub-elements:

- *ElementId*: The service element identification;
- *CarrierId*: The identification of the EO that provides the service element;
- *CarrierDomain*: The domain of the EO that provides the service element. The SO uses this element during the service composition process to build the graph representation based on the domain connectivity between the providers;
- *ReachableCarriers*: The domains that are reachable from the EO that provides the service element. This element is also used during the service composition process. It is optional. Only CoEs have this element, since they provide service connection between the domain of the EO (*CarrierDomain*) and the reachable domains (*ReachableCarriers*); and
- *QoS*: The QoS related information of the service element.

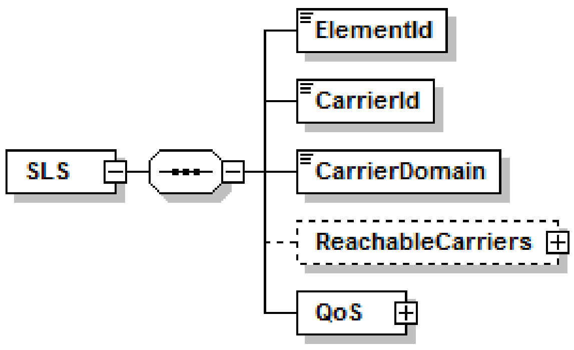


Figure A.4: XSD – SLS

Reachable Carriers

The optional *ReachableCarriers* element (Figure A.5) is composed of a group of *ReachableCarrier* elements. This element is optional because it is only applicable to CoEs. The *ReachableCarriers* element must have at least one *ReachableCarrier* element, which contains the following sub-elements:

- *CarrierId*: The identification of the EO reachable from the EO that provides the service element; and
- *CarriedDomain*: The domain of the EO reachable from the EO that provides the service element.

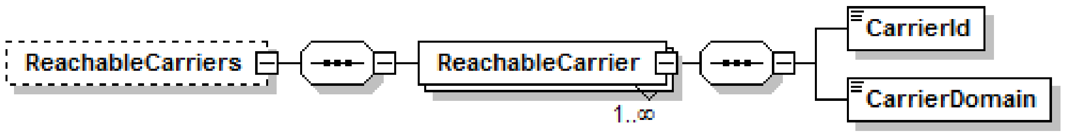


Figure A.5: XSD – Reachable carries

QoS

The *QoS* element (Figure A.6) contains the QoS information of the service element provided by the EO. It encompasses the main part of the technical information considered by SO when composing the service. It contains the following sub-elements:

- *Name*: Name of the QoS class (e.g. Basic, Silver, Premium);

- *Parameters*: This element contains the performance and service-specific parameters the EO guarantees for its service element;
- *Reliability*: This element defines the mean downtime per year (MDT) and the maximum time to repair (TTR) in the case of service disruption, if applicable. Both *MDT* and *TTR* elements have a *Value* and a *Unit* sub-elements; and
- *ExcessTreatment*: This element specifies how the out-of-profile and/or in excess packets should be handled.

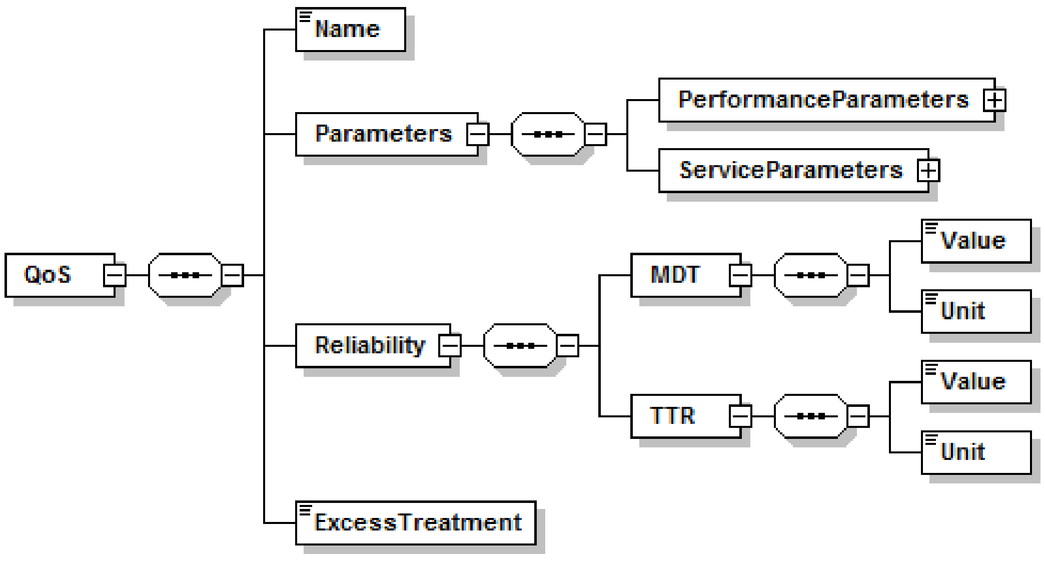


Figure A.6: XSD – QoS

Performance Parameters

The *PerformanceParameters* element (Figure A.7) represents QoS performance parameters that the provider (EO) guarantees to the customer (SO). It may contain several sub-elements depending on the type of the service element the EST represents. For this reason, all sub-elements of the *PerformanceParameters* element are optional.

Every sub-element at the *PerformanceParameters* element has the same sub-elements. These sub-elements specify the quantitative values of the parameter that the sub-element refers to. An example of a performance parameter is presented in Figure A.8. It shows the *Delay* element, which contains the following sub-elements:

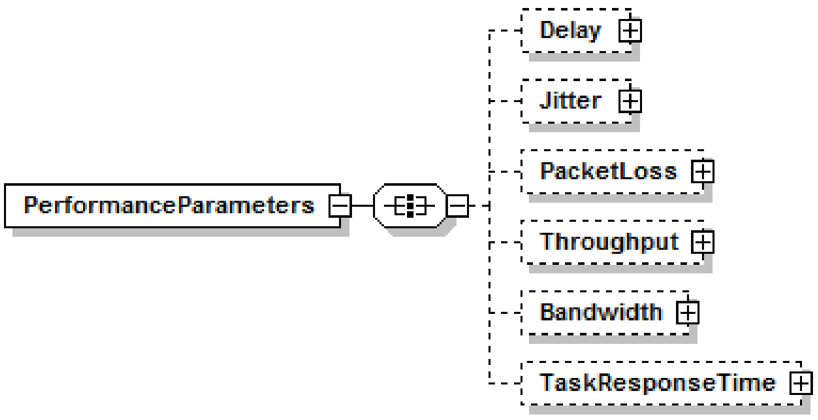


Figure A.7: XSD – Performance parameters

- *QuantitativeMaximum*: It specifies the maximum threshold value of the parameter for the specified quality (e.g. 150 ms for a medium delay);
- *QuantitativeMinimum*: It specifies the minimum threshold value of the parameter for the specified quality (e.g. 50 ms for a medium delay); and
- *Unit*: The unit of the parameter (e.g. ms, %, unitless).

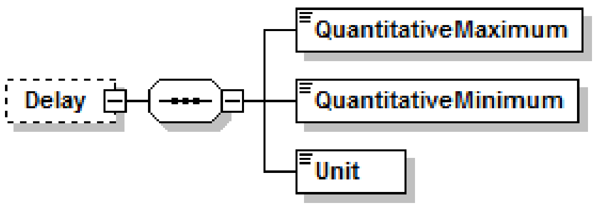


Figure A.8: XSD – Delay

Service Parameters

The *ServiceParameters* element (Figure A.9) represents service-specific parameters that the provider (EO) guarantees to the customer (SO). Depending on the type of the service element, each sub-element of the *ServiceParameters* element may or may not appear. For this reason, they are all optional. The sub-elements are:

- *Value*: The value of the parameter for the specified quality (e.g. MPEG, IPSec);
- *Unit*: The unit of the parameter (e.g. fps, unitless).

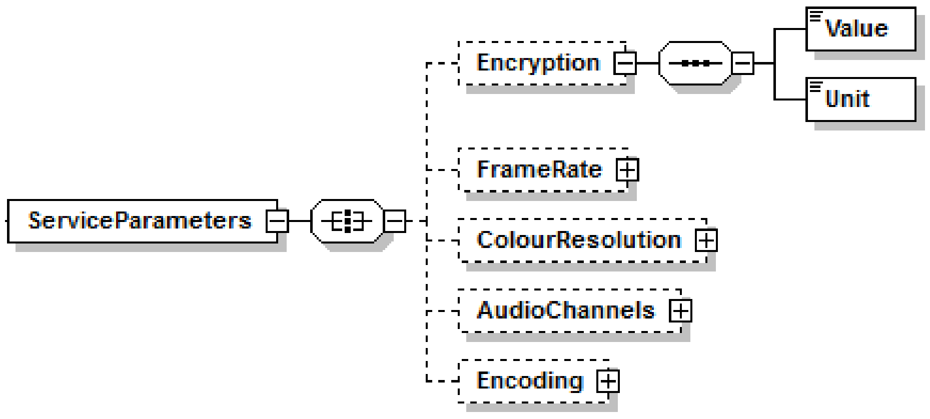


Figure A.9: XSD – Service parameters

Security

The *Security* element (Figure A.10) defines the security requirements of the service element. It may contain several *SecurityConstraints* elements that are optional and all of them have the following sub-elements:

- *Constraint*: It specifies a constraint that must be satisfied; and
- *Action*: It specifies the action that must be taken in the case the constraint is not satisfied. For instance, a constraint may state that service element can only be accessed through an encrypted tunnel. Otherwise, the EO does not provide the service element.

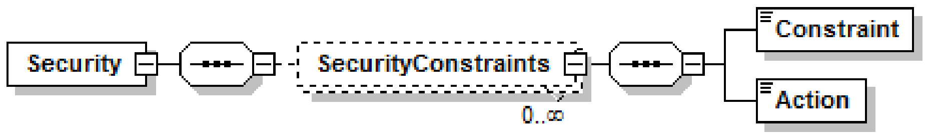


Figure A.10: XSD – Security

Financial

The *Financial* element (Figure A.11) defines the monetary values charged against actions requested by the contractor of the service element. It contains the following sub-elements:

- *Currency*: The currency the EO uses for its monetary transactions;
- *ActivationCharge*: The value EO charges for the activation of the service element;
- *InterruptionCharge*: The value EO charges for the interruption of the service element due to customer request;
- *ChangeCharge*: The value EO charges for any modification of the service element configuration due to customer request.

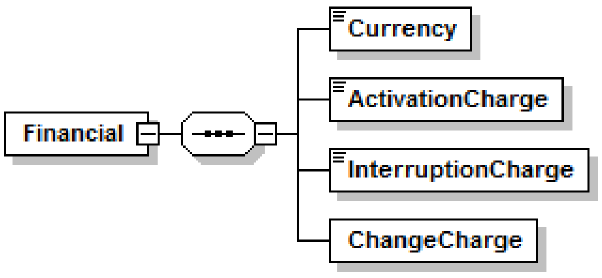


Figure A.11: XSD – Financial