UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA


# Q-Andrew
# A CONSOLIDATED QOS MANAGEMENT FRAMEWORK

**Luís Miguel Dias Costa**

MESTRADO EM SEGURANÇA INFORMÁTICA

Dezembro 2008

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA


# Q-Andrew
# A CONSOLIDATED QOS MANAGEMENT FRAMEWORK

**Luís Miguel Dias Costa**

**Orientador**
Hyong S. Kim

MESTRADO EM SEGURANÇA INFORMÁTICA

Dezembro 2008

## Resumo

As redes IP convergentes são compostas por uma diversidade de tecnologias que suportam múltiplos tipos de serviços com diferentes características. Cada fabricante de equipamento activo de rede usa sistemas de manutenção proprietários, incompatíveis com equipamentos de outros fabricantes. Para um operador de telecomunicações a gestão da Qualidade de Serviço, numa rede composta por vários fabricantes, é uma tarefa complexa e dispendiosa. Algumas tarefas requerem configuração manual para garantir a compatibilidade entre configurações de equipamentos de fabricantes diferentes. Melhorar a resposta operacional e reduzir os custos de operação nestas circunstâncias é apenas possível com a consolidação da gestão de rede. Para responder a este desafio, propomos: – Um conjunto de mecanismos geradores de configurações de Qualidade de Serviço, consistentes entre equipamentos de diversos fabricantes; – A definição de um modelo abstracto de representação destas configurações, reutilizável em futuras aproximações de gestão consolidada de rede; – Por fim, descrevemos uma aplicação de demonstração onde algumas das propostas apresentadas são concretizadas, tendo como objectivo futuro a sua utilização numa rede real de um operador de telecomunicações nacional, onde são utilizados equipamentos de diversos fabricantes.



Palavras-chave: Gestão de Redes; Gestão de Qualidade de Serviço; Gestão de Redes Convergentes.

## Abstract

Converged IP networks consist of diverse technologies and support both legacy and emerging services. Different vendors use separate management systems to achieve similar goals. Manual provisioning today represents a large portion of the total effort required to manage a complex IP network. A consolidated Quality-of-Service policy is difficult to implement in heterogeneous networks. Creating and maintaining such policies is very demanding in terms of operations. For this reason, reducing operational costs while improving Quality-of-Service Management is only possible through a consolidated approach to network management. To leverage operations in converged IP networks, we propose the following: – A mechanism to automatically generate consistent configurations across a network with equipment from different vendors; – A framework definition such that network element configurations can be specified using a common model; – Applying some of the methods proposed to an application that can be used in a real network with diverse technologies and equipment vendors.

Keywords: Network Management; Quality of Service Management; Converged Network Management.

## Acknowledgments

I would like to thank Carnegie Mellon University, the Faculty of Sciences at the University of Lisbon (Faculdade de Ciências da Universidade de Lisboa) and the sponsors of the MSIT-IS program for their support during the last 16 months.

I would like to thank Professor Hyong Kim for his inspiration during the development of this work.

I would like to thank Professor Paulo Sousa for his insights during the development of this report.

I would also like to thank my MSIT-IS program colleagues Ricardo Marques, Ricardo Oliveira and Tiago Carvalho for their friendship throughout this program.

Lisbon, December 2008

*Dedicated to my wife, Vera.*

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Service Provider networks consist of many network elements from different vendors. Networks are complex to maintain since they support a large number of services and new services require timely deployment and quality guarantees. Given the range of new application requirements, such as IP Television Broadcast (IPTV) and Video on Demand (VoD), IP networks are required to provide Quality of Service (QoS).

Each vendor's management interface is incompatible with other vendors' devices. The data plane is dependent on the management plane and each vendor has his own configuration method. Furthermore, the evolution of protocols, services and network elements has increased the complexity of management interfaces. Given this diversity, configuration deployment has become a critical task. A Converged Management approach to network operations is necessary to ensure timely and trustworthy configuration deployment in a large Service Provider network. A management interface must be capable of handling service and network diversity. The Converged Management approach addresses this topic: network management in the presence of different services, in a network consisting of devices from several vendors.

We propose a Management Framework where network configurations, application requirements and other sources of information are represented in a common model. We provide re-usable components so that many management applications can be developed on top of the framework. We describe QoS management mechanisms for a network with device diversity. An application is developed, demonstrating the use of the framework proposed and the implementation of some of the QoS management mechanisms is described. Vendors use different QoS mechanisms and their underlying implementation is not directly comparable between different devices. The application demonstrates the QoS policy configuration automation in a network with several vendors, unifying the end-to-end requirements of different vendors.

Finally, we present the Q-Andrew application and the Network Management Framework

developed to a European Service Provider representative. The Service Provider is used for reference in this work.

## 1.1 Reference Service Provider

A European Service Provider is used for reference in this work. The following paragraphs list the topics that the Service Provider considers important to address:

- Large number of devices to configure:
  - The Service Provider has many network locations, all interconnected.
- Device-diversity:
  - Different kinds of devices, from entry-level routers and switches, to high-grade routers;
  - Vendor diversity: Cisco, Juniper, Enterasys and others are used in the network.
- Network organization complexity:
  - The evolution of Service Providers as a group of companies has resulted in several networks without a consistent design policy;
- Management responsibilities:
  - Network management responsibility is spread across several departments;
  - QoS management requires uniform configuration across the network, providing the quality requirements for different services;

The Service Provider provides configuration data, application usage, and logical network topology. Using this data and the Service Provider concerns previously listed, the following topics are identified as requiring special attention:

1. Network Management teams:
   (a) Handling many different management teams with different responsibilities;
2. Monitoring facilities must be used to analyze the QoS policies deployed:
   (a) Constant active and passive monitoring ensures that policies applied have the expected results;

3. Many vendors, hardware and software versions:

   (a) Each vendor has his own management platforms;

   (b) Each vendor has his own operating systems;

   (c) When software versions differ between platforms, configuration can be different;

4. Network topology is difficult to determine in a complex, constantly evolving network environment.

This work addresses topics 3 and 4. A framework and a demonstration application are proposed addressing these topics.

## 1.2 Previous Work

Network vendors, such as Cisco and Juniper, develop their own software management solutions capable of handling a large number of devices in complex network environments. Although most network vendors offer mature management products, they require networks composed exclusively of their own networking equipment.

### 1.2.1 Cisco - CiscoWorks

CiscoWorks is a modular management software platform from Cisco, where the customer buys only the required functionalities. The Quality of Service Policy Manager is one of the modules available:

Information from the vendor: CiscoWorks Quality of Service Policy Manager provides centralized management of quality of service policy creation, validation, deployment, and monitoring to facilitate the secure and predictable delivery of networked services, such as business application, video, voice over Internet Protocol (VoIP), Cisco TelePresence®, and other networked applications [1].

### 1.2.2 Juniper Networks - Service Deployment System

Juniper Network's Service Deployment System (SDX), relies on gateways and tracking applications enabling the configuration of the network according to the monitored usage and Quality of Service goals.

Information from the vendor: Juniper Networks SDX-300 is a task-focused software-based policy server that identifies, allocates, and modifies network resources in response to subscriber and application requests, privileges, and priorities. The SDX-300 supports a variety of optional features – such as the Admission Control Plug (ACP), which authorizes

and tracks the use of network resources and enables application aware call admission control (CAC) — as well as Web portals that extend service control to individual users and administrators [2].

### 1.2.3 Management Framework and Q-Andrew

We propose a Management Framework that can be re-used by the Service Provider to extend its management tools. Vendors' sell modular management software that is only compatible with their own devices. With this limitation, the operational problem remains the same, configurations are difficult to deploy in a network with different management interfaces. To overcome this issue, we provide a demonstration application entitled Q-Andrew, supporting Cisco and Juniper management interfaces. This tool gives the network operator the ability to deploy a unifying QoS configuration across a network with devices from these vendors. The application is developed using the Management Framework mentioned, enabling its development in the future to better suit the Service Provider operational requirements.

## 1.3 Document Organization

In Chapter 2, available QoS Architectures and Mechanisms are described. The Differentiated Services use in this work is justified. In Chapter 3, a Management Framework is described. The configuration parsing process and the resulting abstract model is presented. This framework is then used to develop the Q-Andrew application. Chapter 4 presents the Q-Andrew application that is used to demonstrate some of the mechanisms proposed and the reference QoS Policy_described in Chapter 2. The application uses the Management Framework to create a unifying QoS management application, where QoS policy configurations are generated automatically, using data from user input and from the abstract model. Chapter 5 presents the feedback from the Service Provider regarding this work. Chapter 6 presents the conclusions from this work and future work possibilities are identified.

# Chapter 2

# Quality of Service Management

## 2.1 Introduction

We use several data-plane mechanisms allowing the implementation of a Converged QoS Policy across multiple devices from different vendors. To achieve this goal, vendors have to support the same, or equivalent mechanisms in their operating systems. This fact allows the definition of a global policy across the managed domain. There are several QoS Architectures available that can be used to provide a Converged QoS Policy. The chosen QoS architecture for this project is Differentiated Services (DiffServ [3]), since it is scalable, widely supported and currently in use in our reference Service Provider.

The IP Precedence field in the original Postel's Internet Protocol [4] RFC, defines the notion of precedence as a simple priority scheme that does not provide a set of capabilities for today's network SLA requirements. DiffServ maintains backward compatibility with the IP Precedence Field, using Class Selector Code Points (CS), since the Differentiated Services Code Point (DSCP) that replaces it does not interfere with the IP Precedence-marked packets. The definition of type-of-service evolved and the IP Type of Service (ToS) [5], is intended to give additional meaning to some of the bits in the IP ToS header. The ToS field was not created to address queuing at a network node, but rather the actual path a packet should take. Integrated Services [6] and DiffServ have been developed to overcome the limitations of IP Precedence and IP Type of Service. IntServ provides mechanisms to manage bandwidth management. The Resource Reservation Protocol [7] is used by IntServ to signal end-to-end paths and admission control is executed for each new flow. Since IntServ requires the handling of each flow independently, the amount of resources required to support it have made it difficult to use and configure widely.

DiffServ [3] was defined to overcome the scalability issues of IntServ. DiffServ configurations are provisioned instead of signaled (as in IntServ) in a path for each flow. At the DiffServ Domain Edge, traffic is marked and classified using filters or access control lists and

scheduling is used to handle traffic according to the Per-Hop Behavior configured. Since DiffServ does not keep per-flow state, each hop requires specific configuration according to its location within the DiffServ Domain. DiffServ has been the most widely adopted QoS Architecture and support has been added to IPv6 [8, 9] and Multi-Protocol Label Switching (MPLS) networks [10, 11], making it future proof and the most suitable architecture for this project. The limitation of DiffServ, and most QoS architectures, is that SLAs can only be provided within a DiffServ domain or domains. Providing SLA for Internet services requires handling different DiffServ domains, routing domains without QoS support or domains with no QoS agreement established.

QoS Mechanisms are used and implemented differently by vendors. The use of scheduling, queuing disciplines and other mechanisms is not standardized, with each vendor offering a different set of options for what should be the same configuration goal. Additionally, the underlying implementation reflects the technical options of each vendor when developing the software. In this scenario, using devices from different vendors, or even different devices from the same vendor, results in different implementations of the same mechanisms. We overcome these limitations by creating DiffServ QoS policies that are compatible between different vendors in the same DiffServ domain. We analyze each configuration section in detail, unifying the configurations between different vendors, providing an operational interface allowing an easy deployment of such policies. This approach is applicable to the DiffServ configuration in a heterogeneous environment were different transport mechanisms are used. We develop a demonstration application entitled Q-Andrew with the ability to create unified QoS policies in a DiffServ domain with network elements from Cisco and Juniper, interconnected with Ethernet links. MPLS forwarding mechanism can be viewed as an additional DiffServ domain as described in 4.4, where DiffServ configuration requires additional settings to be deployed in such areas. Q-Andrew application does not demonstrate MPLS domains' configuration but the tool can be extended in the future to overcome this limitation. Given the limited time available for development, Q-Andrew demonstrates the unifying configuration capability over a single DiffServ domain with static routing configurations and Ethernet links.

## 2.2 QoS Mechanisms

A QoS architecture uses a combination of mechanisms to provide the various traffic differentiation behaviors of services. The QoS mechanisms used in this work are briefly described in the next sections.

### 2.2.1 Packet Classification and Marking

Packet marking, or packet coloring, is the mechanism that allows an IP or MPLS packet to be classified into a given traffic class according to a previously defined QoS Policy. A packet incoming to a DiffServ domain must be marked according to a QoS policy, allowing hops within the domain to classify marked traffic to the proper service class. This marking uses designated fields in the IP or MPLS header. Access-control lists or firewall filters are used to match traffic with its corresponding service class.

### 2.2.2 Queuing

#### 2.2.2.1 Priority Queuing

Priority queuing ensures that queues are emptied as determined by a simple priority scheduling algorithm, allowing traffic in a higher priority queue to be forwarded before traffic in a lower priority queue. Starvation must be avoided by the use of other mechanisms, such as rate or admission control.

#### 2.2.2.2 Rate Queuing

Rate queuing employs a similar approach to Priority Queuing, with the additional specification of a queue emptying rate. Each queue is emptied at a specified rate. Weighted Fair Queuing (WFQ) and Weighted Round Robin (WRR) are common algorithms that can be found in network devices. These algorithms delay a packet for a period that depends on queue occupancy status and its occupancy relative to other queues.

### 2.2.3 Active Queue Management (AQM)

Active Queue Management algorithms are used to avoid congestion. Before congestion occurs, packets may be dropped providing the feedback needed for end-systems to adjust their throughput. A common mechanism is Random Early Detection (RED) [12]. RED configuration is composed of parameters that set a minimum and maximum threshold for queue size, and a drop probability. RED is better suited to protocols that rely on feedback mechanisms to control throughput, such as TCP.

### 2.2.4 Policing/Admission Control and Traffic Conditioning

Policing is a mechanism used to ensure that a given traffic flow does not exceed the committed maximum rate. The simplest way to implement this mechanism is to use a single

token bucket [13], specifying committed rate (the rate tokens are added to the bucket), and a burst size (the bucket depth) for a given flow.

A Single Rate Three Color Marker (srTCM) defined in [14], is a policer that works in a traffic-light scheme. The three colors stand for the three possible outcomes of the policer. The policer is implemented using two token buckets similar to the previous example. The first bucket is the Committed bucket, where traffic conforming to the committed rate is handled and is said to be Green. Traffic not handled by the first bucket is then passed to the second bucket, were the Excess traffic is handled and is said to be in the Yellow state. Finally, the remaining traffic not conforming to the configurations of the previous buckets, is set to the Red state. Each color is then associated to an action that corresponds to the transmit and mark actions for the first two buckets (in-contract and out-of-contract), and drop for the remaining traffic (out-of-contract and unacceptable).

The Two Rate Three Color Marker (trTCM) [15], works in a similar fashion as the srTCM with the difference that buckets are configured with different committed rates. This allows different rates to be set for in and out-of-contract traffic flows.

## 2.3 Differentiated Services QoS Architecture

DiffServ addresses the limitations of IntServ. IntServ requires signaling for each flow, while DiffServ is pre-configured. In DiffServ, network elements are provisioned according to predefined requirements. Additionally, DiffServ has been extended to MPLS, which is used in most of service provider's networks, including our reference European Service Provider. Per-Hop Behaviors (Section 2.3.2) must be configured in a range of devices from different vendors. The DiffServ standard does not impose an underlying configuration for the QoS mechanisms to provide the different PHBs. Instead, it is up to the network designer to deploy the configurations required to ensure a compatible behavior with a given PHB. The following sections briefly describe the Differentiated Services QoS Architecture.

### 2.3.1 DS Field and DSCP

IP Packets are marked at the DiffServ domain edge. An octet from the IP Header is used to mark each packet [3]. This field is named Differentiated Services (DS) Field and within this octet-field, bits 0 to 5 are used as the Differentiated Services Code Point (DSCP), representing the corresponding Per-Hop Behavior (PHB) for the packet (Table 2.1).

| **Codepoint** | **DSCP** | | |
|---|---|---|---|
| Default/CS0 | 000000 | | |
| EF PHB | 101110 | | |
| CS1 | 001000 | | |
| CS2 | 010000 | | |
| CS3 | 011000 | | |
| CS4 | 100000 | | |
| CS5 | 101000 | | |
| CS6 | 110000 | | |
| CS7 | 111000 | | |
| **AF PHB Group** | **Drop Precedence** | | |
| **AF Class** | **Low (AFx1)** | **Medium (AFx2)** | **High (AFx3)** |
| AF1x | AF11=001001 | AF12=001010 | AF13=001011 |
| AF2x | AF21=010001 | AF22=010010 | AF23=010011 |
| AF3x | AF31=011001 | AF32=011010 | AF33=011011 |
| AF4x | AF41=100001 | AF42=100010 | AF43=100011 |

Table 2.1: DSCP Markings

### 2.3.2 Per-Hop Behavior (PHB)

The DSCP marking allows a network node to identify the Service Class a given packet belongs to. According to this marking, the corresponding traffic conditioning is applied. This behavior is identified as a Hop Behavior, which occurs at every hop in a DiffServ domain. DiffServ RFC [3] does not explicitly identify the scheduling behavior and queuing disciplines that should be used to provide a given hop-behavior. Instead, abstract Per-Hop Behaviors are defined, which represent the intended external observable behavior.

#### 2.3.2.1 Expedited Forwarding (EF) PHB

EF PHB[16] is used to support applications such as VoIP where low delay, jitter and loss are critical requirements. An EF PHB is typically deployed using a strict priority queuing mechanism, ensuring that EF traffic is given more priority than other competing traffic. It is also required an additional setting to control the admission of EF PHB traffic, preventing other traffic classes from being starved.

#### 2.3.2.2 Assured Forwarding (AF) PHB

AF PHB[17] guarantees, with high probability that packets from this PHB are delivered to their destination. Absolute or relative bandwidth guarantees are also provided by this PHB. This traffic is typically allocated to queues using Weighted Fair Queuing (WFQ) or Weighted Round Robin (WRR) packet scheduling algorithms [18].

#### 2.3.2.3 Class Selector (CS) PHB

The Class Selector PHB [9] was originally developed to provide backward compatibility with the IP precedence field behaviors. Recently, this PHB is used to facilitate the mapping between DiffServ markings and the MPLS experimental field, to provide PHBs' adoption in MPLS domains.

# Chapter 3

# Management Framework

The Management Framework facilitates the development of management applications for large networks. We use the framework described in this Chapter to develop Q-Andrew application, demonstrating both the Management Framework extensibility and also to implement a unifying QoS Management configuration application. This framework is developed taking into account the existence of several vendors and different types of network elements. To ensure future usability, the framework is developed with high-level abstraction approaches both in the model representation as well as internal Java code class organization. The following Sections present the Design Assumptions taken during the development of the Management Framework.

## 3.1 Design Assumptions

In this Section, we describe the design assumptions of the proposed management framework.

### 3.1.1 Configuration Availability

We assume that configurations scripts are available allowing a complete model to be devised, i.e. we consider a model where virtually all the configurations from all the network elements are known. In some situations this may be unfeasible, for instance, due to network complexity or privacy issues. Alternatively this can be addressed by manually identifying the missing configurations, creating virtual configuration hosts representing the missing configuration(s) directly in the model or updating the graphical user interface described in Chapter 4, to allow the network operator to add the missing nodes or network segments to the model.

### 3.1.2 Standards-based Configurations

In the proposed abstract model, only standards-based configurations are considered. Proprietary protocols that are not implemented by other vendors are not considered in this project since they violate the Converged Management principle, where different devices from different vendors can be configured using the same protocols.

### 3.1.3 Existing Configurations

Existing QoS configurations already deployed and present in network device configurations are not parsed and imported into the model. The assumption is that the application and model will produce a QoS policy in a network that does not have any QoS configuration. This assumption is required due to the complexity of parsing different QoS policies that would not be consistent across the network. Instead, this work focuses on automated QoS configuration generation. This limitation can be addressed as future work where a complete configuration cycle can be developed ensuring the continuous configuration process of a network.

### 3.1.4 Monitoring Data

Since the main goal of the project is to automate the configuration script generation process, it is assumed that monitoring facilities already exist and that usage data can be extracted. Active and passive monitoring facilities are used to verify that the deployed configurations satisfy the application requirements.

## 3.2 Architecture

The main purpose of the framework is to allow several management principles to be implemented using a single common model. This model is a formal representation of a real network configuration. In this approach, the model is created by getting information about network element configurations. Configurations from a large SP are used as the starting point for the creation of the model.

As expected, the reference has network elements from different vendors. Different vendors have distinct management interfaces, since there is no standard for representing a configuration script. Additionally, some devices from the same vendor have to be configured differently from each other (e.g., Cisco IOS, IOS XR, CatOS). The result is additional work for the network operator since each vendor or vendor/element requires special attention to details for the same configuration across a path.

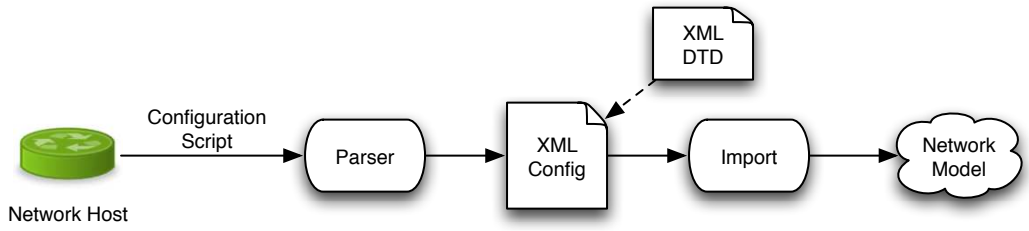


Figure 3.1: Configuration to Model process

Once configurations are collected from the routers, each configuration is parsed and converted into an XML common format. Proprietary information, not relevant to the model, is not considered. With the XML version of all the required configurations, the common abstract model is created, based on the collected information. The XML mapped information is translated the into the common configuration model (Figure 3.1). This can be achieved since we abstract from the fact that the hosts are configured using different languages or methods. This way, we have a unique mechanism to represent compatible and standards-based configurations (Section 3.1.2). Other sources of knowledge such as SLA objectives, monitoring data or physical network interconnections can be represented and stored using this model. This allows the implementation of functions that otherwise could not be developed using only configuration knowledge. Figure 3.2 shows the possible knowledge sources, described in the next Sections.

## 3.3 Knowledge Sources

Figure 3.2 represents the possible sources of information considered in this project.

### 3.3.1 Non-XML Network Element

A non-XML Configurable network element is configured using a proprietary mechanism. These mechanisms are usually vendor-specific configuration scripts or specific graphical user interfaces that configure the hosts using a proprietary mechanism. A familiar example would be the Cisco IOS scripting language and the Juniper JUNOS configuration script.

Due to the proprietary properties of the configurations of each vendor, we use a common abstract model to associate compatible configurations using an intermediate XML representation. This is achieved using a parser that recognizes the configurations according to each vendor's proprietary format. This XML configuration is later imported to the model by an XML engine that has some knowledge about the application logic, creating initial properties on the model.

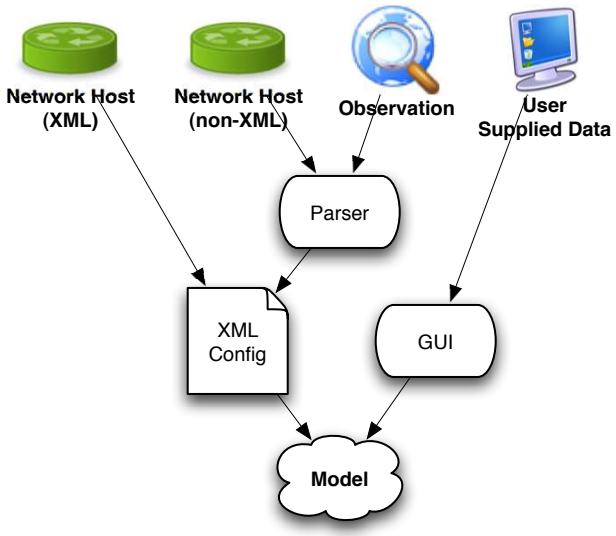


Figure 3.2: Knowledge Sources

**Implementation**

The parser implemented for Cisco routers is capable of handling the configuration sections of Cisco IOS. Real configurations are used to test the parser that is able to read Cisco IOS versions from 11.0 to 11.3 and from 12.0 to 12.4. The QoS sections of the configurations are not imported into the model, since we assume that we are configuring a complete model from the beginning and that no legacy configurations need to be supported (Section 3.1).

### 3.3.2 XML Network Element

An XML Network element is configured using a vendor-specific API that interacts with the target using XML-based information. This approach has many advantages in terms of automation of tasks and the creation of specific tools to interact with the hosts to be configured. For instance, both Cisco and Juniper provide an API and XML schema information to allow third party development. This is an important achievement that proves that our approach is future proof in terms of the basic mechanisms used by major network equipment vendors. A configuration represented in XML is readily available to a simpler mapping to the model proposed in this project.

### 3.3.3 User Supplied Data

The network operator interacts with the model, suppling information about the QoS policy, services, and the paths to configure in the DiffServ domain. This is achieved either using Q-Andrew's Graphical User Interface (GUI) (Section 4) or by editing the Ontology manually (Section 3.4). Additionally, the topology can be constructed manually adding nodes to the graph representing the network. This simple approach allows the creation of new configurations for new devices. This feature is not implemented, but both the framework and the application support this simple extension with additional development.

#### Implementation

Path configurations fall into this category. The model reflects the traffic paths configured for each protocol by the network operator. This information is later used to compute the configuration scripts to apply to the target network.

### 3.3.4 Network Data

As configurations are applied to a network of devices, the results of applying these configurations have to be confirmed to do what they were planned to. Observations should be included in the same model as network configurations, storing the current and past results of network monitoring. This allows the network operator to make decisions based on a history of configurations. Using our QoS management proof-of-concept application as an example, observations would be protocol usage statistics or the amount of delay at a given network node. These observations would then be used by the network operator to confirm that the results of applying a set of configurations have the expected results or to change the QoS policy accordingly. Although the implementation of this mechanism was not possible due to lack of data relative to automated configuration with usage information, it is fair to assume that a real deployment of the Q-Andrew application will require such inclusion.

### 3.3.5 Additional Sources of Data

Other information that does not result from network data or GUI-User supplied data can also be represented in the model, such as physical location of Hosts, business objectives, SLA requirements, etc. This kind of information is usually kept in distinct databases, which are not easily accessible by different applications. This is an additional benefit of using the approach presented in this work where information can be easily related and accessible from the application being developed. We describe the knowledge representation and storage possessing these properties in the following section.

## 3.4 Ontology

In general terms, an ontology is a definition of a set of primitives upon which a domain of knowledge can be modeled. In the context of database systems, an ontology is a level of abstraction of data models, similar to hierarchical and relational models, but intended for modeling knowledge about individuals, their attributes and their relationships to other individuals [19]. In this project, the Web Ontology Language and its primitives are used to represent the general domain of network management and, particularly, the QoS management domain.

### 3.4.1 OWL Web Ontology Language

OWL is a semantic markup language that can be used to share ontologies on the World Wide Web. This futuristic view of the Web is sometimes called Semantic Web or Web of data.

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries [20]. The purpose of using such methodology to represent knowledge is to enable the re-usability of the ontology and facilitate the interoperability across multiple and heterogeneous systems. OWL represented knowledge can easily be machine interpretable by other applications. This eases the integration of our model with other applications.

With more applications being based on OWL, others may arise that do not depend exclusively on internal application logic to represent relationships between data from several distinct sources of knowledge. Although this sounds intuitive, we are far from reaching this level of integration, since information is typically stored in particular formats within organizations and closed within each application. This approach opens the OWL expressiveness to the network model representation, allowing new data relationships and management applications to emerge. Future development also benefits from this approach.

This model is adopted in this project, since future network management applications will use several sources of information that typically are not used in today's approaches to network management software. Our approach addresses the limitation that resulted in today's network management software limitations: the lack of relationship between several sources of information that happen to be in the same knowledge domain.

Although not used in demonstration application, inference engines and rule-based engines can be used in the future to complement the model. These engines are used to derive new assertions in the model, taking advantage of the expressiveness of the knowledge representation.

### 3.4.2 Class Hierarchy

Figure 3.3 depicts the class hierarchy of the ontology developed for the Management Framework and used in the demonstration application. Top-level classes represent the most important concepts identified in the network management domain and that are required to develop the proof-of-concept application for QoS management. All these classes or concepts, have properties that have connected them to each other, representing the cardinality of the required relations and the expressiveness needed to represent the knowledge domain of network management.

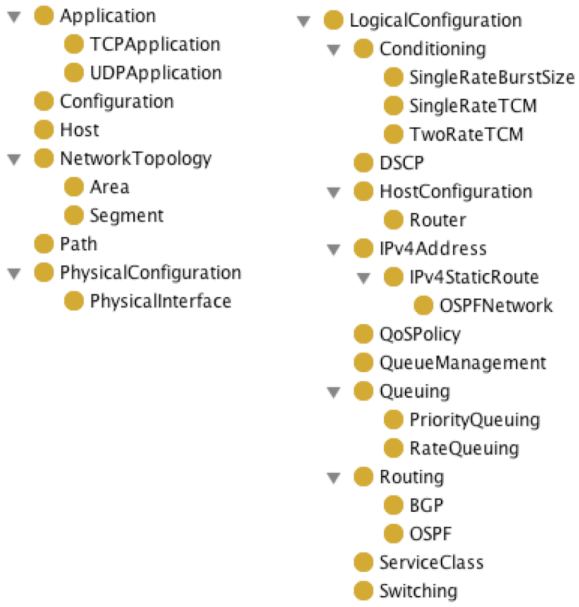


Figure 3.3: Framework Ontology

- Application
  - This concept allows the association of a given TCP or UDP application to the corresponding QoS Service Classes.
- Configuration
  - The Configuration concept is defined to enable the storage of several versions of a Logical and Physical Configuration for a given Host. Although unused in the demonstration application since it is out-of-scope, a visualization mechanism of configuration history for a given Host can be one of the most important outcomes of storing Hosts' configurations using this approach;
- Host

- A Host represents a Network Element. It holds several Configuration instances and its individuals are unique in the model. The name Host allows the creation of individuals of this class that may not be active networking elements. For instance, application, database or other kinds of servers may be represented in the future using this class;

- Network Topology
  - The application logic developed in Q-Andrew uses this class to instantiate the network interconnections existing in parsed router configurations. This process is explained in Section 3.5;
  - Area subclass represents a routing area, routing domain or a DiffServ administrative domain;
  - Segment subclass is, in essence, a network prefix and a netmask. Logical IP interfaces of a router connect to a given Segment individual;

- Physical Configuration
  - In the proposed ontology, the Physical Configuration concept only represents the existing physical interfaces for which Logical Configurations, such as IP Address and Netmask, are defined;
  - Further development can be made to include the Host physical configuration. For instance, high-grade routers are modular pieces of equipment and this information may be required for other management scenarios;

- Logical Configuration
  - The concept Logical Configuration contains many subclasses. This is necessary as Hosts' configurations have to be stored with the necessary information in order to develop the QoS management application;
  - IP, Routing, QoS architecture and QoS mechanisms are represented as subclasses of Logical Configuration.

### 3.4.3 Limitations of the Model

The limitation of this model is scalability, given the added computation overhead and additional storage space required to handle this kind of knowledge representation. To overcome this limitation, we can resort to an Oracle database that is able to store OWL ontologies' format. This will dramatically improve the availability and access performance to stored data, thus providing an effective way to store this kind of information without the limitations of traditional relational databases.

## 3.5 Graph

In order to create the network Graph, information about the network connectivity of each router is extracted from the XML versions of the parsed configurations as described in Section 3.2. Both physical and logical configuration information is used to construct the graph.

### 3.5.1 Definitions

The Network Graph is composed of two types of Nodes, and by Edges that interconnect them:

- A Graph Node is a Host (H-Node) or a Network Segment (S-Node);
- A Graph Edge can only connect a H-Node to a S-Node.

A H-Node represents an ontology individual of the type Host. A S-Node represents an ontology individual of the class Network Topology/Segment. A S-Node is identified by its IP network address and network mask.

#### 3.5.1.1 Remarks

- Since most real network links are bi-directional, the represented Edges are assumed to be bi-directional;
- Since IP Routing is used, a Connected Graph does not necessarily mean that an IP Path exists from a Node to any other given Node in the Graph.

### 3.5.2 Graph Connections

Graph connections are, essentially, network connections at the IP layer. Since many physical interfaces, such as Ethernet, support more than one IP logical address configured in the same interface, these connections are represented in the graph individually. Figure3.4 shows a Q-Andrew visualization detail of two routers, d1 and d2, with 2 and 3 logical interfaces respectively. Network nodes are depicted with their network prefix and mask information. H-Node router labeled d1, connects to S-Nodes network segments, labeled 10.41.1.0/24 and 10.42.1.0/24. The H-Node labeled d2, connects to S-Nodes labeled 10.42.1.0/24, 10.44.1.0/24 and 192.168.40.0/24. H-Nodes are connected in terms of

the network Graph representation. This connection does not necessarily imply that traffic from/to d1 passes through d2.

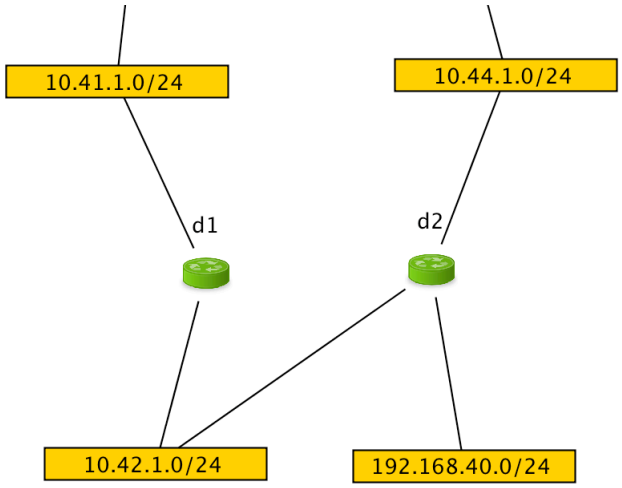


Figure 3.4: Q-Andrew Graph Visualization (detail)

#### 3.5.2.1 Private Networks and VPNs

In large networks, the private network address ranges stipulated in current practices [21] (Table 3.1), can appear in distinct topological locations. This may happen due to the common use of Network Address Translation functionalities in routers and firewalls and their use with Virtual Private Networks (VPNs) to extend private networks to another location (NAT [22]). Setting the connectivity as described earlier would result in an erroneous Graph wherever the private ranges are re-used. This issue is not addressed in the demonstration application, but can be mitigated using monitoring information from the physical interfaces. Observing which Ethernet addresses exist in each physical interface, a simple mechanism can be used to eliminate the logical IP interfaces that do not have matching Ethernet addresses in the forwarding table of a switch or ARP table in a router. Since the latter is not a reliable method to clearly identify this issue, active monitoring measures must be taken into account. With active monitoring facilities, paths and interconnections can be determined using for instance, specific *marked* packets or flows to obtain or confirm topology information.

| Address Range | Prefix |
|---|---|
| 10.0.0.0 - 10.255.255.255 | 10/8 |
| 172.16.0.0 - 172.31.255.255 | 172.16/12 |
| 192.168.0.0 - 192.168.255.255 | 192.168/16 |

Table 3.1: Address Allocation for Private Internets

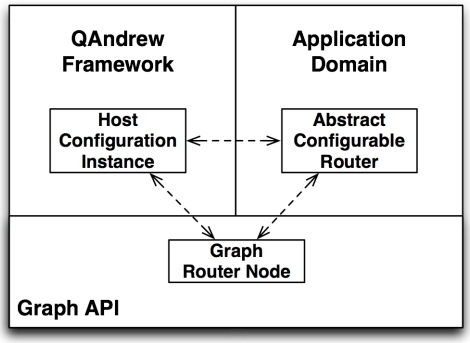


Figure 3.5: Internal Host Representation

### 3.5.3 Graph Node Associated Information

Figure 3.5 depicts the internal representation of a single Host in the framework, at the application layer. The abstract model is used to create a Host Configuration Instance at the application level. This instance represents for example, the latest network element configuration. Since we store the configuration history of a network element, this can be later used for other types of management applications. Q-Andrew only uses the most recent network element's configuration.

The Graph Node is used with a Graph API where the network topology is created. Finally, the Abstract Configurable Router is created dynamically according to the Router's Vendor, Operating System Version and applicable QoS Policy. This instance is used to construct the configuration script according to the network element's vendor. This dynamic approach is adaptable to other network management configuration mechanisms besides QoS.

## 3.6 Framework Implementation

Configurations are imported to the model using a parser developed in Perl. Other functionalities are implemented using Java Programming Language. The OWL Ontology is handled using the Protégé and Jena APIs [23, 24].

# Chapter 4

# Q-Andrew Application

The Q-Andrew Application uses the Management Framework defined in Chapter 3 to create a tool for consistent QoS policy deployment in a heterogeneous network. Firstly, the generic development process is described, introducing the application architecture and the Simulation Topology used during the development process. Secondly, the required QoS configuration mechanisms are described, as well as the QoS Reference Policy used to illustrate the configuration generation process. Finally, Q-Andrew application is used to show the configuration script results for a small demonstration network topology, where a QoS policy is deployed using the proper mechanisms to deploy them.

## 4.1 Application Architecture

The application architecture is shown in Figure 4.1. Router configurations are imported to the framework's abstract model. The Q-Andrew demonstration application uses the framework to provide the necessary operations for QoS management. The different kinds of hosts represented internally in the application are used for different purposes (Section 3.5.3). The resulting configuration script can be analyzed or deployed directly into the router. The GUI shows the logical network topology, taken from the configurations imported into the framework's model.

## 4.2 Simulation topology

To use configurations from the Service Provider mentioned in Section 1.1, extensive work is necessary to identify every missing router configuration that is available. Several types of devices are found in a large network, resulting in missing connections that are handled by firewalls, bridges or even other routers from which it is difficult to extract configuration

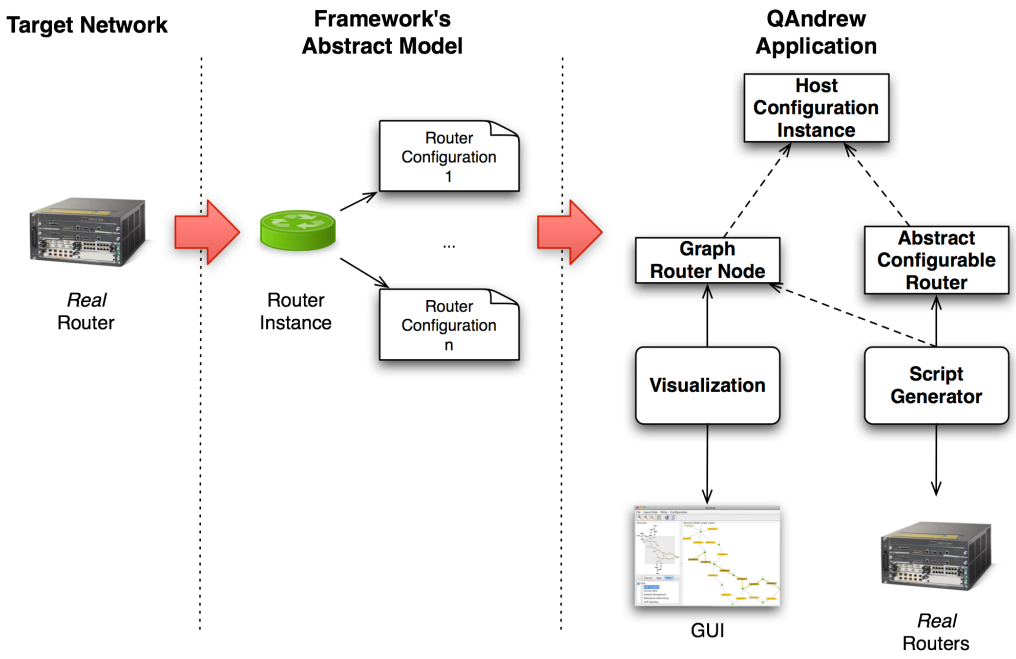


Figure 4.1: Application Logic Architecture

information. For this reason, the configurations from this Service Provider are only used to implement and test the parser quality for Cisco devices. To implement the methods proposed in this document, the simulation topology is used, ensuring control over the development environment and that the Complete Model assumption is met.

The simulation topology consists of twenty routers with multiple IP connections. The script-based configurations are created in Cisco IOS format. Once the scripts are parsed and imported into the model, the Q-Andrew application reads the configuration independently from the original vendor configuration format. When a configuration is required from a Juniper Network's router, the instance is changed directly in the model, updating the corresponding vendor information. With these changes, implementations are tested in a fast and effective way, maintaining the compatibility with the principles described in this report. Some of these configurations are shown in Section 4.7, where the results of using Q-Andrew are presented.

Figure 4.2 shows the simulation topology graph from Q-Andrew Application.

## 4.3 Path Configuration

A DiffServ Domain policy deployed in a Service Provider's network is usually very consistent across a large number of network elements. This ensures that the QoS policy is the

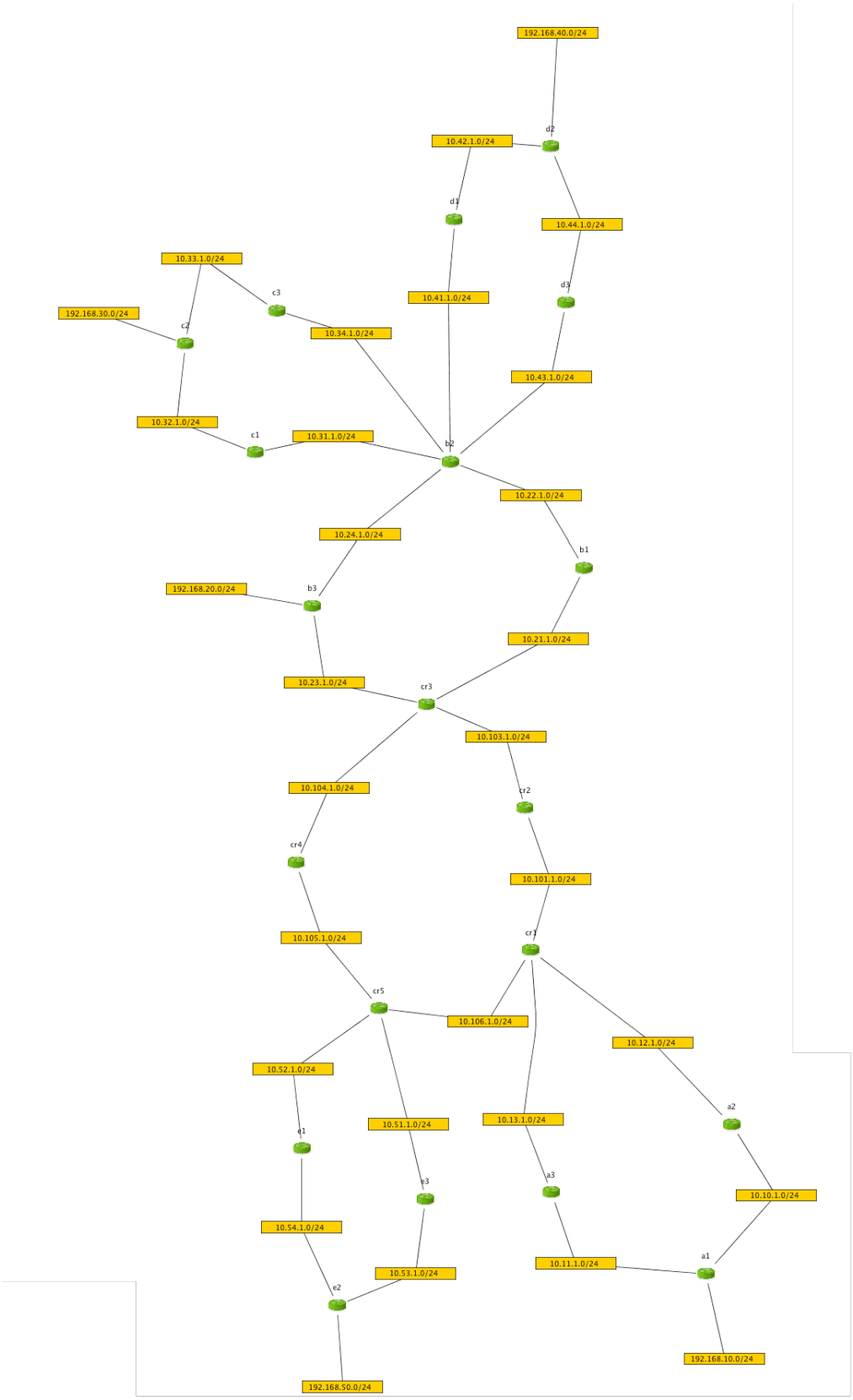


Figure 4.2: Q-Andrew Graph-based Topology

same in a Domain. A theoretical approach would identify these consistent domains as independent routing domains. However, in large domains, several routing protocols are used within the same DiffServ Domain. This limits the automated search for DiffServ domains, where ingress and egress policies would be adopted according to the different routing domains crossed. In such a dynamic scenario, we opted to identify the most elementary QoS configuration operation. This operation is then used multiple times to create a DiffServ policy in a configuration domain composed by several routing protocols. This operation is the configuration of a single, unidirectional path in the network, given a source, destination network and netmask as well as the service requirements to be serviced by this path. In Q-Andrew, the path is created in the application GUI, using data from user input.

#### Implementation

In the reference topology, every node is configured with static routing, meaning that every node belongs to a single, static-routing domain as defined in Section 4.4.3. Each path is calculated using a route search algorithm (LC-TRIES [25]), using the graph-network model described in Section 3.5.

## 4.4 DiffServ Configurable Domains

### 4.4.1 DiffServ Domain

DiffServ domains are defined as a set of routers and network segments where a consistent QoS policy is applied. The policy to apply is an administrative domain policy or a path policy. When an Administrative Domain policy is configured, predefined sets of routers are configured using the same global policy. In a path policy, the network operator specifies a source and a destination network for a given QoS policy. The routers to configure must be found dynamically, using routing configuration information from the model. The QoS policy to apply is determined using the application specified by the network operator and using the corresponding service class configurations as described in Chapter 2.

### 4.4.2 Administrative Domains

Administrative Domains have no special constraint regarding their composition. An implementation of these kinds of domains requires the network operator to specify which devices belong to each administrative domain.

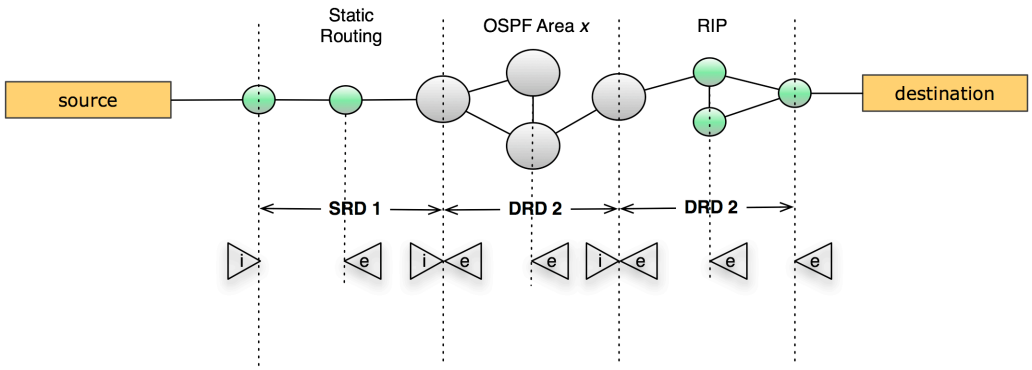


Figure 4.3: QoS Policies across multiple Routing or Administrative Domains

#### Implementation

The model includes classes and properties to represent these kinds of domains. The Q-Andrew application does not implement the configuration of Administrative Domains.

### 4.4.3 Static Routing Domains (SRD)

Domains that only use static routing to perform IP forwarding are identified as Static Routing Domains. A path in these kinds of domains is identified using the static routing information.

#### Implementation

The Q-Andrew Application demonstrates the configuration of Static Routing Domains.

### 4.4.4 Dynamic Routing Domains (DRD)

DiffServ domains are identified using dynamic routing information from the configuration files. The topology of a given network area can change according to route updates from dynamic routing protocols. The path to configure is defined taking this possibility into account. All routers belonging to a given area (such as an OSPF or IS-IS area) are configured using the same QoS policy.

Figure 4.3 shows a scenario where multiple routing domains are crossed between the source and destination networks of the path. The first domain (SRD1) is a static routing domain and the path in this domain is also static. The second and third domains (DRD1 and DRD2), belong to dynamic routing domains and all the routers from the same area

must be configured, since any router in these areas can forward traffic for this path. Additionally, at domain crossing points, the egress (e) and ingress (i) policies for the domains are different. This ensures that different routing domains apply a policer at the domain edges. Routers with all the interfaces within the same routing domain apply the same egress (e) policy.

#### Implementation

Dynamic routing information from OSPF and RIP protocols is imported to the model. Other protocols, such as BGP and ISIS are not imported. The Q-Andrew application does not implement the configuration of Dynamic Routing Domains.

## 4.5 Reference QoS Policy

The application developed to demonstrate the framework already includes in its model a set of definitions of Service Classes (Table 4.1) and QoS mechanisms (Table 4.2) for demonstration purposes. These definitions are used to create the consolidated QoS policy across a network composed of devices from vendors Cisco and Juniper. The Application types are defined to represent different kinds of requirements found at the Service Provider's network.

| Service Class | Loss Tolerance | Delay Tolerance | Jitter Tolerance | DiffServ PHB |
|---|---|---|---|---|
| Network Control | Low | Low | Yes | CS6 |
| VoIP Transport | Very Low | Very Low | Very Low | EF |
| VoIP Signalling | Low | Low | Yes | CS5 |
| IPTV Video | Very Low | Medium | Low | CS3 |
| OAM | Low | Medium | Yes | CS2 |
| Low-latency | Low | Low-Medium | Yes | AF21, AF22, SF23 |
| High-Throughput | Low | Medium-High | Yes | AF11, AF12, AF13 |
| Standard | Not specified | Not specified | Not specified | CS0 |

Table 4.1: Service Classes

### 4.5.1 QoS Service Classes

The QoS Service Classes created (Table 4.1) are based on the configuration guidelines presented in [26], which use the QoS mechanisms described in Section 2. This RFC recommends a set of DiffServ Service Classes for several possible deployment scenarios. In the next Sections, DiffServ classes are described according to the generic requirements for the target Service Provider network. Additional service classes can be added to the

model, or each service class defined can be changed to better suit the reality of the network being configured.

| Service Class | Conditioning at DS Edge | Queuing | AQM |
|---|---|---|---|
| Network Control | sr + bs | Rate | Yes |
| VoIP Transport | sr + bs | Priority | No |
| VoIP Signalling | sr + bs | Rate | No |
| IPTV Video | sr + bs | Rate | No |
| OAM | sr + bs | Rate | Yes |
| Low-latency | srTCM | Rate | Yes |
| High-Throughput | srTCM | Rate | Yes |
| Standard | - | Rate | Yes |

Table 4.2: Service Differentiation Mechanisms
AQM = Active Queue Management; sr = Single Rate; bs = Burst Size; srTCM = Single Rate - Three Color Marker.

### 4.5.2 Quality Requirement of Service Class

Applications that are user-interactive or delay-sensitive require special attention when generating the configuration for each Hop in a traffic path. Voice-over IP and IPTV traffic paths are correctly configured if we take into account their specific requirements. The goal is providing appropriate quality of service to the transport of such traffic ensuring that the end-user is not affected by temporary glitches on the network. In terms of QoS configuration, we present the approach used when generating the QoS configuration scripts in Sections 4.5.2.1, 4.5.2.2 and 4.5.2.3. Other issues such as availability, switching and routing delays due to network failures or path reconfiguration are not addressed, but we give enough flexibility to the network administrator to change the global upper-bounds defined according to the characteristics of the targeted network. Table 4.3 describes the values from the network operator using Q-Andrew, where which service requirement is identified, either in absolute bandwidth units or in percentage relative to the total physical interface bandwidth of a given network element.

| Service Class | Configuration Requirements | Default |
|---|---|---|
| Network Control | Link Bandwidth %<br>or Bandwidth kbps | 5% of Link bandwidth |
| VoIP Transport | Number of Concurrent Sessions<br>Bandwidth kbps per Session | - |
| VoIP Signalling | (Number of Concurrent Sessions) | 5% of VoIP Transp. bw. |
| IPTV Video | Number of Concurrent SD Streams<br>Number of Concurrent HD Streams | - |
| OAM | Link Bandwidth %<br>or Bandwidth kbps | 5% of Link bandwidth |
| Low-latency | Link Bandwidth %<br>or Bandwidth kbps | - |
| High-Throughput | Link Bandwidth %<br>or Bandwidth kbps | - |
| Standard | Link Bandwidth %<br>or Bandwidth kbps | Remaining bandwidth<br>or 5% of Link bw. |

Table 4.3: Quality Requirement of Service Class

#### 4.5.2.1 VoIP Transport Service Class

The ITU-T G.114[27] recommendation suggests that a 150ms delay provides a good user quality mouth-to-ear experience. More than 400ms of delay is considered unacceptable. Since we are automating the configuration generation for multiple hops in a path, it is hard to determine the exact delay each hop introduces when forwarding traffic, which is valid for every service class. When forwarding VoIP packets, we need to commit to the lowest delay possible. When generating a configuration for multiple hops, one way to determine this is to establish a maximum number of hops allowed in a VoIP path. Delay in a VoIP conversation can be broken down into four components:

1. Encoding delay:
   (a) Representing the delay introduced by the encoder to convert the voice samples into a digital signal;
   (b) Depends on the codec used, but it is considered constant.
2. Network delays:
   (a) Fixed network delays:
      i. The minimum propagation time that a packet takes from source to destination, and can be considered constant;
   (b) Variable network delay:

i. Congestion and Serialization delay across the path.

3. Decoding Delay:

   (a) The correspondent decoding of the packets into an audible stream;

   (b) Depends on the codec used, but constant for a given codec.

Variable network delays are difficult to calculate, so we define a global parameter, representing the total delay contribution from all the hops in a path. This parameter is tuned up or down by the network operator according to the targeted network. To define a reasonable value for this parameter, we resort to simulation results from [18], where it is demonstrated that a random call arrival at a link with 80% VoIP transport load, using a G.729A codec, with a 20ms voice sampling rate, introduces less than 7ms delay for a 512kbps link and less than 1ms for a 10Mbps link. With this reference, we defined the upper-bound value at 10ms for the maximum hop delay contribution in a VoIP transport path. For example, when the network operator configures a path with more than 15 hops, a warning is issued stating that 15 hops can introduce more than 150ms (10ms * 15 hops) of delay in the path. Another important piece of information is the expected number of concurrent VoIP sessions in a given path. A warning is also issued if the number of concurrent VoIP sessions specified may not be serviced by this path. This can happen if some hops in the path do not have the bandwidth available for this number of concurrent sessions, either from link capacity or because it has more service classes already reserving the bandwidth required. For the VoIP Transport service class, some additional calculations are made to determine if a path can be configured. The ITU-T Codec G.711 has a bit rate of 64 kbps, with 50 pps, or 160 bytes. IP overhead is 40 bytes and Ethernet 38 bytes, representing a total of 238 bytes. The bandwidth required for a single session, with no noise reduction, is (160+40+38)*50*8 = 95,200 kbps. If a network operator needs to configure 120 concurrent VoIP sessions in a 10Mbps link, this codec will require more than 11Mbps and thus, a warning is issued. These calculations are executed for other classes as well, considering the bandwidth rates they require (Table 4.3).

#### 4.5.2.2 IPTV Service Class

Video broadcast services require a different approach than VoIP transport. Video broadcasting using IP uses IP Multicasting to deliver the same channel to several customers. Video-on-Demand (VoD) uses IP Unicast. The number of frames required for a video stream is not constant over time and varies according to the encoding used. IPTV is currently being deployed in Portugal and uses the latest MPEG4 (Moving Picture Experts Group version 4) standard [28, 18, 29]. MPEG frames vary according to the variability of the original video. Frames may represent a complete frame, only the main changes from

the previous frame or use the previous sent frame as reference for the current frame. A video stream delay is divided into several components: network delays, multicast processing and several set-top box components. A set-top box is the communications end-point at the customer site, receiving the IPTV broadcast or VoD stream. The set-top box handles frame buffering and frame delay. Additionally, the set-top box introduces more delay if Forward Error Correction (FEC) and encryption are used. In the total delay, the variable network propagation and variable delay represent only a small fraction of the total delay expected for this service class. For this reason, what we need to ensure is that the total number of concurrent MPEG streams is serviced without influence from other classes. To ensure this property, we ask the network operator to specify the total number of concurrent video streams in a given path. Note that this parameter is not easy to determine: the number of concurrent video streams varies according to the network link location, multicast streams, VoD unicast streams and video quality. The European Service Provider's example with the Microsoft IPTV solution, uses two different video qualities: Standard Definition (SD), requiring a maximum of 1.8Mbps and High-Definition (HD, 720p) requiring a maximum of 9Mbps [28]. We handle this issue by asking the network operator for the number of concurrent SD and HD video streams required in a given path. With these definitions, the test for over-subscription is also possible.

#### 4.5.2.3 Other Service Classes

The remaining service classes include different types of applications with different requirements. To correctly automate the configuration in this scenario, each application must be associated with the most suitable available service class previously defined. There are situations where new or specific applications require special handling, as is the case of VoIP and IPTV. In this situation, new service classes need to be defined. The reference QoS Policy defined earlier only includes the classes needed to define a set of criteria required for the implementation of the Q-Andrew demonstration application. To define new service classes, the model has to be enriched with new definitions and the applications that use the model, updated with new user dialogs where different inputs need to be requested of the network operator. The service classes that fall into this category are configured with the network operator input, where the total bandwidth usage has to be defined. Applications belonging to such service classes are then configured and validated against the rest of the classes ensuring that link oversubscribing does not occur. Table 4.3 summarizes the default values of some classes as well as the inputs required from the network operator. Default values are global parameters and can be updated by user intervention. Default bandwidths are defined for Network Control and Operations and Management Service Classes to ensure that a minimal QoS configuration exists for these critical classes. The value is expressed in percentage terms to facilitate the adoption of several link capac-

ities in a path. Although it may not always be the case, edges tend to have smaller links than core links, but traffic aggregation is also much larger as we get closer to a network core. Note that the Standard Service Class defaults to the remaining bandwidth. As the configuration of paths evolves, the value of the default service class can reach a minimum of 5% if the other service classes are configured with the total amount of bandwidth. Again, the network operator can update this value.

## 4.6 Configuration Generation Process

To support multiple vendors, an object-oriented approach allows the configuration script to be generated after successive path configurations. This approach facilitates the adoption of more vendors' script formats. This is achieved by abstracting common properties of a configuration into common, abstract classes.

Consider the Cisco ACL/ Juniper Filter generation process described in Section 4.6.1. Both require the definition of a source and destination to match a particular packet or packets. Cisco ACLs are numbered, but Juniper Filters are not. This difference is handled by the use of inheritance from the *AbstractHost* class. The *CiscoHost* class inherits all the properties and methods of an *AbstractHost*, but implements a counter for ACL numbering. The same thing is not required in Juniper. The object-oriented approach for the internal host configuration greatly simplifies the script generation process. Since all the configurations are added to the host's instance as each path is configured, the final result is a network element instance with all the required information stored internally in a highly structured way. Using the same inheritance properties described earlier, the method *getConfigString* is defined in each abstract class. This method implements the required steps to generate a script for the correspondent instance. For each class that inherits an abstract class, this method is required to be implemented. For a given vendor, the script generator method is implemented according to the correspondent vendor script format. The final configuration script is generated by the class that inherits from *AbstractHost* class, issuing the script generator methods in the order required to get a correct host configuration script.

Once these properties are identified, each vendor generation process is developed. Q-Andrew implements this approach for Cisco IOS 12.4 and Juniper JunOS 9.3. Each QoS Mechanism from Chapter 2 is configured and the resulting scripts are shown in the next section for both vendors' script format. A full configuration example is shown in Section 4.7.

Network addresses and ports used in the next sections have no particular meaning, they are used for illustration purposes.

### 4.6.1 Classification and Marking

The following paragraphs show the classification and marking at the DiffServ domain edge, from 192.168.10.0/24 to 192.168.20.0/24, using the service VoIP. For simplicity, assume that VoIP transport only uses the port UDP 16000. VoIP Transport software generally uses dynamic ports. This port is used to demonstrate the Cisco ACL and Juniper Filter scripts generated. Since the VoIP transport application belongs to the Service Class with same name, the corresponding DSCP marking is EF (Table 4.1). The physical interfaces are determined using the information from the network-graph (Section 3.5).

The complete path from 192.168.10.0/24 to 192.168.20.0/24, static routing is used as described in section 4.3, including every physical interface that is used in the path. This way, the hops and corresponding interfaces requiring configuration, are known when the script is generated.

To configure ACLs and Filters for dynamic port protocols, additional settings can be made. These settings enable the router to identify the protocol used in a given packet. These settings vary greatly according to the vendor and the software version. To limit simplify the description, we do not show these settings.

**Cisco IOS ACLs[1]:**

```
interface FastEthernet0/0
  ip address 192.168.10.1 255.255.255.0
    service-policy input ingress-polmap
interface FastEthernet0/1
  ip address 10.10.1.1 255.255.255.0
  service-policy output egress-polmap
access-list 10 allow tcp 192.168.10.0 255.255.255.0 \
              192.168.20.0 255.255.255.0 eq 16000
policy-map ingress-polmap
  class VoIP-ingress
  set ip dscp EF
policy-map egress-polmap
  class VoIP-egress
class-map match-any VoIP-ingress
  description VoIP ingress class map
  match access-group 10
class-map match-any VoIP-egress
  description VoIP Egress
  match dscp EF
```

[1]The symbol '\' represents a command that continues in the following line.

**Juniper JunOS Firewall Filter rules**

```
set firewall family inet filter SetDSCP term \
  MarkVoIPTransport from address 192.168.0.10/24
set firewall family inet filter SetDSCP term \
  MarkVoIPTransport from destination-address 192.168.20.0/24
set firewall family inet filter SetDSCP term \
  MarkVoIPTransport from protocol udp
set firewall family inet filter SetDSCP term \
MarkVoIPTransport from destination-port 16000
set firewall family inet filter SetDSCP term \
MarkVoIPTransport then dscp ef
set interfaces fe-0/0/0 unit 0 family inet \
  filter input SetDSCP
set firewall family inet filter MatchDSCP term \
  ClassifyVoIPTransport from dscp ef then forwarding-class ef;
set interfaces fe-0/1/0 unit 0 familiy inet \
  filter output MatchDSCP
```

### 4.6.2 Queuing

Queuing management mechanisms are applied at every hop's egress interface.

#### 4.6.2.1 Cisco Priority Queuing (PQ)

Cisco implementation of PQ is not round robin: when there are packets in the high-priority queue, they are sent before any packets in lower-priority queues. If too many traffic types are configured to go into the high and medium queues, packets in the normal and lower-priority queues may never be sent and link queue starvation may occur.

The *list* argument to the *priority-list* command indicates the access control list number 10 configured in 4.6.1 to identify traffic belonging to this queue.

```
priority-list 1 protocol ip high list 10
[...]
interface fastEthernet0/0
  priority-group 1
```

| Priority Queue Argument | Packet Limits |
|---|---|
| high-limit | 20 |
| medium-limit | 40 |
| normal-limit | 60 |
| low-limit | 80 |

Table 4.4: Cisco IOS 12.4 Default Priority-Queue Limits

#### 4.6.2.2 Juniper Forwarding Class Queues

Juniper already has defaults to the several priority queues, according to their DSCP marking. The table 4.5 shows the default configuration for a Juniper Router running JunOS version 9.3. This configuration is not directly comparable with Cisco behavior using Priority Queues. The difference between these vendors is handled choosing the more suited configuration of each approach for each Service Class. Table 4.6 shows the options taken when creating automated configuration scripts for Cisco and Juniper. Although Q-Andrew does not have any interface to configure the global behavior, the framework-model includes a set of classes to instantiate these configurations. This way, further development can easily update the interface to allow a network operator to change the default behavior.

| Queue | Forwarding Class Name | See notes in section4.6.2.2 |
|---|---|---|
| Queue 0 | best-effort (be) | 1 |
| Queue 1 | expedited-forwarding (ef) | 2 |
| Queue 2 | assured-forwarding (af) | 3 |
| Queue 3 | network-control (nc) | 4 |

Table 4.5: JunOS 9.3 Defaults

Table 4.5 notes, from the vendor:

1. The software does not apply any special CoS handling to packets with 000000 in the DiffServ field, a backward compatibility feature. These packets are usually dropped under congested network conditions.

2. The software delivers assured bandwidth, low loss, low delay, and low delay variation (jitter) end-to-end for packets in this service class. Routers accept excess traffic in this class, but in contrast to assured forwarding, out-of-profile expedited-forwarding packets can be forwarded out of sequence or dropped.

3. The software offers a high level of assurance that the packets are delivered as long as the packet flow from the customer stays within a certain service profile that you define. The software accepts excess traffic, but applies a RED drop profile to determine if the excess packets are dropped and not forwarded. Depending on platform type,

| Service Class | Cisco | Juniper |
|---|---|---|
| Network Control | medium | queue 4 |
| VoIP Transport | high | queue 2 |
| VoIP Signalling | medium | queue 3 |
| IPTV Video | medium | queue 3 |
| OAM | low | queue 3 |
| Low-latency | low | queue 3 |
| High-Throughput | low | queue 3 |
| Standard | low | queue 1 |

Table 4.6: Queuing - Global Parameters

up to four drop probabilities (low, medium-low, medium-high, and high) are defined for this service class.

4. The software delivers packets in this service class with a low priority. (These packets are not delay sensitive.) Typically, these packets represent routing protocol hello or keep-alive messages. Because loss of these packets jeopardizes proper network operation, delay is preferable to discard.

#### 4.6.2.3 Remarks

The Juniper scheduler is configured using Weighted Round Robin (WRR), which is not a strict priority mechanism. The use of Weighted Round Robin by Juniper is described in the next section.

### 4.6.3 Rate Queuing

Rate queuing mechanisms are configured based on default-global parameters or using the values indicated by user input. In this example, a HTTP path is configured, reserving 15% of total interface bandwidth in a path. In our reference QoS Policy, HTTP Application belongs to the service class Low Latency. If other applications are configured in the same path, the corresponding Rate Queuing configuration has to take the total amount of bandwidth required into account. This is achieved adding the total amount of possible bandwidth reserved either in percentage (relative to the interface bandwidth) or in total amount of reserved bandwidth units. To issue a warning as described in Section 4.5.2, the maximum amount of reservable bandwidth in a path is the minimum bandwidth found at all the interfaces in the path.

#### 4.6.3.1 Cisco IOS

Cisco implements Class-based Weighted Fair Queuing. Class-based WFQ allows the distinction between traffic that is handled with CB-WFQ and others that are not. If WFQ is used, the resulting Cisco script is very different, since the WFQ configuration is attached to the corresponding interface only. In the VoIP Transport Service Class, priority queuing is used instead of rate queuing. The following examples are for another service class, to demonstrate the differences between Cisco and Juniper configuration scripts.

```
policy-map PolicyMap-fastEthernet000
  class ClassLowLatency
    bandwidth percent 15
```

#### 4.6.3.2 Juniper JunOS

In Juniper, the VoIP Transport Rate Queuing mechanism is configured using a WRR algorithm. This is done by configuring a scheduler, associating it with a scheduler-map that maps the forwarding class to the scheduler and finally and assigning it to the corresponding outgoing interface.

```
set class-of-service schedulers Sched-LowLatency \
  transmit-rate percentage 15
set class-of-service scheduler-maps SchedMap-fe000 \
  forwarding-class assured-forwarding scheduler Sched-LowLatency
set class-of-service interfaces fe0/0/0 scheduler-map SchedMap-fe000
```

For each packet, the WRR algorithm follows the queue service order:

1. High priority, positive credit queues;
2. Low priority, positive credit queues;
3. High priority, negative credit queues;
4. Low priority, negative credit queues.

Positive credit ensures that a minimum bandwidth is given to a queue, according to its configured weight. Negative credit queues are served when one positive credit queue has not used its whole dedicated bandwidth and no more packets exist in a positive queue. The remaining bandwidth from positive queues is fairly shared between all the high priority/negative credit" queues until these become empty. If the high priority negative credit queues are empty and there is still some available bandwidth that can be allocated to packets, the low priority/negative credit queues will equally share it.

### 4.6.4 Active Queue Management

Random Early Detection (RED [12]) active queue management implementations differ between Cisco and Juniper. In Cisco, each class has its own RED profile associated with it. In Juniper the RED profile is configured in a scheduler, according to Packet-Loss Priority (PLP) defined for the PHB.

This approach assumes that the configuration parameter values are directly comparable. This may not be the case, since the underlying implementation greatly differs from both vendors. Since we are configuring equipment in a path, it is not expected, although it is technically possible, that a hop is joined by two network devices from different vendors. A hop may be a single equipment or may be composed of two or more devices using a primary-backup redundancy solution such as HSRP [30] or VRRP [31]. Since this is the most common scenario, RED configurations are not required to have exactly the same behavior. Instead, attention should be on delay as a general objective and not exclusively in configuration similarities.

Additional work is required to ensure the compatibility of RED profiles between any two vendors. In fact, even when two different platforms from the same vendor are considered, the results may differ greatly due to the particularities of certain hardware and software implementations. To ensure compatibility between router configurations, careful measurements must be made to ensure compatibility.

#### 4.6.4.1 Cisco

```
policy-map PolicyMap-fastEthernet000
  class ClassLowLatency
    random-detect dscp-based
    random-detect dscp af21 50 90 4
    random-detect dscp af22 45 90 4
    random-detect dscp af23 40 90 4
```

#### 4.6.4.2 Juniper

Table 4.7, shows the classifier configuration for Juniper Networks Router. For simplicity, only the table is shown and not the JunOS configuration script .

| DSCP | Forwarding Class | PLP |
|---|---|---|
| cs6 | NetworkControl-FC | low |
| cs2 | OAM-FC | low |
| af21-af23 | LowLatency-FC | low |
| af11-af13 | HighPriority-FC | high |
| cs0/be | Standard-FC | high |

Table 4.7: Juniper Classifier Configuration

The HTTP application used in this example belongs to the traffic class Low Latency. The corresponding DSCP marking is AF21, AF22, or AF23, according to user input. Table 4.1 shows which classes require the use of RED profiles and Table 4.7 shows the correspondence between the DSCP marking and the Packet Loss Priority of the forwarding class, used to identify the loss-priority in the following Juniper script that updates the previous scheduler configuration:

```
set class-of-service drop-profiles DropProf-LowLatency \
  interpolated-style-profile interpolate fill-level [50 90] \
  drop-probability [0 25]
set class-of-service schedulers Sched-LowLatency \
  drop-profile-map loss-priority low protocol any any \
  drop-profile DropProf-LowLatency
```

Both vendors have a different parameter for configuring the RED/CB-WRED packet-drop mark probability. Table shows the parameters for each vendor command and how they are accommodated in the model.

| | Juniper | Cisco | Model Value | Model Unit | Conversion |
|---|---|---|---|---|---|
| Minimum Threshold | min% | min% | min | % | - |
| Maximum Threshold | max% | max% | max | % | - |
| Mark probability | prob% | 1/MPD | MPD | integer | prob%=(1/MPD)*100 |

Table 4.8: RED Parameters for Cisco and Juniper

### 4.6.5 Additional Configurations

Traffic Conditioning, Policing/Admission control and Shaping configurations are generated using the same method from the previous sections. Since the reference QoS Policy only required the use of a simple Token Bucket (single rate + burst size control) and a Single Rate Three-Color Marker, the Q-Andrew application only implements these methods as a proof-of-concept. Additional methods can be developed using the object-oriented approach described earlier, extending the model to include additional configuration sections.

## 4.7 Configuration Results

Using a subset of the configurations from the demonstration topology (Table 4.9), the next sections present the process of creating a consistent QoS configuration using Network Elements from Cisco and Juniper.

| Hostname | Vendor | Interfaces | Routing Table |
|---|---|---|---|
| a1 | Cisco | FastEthernet0/0 192.168.10.1/24<br>FastEthernet0/1 10.10.1.2/24 | default = 10.10.1.1 |
| b1 | Cisco | FastEthernet0/0 172.16.1.1/24<br>FastEthernet0/1 10.10.2.2/24 | default = 10.10.2.2 |
| cr1 | Juniper | fe-0/0/0 10.10.1.1/24<br>fe-0/0/1 10.10.2.1/24<br>fe-0/0/2 10.20.1.1/24 | default = 10.20.1.2 |
| cr2 | Juniper | fe-0/0/0 10.30.1.1/24<br>fe-0/0/1 10.30.2.1/24<br>fe-0/0/2 10.20.1.2/24 | default = 10.20.1.1 |
| c1 | Cisco | FastEthernet0/0 192.168.10.1/24<br>FastEthernet0/1 10.10.1.2/24 | default = 10.30.1.1 |
| d1 | Cisco | FastEthernet0/0 192.168.10.1/24<br>FastEthernet0/1 10.10.1.2/24 | default = 10.30.2.1 |

Table 4.9: Network Elements' Configurations Summary

- FastEthernet and fe are 100Mbit Ethernet interfaces
- default represents the default gateway of each Network Element

The configurations are parsed and converted to a common XML format as described in Section 3.2. Cisco and Juniper XML configurations used in this Section:

**Cisco configuration converted to XML:**

```
<host name="a1">
<configuration vendor=”cisco” version="12.4" \
          date="2008-Ago-22" hour="11:50:18">
<interfaces>
        <interface name="FastEthernet0/0">
                <address>192.168.10.1</address>
                <netmask>255.255.255.0</netmask>
        </interface>
        <interface name="FastEthernet0/1">
                <address>10.10.1.2</address>
                <netmask>255.255.255.0</netmask>
        </interface>
</interfaces>
<routers></routers>
<staticroutes>
        <route destination="0.0.0.0" netmask="0.0.0.0" \
               through="10.10.1.1"></route>
</staticroutes>
</configuration>
</host>
```

**Juniper configuration converted to XML:**

```
<host name="cr1">
<configuration vendor=”juniper” version="9.3" \
          date="2008-Ago-22" hour="11:55:18">
<interfaces>
        <interface name="fe-0/0/0">
                <address>10.10.1.1</address>
                <netmask>255.255.255.0</netmask>
        </interface>
        <interface name="fe-0/0/1">
                <address>10.10.2.1</address>
                <netmask>255.255.255.0</netmask>
        </interface>
        <interface name="fe-0/0/2">
                <address>10.20.1.1</address>
                <netmask>255.255.255.0</netmask>
        </interface>
</interfaces>
<routers></routers>
<staticroutes>
        <route destination="0.0.0.0" netmask="0.0.0.0" \
               through="10.20.1.2"></route>
</staticroutes>
</configuration>
</host>
```

Once configurations are parsed and imported into the model as described in Section 3.2 and the network topology Graph is created as described in Section 3.5, the application

Q-Andrew can show the network graph in the GUI as depicted in Figure 4.2. The graph represents a subset of the simulation topology used to develop Q-Andrew, as identified in Section 4.2.

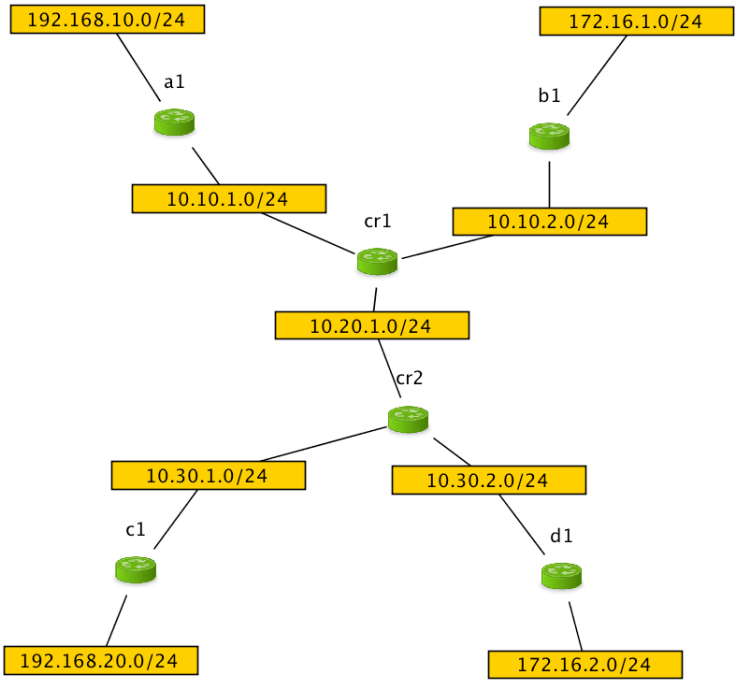


Figure 4.4: Demonstration Topology

Table 4.10 identifies the application services used to create the configuration presented in this Section. For simplicity, only 4 applications are defined. The QoS Mechanisms used in this policy, are identified in Section 4.5, where the correspondence between each service class and the mechanisms required to provide the proper PHB is identified.

| Service Class | Application | Service | Port, Port Type |
|---|---|---|---|
| VoIP Transport | VoIP Transport | VoIP | 16000, udp |
| VoIP Signaling | VoIP Signaling | H.323 Call Signaling | 1720, tcp |
| IPTV Video | Broadcast TV & VoD | IPTV Streaming | 5000, udp |
| OAM | Operations and Management | SSH | 22, tcp |
| Standard | External Access & Other applications | *any* | *any*, udp/tcp |

Table 4.10: Application Service types and Service Class association

- For simplicity, only one port is considered for protocols VoIP Transport and Broadcast TV, which use dynamic or multiple ports (Section 4.6.1).
- VoIP Signaling uses the ITU-T H.323 protocol [32];
- SSH protocol: Secure Shell protocol [33];
- *Any* matches all network addresses or ports.

Considering all the Network Elements as a single DiffServ Configurable Domain as described in Section 4.4, Table 4.11 shows a summary of the QoS Policy, created using several paths as proposed in Section 4.3.

| From | To | Services | |
|---|---|---|---|
| 192.168.10.0/24<br>172.16.1.0/24 | 192.168.20.0/24<br>172.16.2.0/24 | VoIP Transport:<br>H.323 Call Signaling:<br>IPTV Streaming:<br>SSH (OAM): | 100 sessions<br>default (5% transport bw.)<br>20 SD streams<br>default (5% link bw.) |
| 192.168.20.0/24<br>172.16.2.0/24 | 192.168.10.0/24<br>172.16.1.0/24 | VoIP Transport:<br>H.323 Call Signaling:<br>IPTV Streaming:<br>SSH (OAM): | 100 sessions<br>default (5% transport bw.)<br>20 SD streams<br>default (5% link bw.) |

Table 4.11: QoS Policy to Generate

- Service Classes described in Section 4.5:
  - VoIP Transport 100 Sessions: 100 * 95.2 kbps = 9.52 Mbps
  - VoIP Signaling: 5% of the total VoIP transport bandwidth 9.52 Mbps = 476 kbps
  - IPTV Streaming: Each IPTV SD Stream uses a maximum of 1.8Mbps. 20 * 1.8 Mbps = 36 Mbps
  - OAM Service Class uses 5% of total link bandwidth. Considering a 100 Mbps link = 5 Mbps

The resulting configuration is presented in the next paragraphs, were two representative examples are shown (the remaing configurations are similar). Network elements a1, b1,

c1 and d1 are configured with ingress policies, were traffic from the specified networks is marked according to the QoS policy defined in this Section. The cr1 and cr2 are Juniper network elements, configured with a different configuration script. The latter matches traffic already marked at ingress at the several networks, applying the proper configuration generation process as described in Section 4.6.

**Network Element a1 Configuration Script:**

```
interface FastEthernet0/0
  ip address 192.168.10.1 255.255.255.0
  service-policy input FE00-ingress-polmap
  service-policy output FE00-egress-polmap
  priority-group 1
interface FastEthernet0/1
  ip address 10.10.1.2 255.255.255.0
  service-policy input FE01-ingress-polmap
  service-policy output FE01-egress-polmap
access-list 10 allow tcp 192.168.10.0 255.255.255.0 \
               192.168.20.0 255.255.255.0 eq 16000
access-list 10 allow tcp 192.168.10.0 255.255.255.0 \
               172.16.2.0 255.255.255.0 eq 16000
access-list 20 allow tcp 192.168.10.0 255.255.255.0 \
               192.168.20.0 255.255.255.0 eq 1720
access-list 20 allow tcp 192.168.10.0 255.255.255.0 \
               172.16.2.0 255.255.255.0 eq 1720
access-list 30 allow tcp 192.168.10.0 255.255.255.0 \
               192.168.20.0 255.255.255.0 eq 5000
access-list 30 allow tcp 192.168.10.0 255.255.255.0 \
               172.16.2.0 255.255.255.0 eq 5000
access-list 40 allow tcp 192.168.10.0 255.255.255.0 \
               192.168.20.0 255.255.255.0 eq 22
access-list 40 allow tcp 192.168.10.0 255.255.255.0 \
               172.16.2.0 255.255.255.0 eq 22
priority-list 1 protocol ip high list 10
policy-map FE00-ingress-polmap
  class VoIP-ingress
    set ip dscp ef
  class VoIPSig-ingress
    set ip dscp cs5
  class IPTV-ingress
    set ip dscp cs3
  class OAM-ingress
    set ip dscp cs2
policy-map FE00-egress-polmap
  class VoIP-egress
    bandwidth 9520
  class VoIPSig-egress
    bandwidth 476
  class IPTV-egress
    bandwidth 36000
  class OAM-egress
    bandwidth percent 5
    random-detect dscp-based
```

```
    random-detect dscp af21 50 90 4
    random-detect dscp af22 45 90 4
    random-detect dscp af23 40 90 4
policy-map FE01-egress-polmap
  class VoIP-egress
    bandwidth 9520
  class VoIPSig-egress
    bandwidth 476
  class IPTV-egress
    bandwidth 36000
  class OAM-egress
    bandwidth percent 5
    random-detect dscp-based
    random-detect dscp af21 50 90 4
    random-detect dscp af22 45 90 4
    random-detect dscp af23 40 90 4
class-map match-any VoIP-ingress
  match access-group 10
class-map match-any VoIPSig-ingress
  match access-group 20
class-map match-any IPTV-ingress
  match access-group 30
class-map match-any OAM-ingress
  match access-group 40
class-map match-any VoIP-egress
  match dscp ef
class-map match-any VoIPSig-egress
  match dscp cs5
class-map match-any IPTV-egress
  match dscp cs3
class-map match-any OAM-egress
  match dscp cs2
```

### Network Element cr1 Configuration Script:

```
set firewall family inet filter MatchDSCP term \
  ClassifyVoIPTransport from dscp ef then forwarding-class ef;
  ClassifyVoIPSig from dscp cs5 then forwarding-class cs5;
  ClassifyIPTV from dscp cs3 then forwarding-class cs3;
  ClassifyOAM from dscp cs2 then forwarding-class cs2;
set interfaces fe-0/0/0 unit 0 familiy inet \
  filter output MatchDSCP
set interfaces fe-0/0/1 unit 0 familiy inet \
  filter output MatchDSCP
set interfaces fe-0/0/2 unit 0 familiy inet \
  filter output MatchDSCP
set class-of-service drop-profiles DropProfile-OAM \
  interpolated-style-profile interpolate fill-level [50 90] \
  drop-probability [0 25]
set class-of-service schedulers Sched-VoIPTransport \
  transmit-rate 9520000
set class-of-service schedulers Sched-VoIPSig \
  transmit-rate 476000
set class-of-service schedulers Sched-IPTV \
```

```
  transmit-rate 36000000
set class-of-service schedulers Sched-OAM \
  transmit-rate percentage 5
set class-of-service schedulers Sched-OAM \
  drop-profile-map loss-priority low protocol any any \
  drop-profile DropProf-LowLatency
set class-of-service scheduler-maps SchedMap-fe000 \
  forwarding-class expedited-forwarding scheduler Sched-VoIPTransport
set class-of-service scheduler-maps SchedMap-fe000 \
  forwarding-class assured-forwarding scheduler Sched-VoIPSig
set class-of-service scheduler-maps SchedMap-fe000 \
  forwarding-class assured-forwarding scheduler Sched-IPTV
set class-of-service scheduler-maps SchedMap-fe000 \
  forwarding-class assured-forwarding scheduler Sched-OAM
set class-of-service interfaces fe0/0/0 scheduler-map SchedMap-fe000
set class-of-service interfaces fe0/0/1 scheduler-map SchedMap-fe000
set class-of-service interfaces fe0/0/2 scheduler-map SchedMap-fe000
```

**Remarks:**

- The configuration script syntax is used as described in the public websites of vendors Cisco and Juniper [1, 2];
- Scripts are generated using the approach described in Section 4.6, where the Path Configuration process is described. A real deployment will use a more generic approach, which can simplify the scripts presented;
- Scripts were not tested in real devices since there were access limitations to real devices in order to confirm script correctness. This issue is mitigated by the simplicity of the management framework, which is easily updated or corrected.

## 4.8 Graph view

The network graph described can be visualized using the demonstration application. A graphical user interface (GUI) is developed, enabling the network operator to dynamically layout of the graph display. Additional facilities, such as graph printing are also available. Figure 4.5, shows the Q-Andrew application GUI.

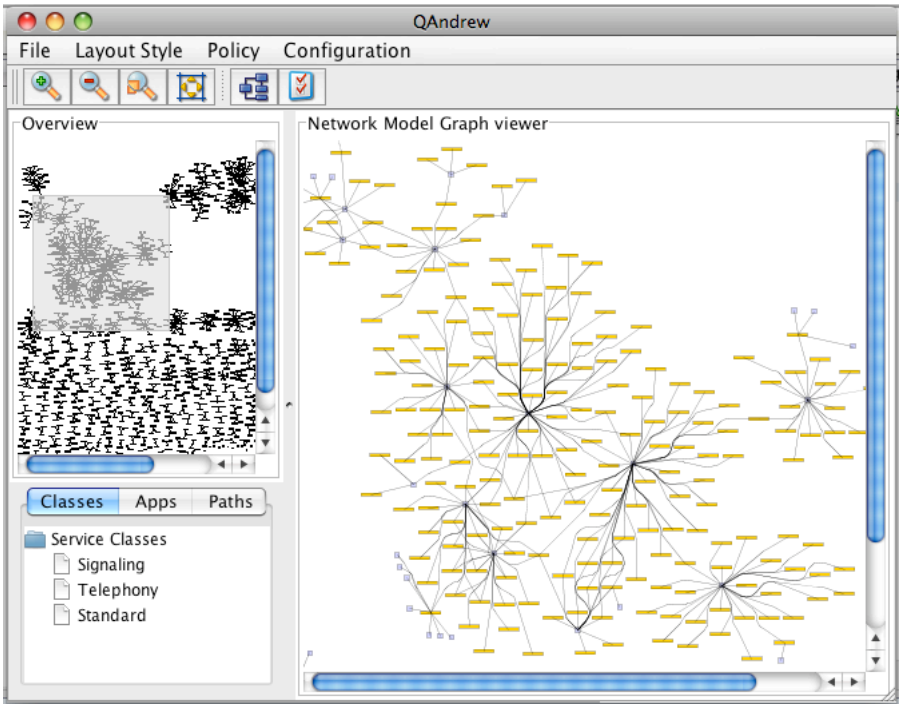


Figure 4.5: Q-Andrew GUI

# Chapter 5

# Service Provider Results

## 5.1 Service Provider Feedback

Feedback collected from the Service Provider representative is presented in the next Sections. The representative works for the Network Engineering and Service Deployment Department in this organization. His main responsibilities include: Network Planning and IP/ MPLS engineering of the Service Provider Networks.

### 5.1.1 DiffServ Routing Domains

*"DiffServ configurable domains should not be associated with a particular routing domain or dynamic routing protocol. Within the same DiffServ domain, we usually find several routing protocols running. The DiffServ configuration is common to several of these routing domains. For instance, local area networks interconnected through a WAN must be configured with the same QoS Policy, but the WAN network itself has a different QoS Policy."*

### 5.1.2 Statistics

*"The results of applying the configurations to the network should be directly observed and controlled in the application. Statistics should be provided to the network operator, allowing configurations to be adjusted according to the requirements. The application demonstrated uses the configuration to show the topology and to configure a set of devices in the network, but a feedback mechanism is missing. This mechanism is required to verify the policy correctness and what configurations need adjustments."*

### 5.1.3 Network Operations

*"Network operators do not feel comfortable with tools that automatically change configurations in the network. They are used to having more control in what is configured in the network. This is especially important for tools that automatically deploy configuration over a large number of network devices. This tool must give enough visibility and control to the operator, ensuring that every operation executed in the network can be controlled by the operator."*

### 5.1.4 Conclusions

*"I believe that the system proposed can be used in the future to manage our networks. Given the existing topology, protocols, devices and requirements, this is a complex task, but this system shows a practical way to handle this complexity."*

## 5.2 Network Configuration Completeness

To address the network configuration completeness requirement, reference configurations are created, allowing the model to be used with the Q-Andrew application. Although this is a limitation in this phase, this issue will be easily solved in a deployment scenario, as this would be a requirement for any configuration management solution. Given the time frame, it was not possible to obtain all the configurations, since this would require a continuous effort over an extended period of time.

### 5.2.1 Real Topology

A partial view of the complete topology of the Service Provider network is shown in Figure 5.1. Configurations from 319 hosts were parsed, imported to the model and shown in the Q-Andrew application. A simulation topology was used instead of this one for two reasons: firstly, as a proof-of-concept, only the Cisco parser was developed, given the complexity of developing a parser and the limited three-month period of the project. Secondly, the availability of a complete network configuration is not met. This results from the difficulties of the SP facilitating a complete set of configurations. This happened mainly due to logistical, time or privacy issues.

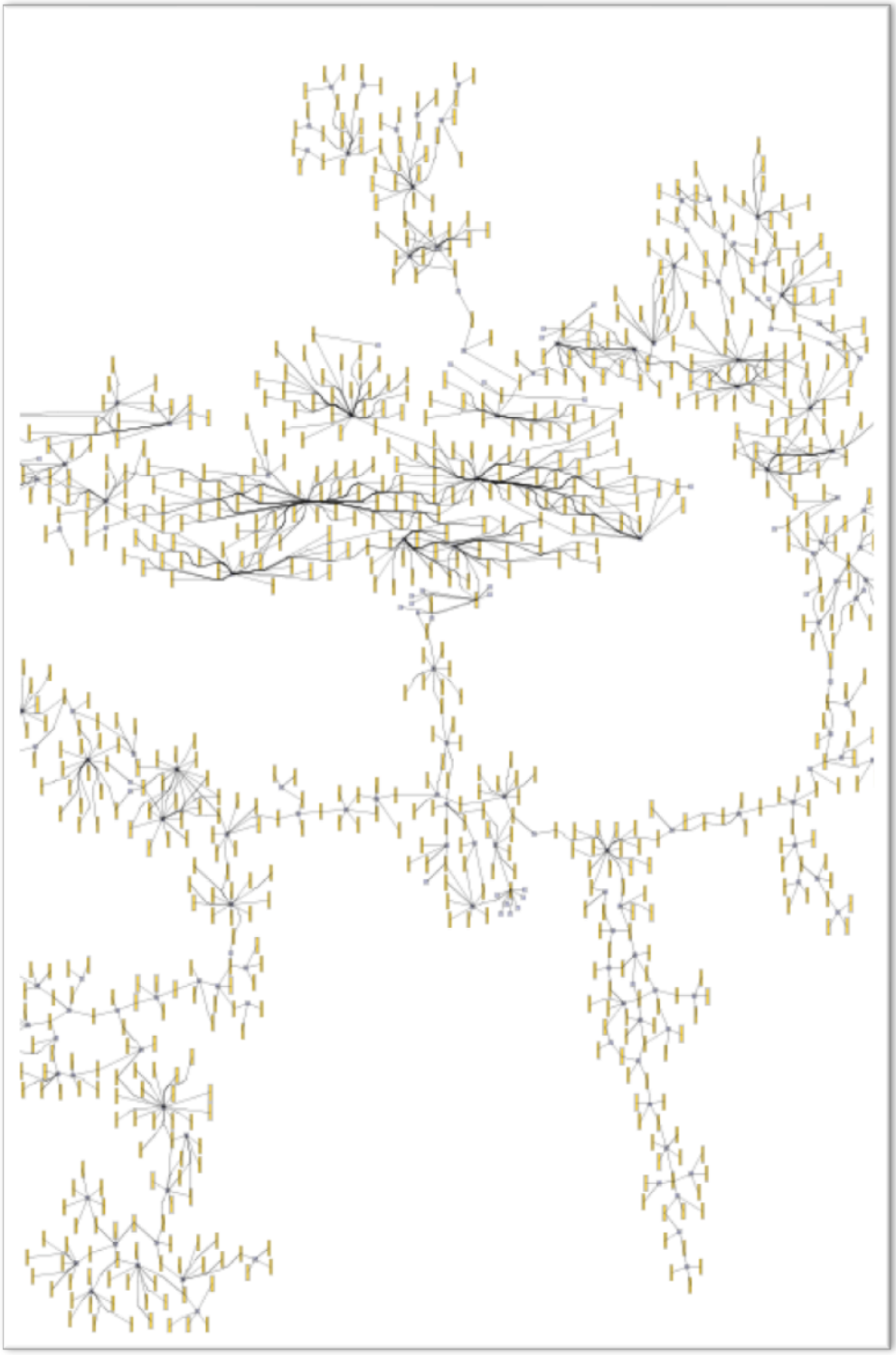

Figure 5.1: Partial View of the SP's Real Network Topology
(diagram rotated 90º)

## 5.3 Configuration Generation Testing

The configuration generation process developed could not be tested in a real deployment scenario due to time limitations and the need to test the application in a set of network equipments from Cisco and Juniper. Additionally, other vendors would be required, as 30% of the configurations available were from Enterasys devices. However, whenever real devices were available for testing, most configurations were tested for correct syntax and were deployed in a controlled environment. This does not ensure that configurations are usable in a real scenario when extracted directly from the Q-Andrew application. Further tests are required to ensure the correct configuration generation in real scenarios. Both the framework and the application were developed with this limitation in mind. For this reason, updates to application logic and/or to the abstract model can be easily deployed.

The vendor's management interfaces used for the configuration generation process show many differences between each vendor's underlying implementation of queuing models, priority mechanisms and marking schemes. We identify the differences between vendors and propose simple mechanisms to contain them. Configurable upper bound delays are defined for delay-sensitive services such as IPTV and VoIP transport. The router forwarding behavior is configured for each vendor using the most suited configuration, even when the same QoS mechanism is not comparable between vendors (e.g., Cisco Priority Queuing and Juniper Forwarding Class Queues in Section 4.6.2). To make sure that configurations are compatible with the required forwarding behavior, active monitoring tests are required, where the forwarding behaviors for a given configuration must be tested and observed. With this data, the model can be enriched enabling the configuration generation process to be fined tuned according to each vendor's forwarding behavior.

# Chapter 6

# Conclusion and Future Work

## 6.1 Conclusion

Network Management is a complex topic, since the lack of standard management interfaces has increased the time required to configure a large network. Vendor diversity introduces interoperability and compatibility issues when a common configuration is deployed. Nowadays, most of this work is executed either manually or using each vendor's proprietary mechanisms.

A Network Management Framework is developed to handle the configuration of devices from multiple vendors. To demonstrate the use of the framework, QoS management methods are proposed that automate the creation of QoS policy for multiple vendors' devices. The Q-Andrew application is created demonstrating the use of the framework and some of the methods proposed for QoS management. The results are shown to the Service Provider used as reference, proving that the methods proposed in this work will become a usable product in the future.

## 6.2 Future Work

### 6.2.1 Framework Extensions

#### 6.2.1.1 Other Management Mechanisms

The generic framework presented can be used in the future to create additional management applications. This model can be extended to handle configurations and topology information from Security Managed Services, as described in Ricardo Oliveira's "Operational Optimization of Security Managed Services in Large WAN/LAN Corporate Networks"

[34]. On the other hand, the proposed automated configuration mechanisms can be used to deploy configurations required in [34].

#### 6.2.1.2 Additional Vendors

In Section 4.6, the configuration generation process is described. The configuration structure is created using object-oriented primitives, allowing the re-use of the methods implemented with other vendors' management interface.

### 6.2.2 Monitoring

The objective of this project was to demonstrate the possibility of creating a re-usable framework to deploy QoS configurations in devices from different vendors. The QoS management application requires a feedback mechanism from active and passive network monitoring. This was considered an assumption for this project (Section 3.1.4) but is a mandatory add-on in the future.

# Bibliography


[1] Cisco Systems. Cisco Public Website. http://www.cisco.com. 1.2.1, 4.7

[2] Juniper Networks. Juniper Networks Public Website. http://www.juniper.net. 1.2.2, 4.7

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. *RFC2475*, December 1998. 2.1, 2.3.1, 2.3.2

[4] J. Postel. Internet Protocol. *RFC791*, 1981. 2.1

[5] P. Almquist. Type of Service in the Internet Protocol Suite. *RFC1349*, July 1992. 2.1

[6] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. *RFC1633*. 2.1

[7] R. Braden, L. Zhang, S. Berson, and S. Herzog aand S. Jamin. Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification. *RFC2205*, September 1997. 2.1

[8] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6). *RFC2460*, December 1998. 2.1

[9] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. *RFC2474*, 1998. 2.1, 2.3.2.3

[10] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. *RFC3031*, January 2001. 2.1

[11] F. Le Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, and J. Heinanen. Multi-Protocol Label Switching (MPLS) Support of Differentiated Services. *RFC3270*, May 2002. 2.1

[12] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, August 1993. 2.2.3, 4.6.4

[13] A. Leon-Garcia and I. Widjaja. *Communication Networks: Fundamentals Concepts and Key Architectures*. McGraw-Hill Science/Engineering/Math, May 2001. 2.2.4

[14] J. Heinanen and R. Guerin. A Single Rate Three Color Marker. *RFC2697*, 1999. 2.2.4

[15] J. Heinanen and R. Guerin. A Two Rate Three Color Marker. *RFC2698*, 1999. 2.2.4

[16] K. Nichols and K. Poduri. An Expedited Forwarding PHB. *RFC2598*, June 1999. 2.3.2.1

[17] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. *RFC2597*, June 1999. 2.3.2.2

[18] J. Evans and C. Filfils. *Deploying IP and MPLS QoS For Multiservice Networks*. Morgan Kaufmann, 2007. 2.3.2.2, 4.5.2.1, 4.5.2.2

[19] L. Liu and M. Tamer. Encyclopedia of Database Systems. Technical report, Springer-Verlag, 2008. 3.4

[20] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001. 3.4.1

[21] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. *RFC1918*, 1996. 3.5.2.1

[22] K. Egevang and P. Francis. The IP Network Address Translator (NAT). *RFC1631*, 1994. 3.5.2.1

[23] The Protégé Ontology Editor and Knowledge Acquisition System. http://protege.stanford.edu/. 3.6

[24] Jena – A Semantic Web Framework for Java. http://jena.sourceforge.net/, 2008. 3.6

[25] Stefan Nilsson and Gunnar Karlsson. Fast IP Routing with LC-Tries. *http://www.ddj.com/184410638/*, August 1998. 4.3

[26] J. Babiarz and K. Chan. Configuration Guidelines for DiffServ Service Classes. *RFC4594*, 2006. 4.5.1

[27] ITU-T. International telephone connections and circuits – General Recommendations on the transmission quality for an entire international telephone connection. Technical report, TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, May 2003. 4.5.2.1

[28] M. Al-Khatib and M. Alam. IPTV Multimedia Networks: Concepts, Development, and Design. Technical report, International Engineering Consortium, 2007. 4.5.2.2

[29] ISO/IEC. Coding of audio-visual objects - Part 12: ISO base media file format. Technical report, ISO/IEC, 2008. 4.5.2.2

[30] T. Li, B. Cole, P. Morton, and D. Li. Cisco Hot Standby Router Protocol (HSRP). *RFC2281*, March 1998. 4.6.4

[31] R. Hinden. Virtual Router Redundancy Protocol (VRRP). *RFC3768*, April 2004. 4.6.4

[32] ITU-T. Infrastructure of audiovisual services – Systems and terminal equipment for audiovisual services. Technical report, TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, 2006. 4.10

[33] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Protocol Architecture. *RFC4251*, 2006. 4.10

[34] R. Oliveira. The Prometheus System - Operational Optimization of Security Managed Services in Large WAN/LAN Corporate Networks. *CMU MSIT-IS Thesis*, 2008. 6.2.1.1