ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE AVEC CONCENTRATION EN RÉSEAUX DE TÉLÉCOMMUNICATIONS
M. Ing.

PAR
MARC-ANDRÉ BRETON

DÉVELOPPEMENT D'UN SYSTÈME DE SURVEILLANCE DES MÉCANISMES DE QUALITÉ DE SERVICE DANS LE CONTEXTE DES RÉSEAUX DE PROCHAINE GÉNÉRATION

MONTRÉAL, LE 06 JUILLET 2006

CE MÉMOIRE A ÉTÉ ÉVALUÉ
PAR UN JURY COMPOSÉ DE :

Mme Maria Bennani, directrice de mémoire
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Michel Lavoie, président du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Jean-Marc Robert, membre du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC
LE 26 JUILLET 2006
À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

DÉVELOPPEMENT D'UN SYSTÈME DE SURVEILLANCE DES MÉCANISMES DE QUALITÉ DE SERVICE DANS LE CONTEXTE DES RÉSEAUX DE PROCHAINE GÉNÉRATION

Marc-André Breton

SOMMAIRE

Afin de faciliter la configuration et la surveillance des mécanismes de qualité de service mis en place dans un réseau, un outil approprié doit être mis à la disposition des administrateurs réseau. Cet outil doit permettre une visualisation des configurations et une visualisation de statistiques relatives à la qualité de service. Un tel outil permet, par conséquent, de valider l'homogénéité des configurations à travers l'ensemble du réseau de l'administrateur en plus d'identifier les sources de dégradation de la qualité de service.

En plus de définir la place que peut occuper cet outil dans le contexte des réseaux de prochaine génération (NGN), ce document présente le développement d'une architecture de base permettant la visualisation des mécanismes de qualité de service dans un réseau hétérogène. Il décrit, entre autre, les diverses composantes de l'architecture ainsi que le développement de chacune d'elles.

Ce développement, réalisé au *Laboratoire de gestion de réseaux informatiques et de télécommunications* (LAGRIT), a été validé par une série d'essais réalisés dans les laboratoires de Bell Canada. Ce projet est donc considéré comme un projet industriel puisqu'il a abouti à un produit pouvant être utilisé par un administrateur de réseau.

Finalement, certaines suggestions ont été apportées afin de permettre, dans un premier temps, d'améliorer les performances du système et dans un deuxième temps, de développer d'autres fonctionnalités pouvant être implémentées dans un contexte de recherche future.

DEVELOPMENT OF A MONITORING SYSTEM FOR QUALITY OF SERVICE MECHANISMS IN THE CONTEXT OF THE NEXT GENERATION NETWORKS

Marc-Andre Breton

ABSTRACT

To facilitate the configuration and the monitoring of the quality of service mechanisms implemented in a network, an appropriate tool must be placed at the disposal of the network administrators. This tool must provide the visualization of both configurations and statistics of the quality of service mechanisms. It could then help the network administrator to validate the homogeneity of the configurations throughout the entire network. Moreover, this tool could help the administrator to identify the source of the quality of service degradation.

In a first step, this document defines the place that this tool can occupy in the next generation network (NGN) context. Next, it develops a basic architecture which can provide the visualization of the quality of service mechanisms in a heterogeneous network. In fact, it gives a general description of each component of the architecture and it describes the development of each one.

This development has been realized at the *Laboratoire de gestion de réseaux informatiques et de télécommunications* (LAGRIT) and it has been validated by a series of tests realised at Bell Canada's laboratory. Thus, this project is considered as an industrial project because it leads to a product that can be used by a network administrator.

Finally, this document finishes by presenting some recommendations. First, a new architecture is proposed to improve the performances of the entire system. Next, a few suggestions are made to develop some new functionalities that can be implemented in the context of future researches.

REMERCIEMENTS

Je tiens d'abord à exprimer toute ma reconnaissance à Mme Maria Bennani, professeure à l'École de technologie supérieure de Montréal. D'abord pour m'avoir permis de réaliser ce projet et pour m'avoir dirigé tout au long de celui-ci. Par ailleurs, pour m'avoir soutenu matériellement et financièrement sans quoi, l'aboutissement de ce projet n'aurait pu être possible.

Pour m'avoir permis d'accéder aux équipements du laboratoire de Bell Canada et pour m'avoir permis d'acquérir une expérience très pratique des mécanismes de qualité de service, j'aimerais particulièrement remercier M. Guy Côté de Bell Canada.

Par la suite, j'aimerais remercier certaines personnes qui m'ont aidé lors du développement du projet. Entre autre M. Nicolas Manen, ancien stagiaire du LAGRIT et maintenant employé de Bell Canada, qui m'a fourni une aide précieuse quant à la manipulation des divers équipements de test. Par la suite, M. Olivier Truong, étudiant de maîtrise qui a développé l'aspect MPLS du système, pour ses conseils judicieux lors du développement de ce dernier. Mlle Virginie Convert, qui m'a permis de corriger certains problèmes dans le système développé et ce, afin d'en faire un meilleur produit. Finalement M. Abdelghani Benharref, pour ses conseils sur la rédaction de ce mémoire ainsi que pour ses corrections rapides.

Pour terminer, j'aimerais remercier le LAGRIT pour m'avoir fourni l'espace et le matériel nécessaire à tout le travail réalisé dans le cadre de ma maîtrise. De plus, merci à tous les membres du LAGRIT pour leur soutien lorsque requis.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS ET SIGLES

AF	Assured Forwarding
API	Application Programming Interface
CBQ	Class-Based Queueing
CE	Customer Edge device
CLI	Command Line Interface
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
Codec	Coder Decoder
COPS	Common Open Policy Service protocol
COPS-PR	COPS Usage for Policy Provisioning
CoS	Class of Service
DiffServ	Differentiated Services
DSCP	Differentiated Services Code Point
EF	Expedited Forwarding
FAQ	Frequently Asked Questions
FQ	Fair Queueing
FTP	File Transfer Protocol
IETF	Internet Engineering Task Force
IntServ	Integrated Services
IP	Internet Protocol
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
LAGRIT	Laboratoire de gestion de réseaux informatiques et de télécommunications
LAN	Local Area Network
LLQ	Low Latency Queue
LSR	Label Switched Router

MGCP	Media Gateway Control Protocol
MIB	Management Information Base
MPD	Mark Probability Denominator
MPLS	Multi Protocol Label Switching
MSF	Multiservice Switching Forum
NAT	Network Address Translation
NGN	Next Generation Network
NMS	Network Management System
NRCP	Network Resource Control Protocol
OSI	Open System Interconnection
P	Provider device
PE	Provider Edge device
PHB	Per Hop Behavior
POTS	Plain Old Telephone Service
PQ	Priority Queueing
PSTN	Public Switched Telephone Network
QdS	Qualité de Service
QMA	QoS and MPLS Assistant
QoS	Quality of Service
QPM	QoS Policy Manager
RED	Random Early Detection
RR	Round Robin
RSVP	Resource Reservation Protocol
RTP	Real Time Protocol
SBM	Subnetwork Bandwidth Manager
SIP	Session Initiation Protocol
SIP-T	SIP for Telephones
SNMP	Simple Network Management Protocol
SS#7	Signalling System Number 7

SSH	Secure Shell
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
TMN	Telecommunication Management Network
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VLAN	Virtual LAN
VoIP	Voice over IP
VPN	Virtual Private Network
WFQ	Weighted Fair Queueing
WRED	Weighted RED

INTRODUCTION

Ce mémoire présente le système *QoS and MPLS Assistant* (QMA) développé au *Laboratoire de gestion de réseaux informatiques et de télécommunications* (LAGRIT) et testé dans les laboratoires de Bell Canada.

Le premier chapitre définit la problématique du projet. Cette problématique fait ressortir les besoins qu'ont les opérateurs de réseau pour pouvoir effectuer une bonne gestion des divers mécanismes de qualité de service.

Le second chapitre présente l'architecture des réseaux de prochaine génération, dans laquelle s'introduit le système développé dans le cadre de ce mémoire. Par ailleurs, comme la qualité de service est un élément indissociable des réseaux de prochaine génération et comme le système, développé dans ce mémoire, a pour principal but la surveillance des mécanismes de qualité de service, ces derniers ont été décrits en détail dans ce chapitre. Par la suite, les divers protocoles standardisés pour effectuer la surveillance de réseau ont été décrits et comparés. Finalement, un survol du marché a été réalisé, en ce qui a trait aux systèmes pouvant surveiller les mécanismes de qualité de service dans un réseau, et les caractéristiques de chacun ont été énumérées. De plus, pour chaque système présenté, les lacunes ont été décrites afin de faire ressortir la nécessité d'avoir un système tel que celui développé dans ce mémoire.

Le troisième chapitre situe le système, développé dans ce mémoire, dans le contexte des réseaux de prochaine génération qui ont préalablement été présentés dans le chapitre 2. Par la suite les spécifications fonctionnelles et non fonctionnelles du système sont énumérées afin de définir les exigences que doit respecter le système une fois son développement complété.

Le quatrième chapitre décrit la conception du système en définissant d'abord l'architecture générale utilisée. Par la suite, chaque composante de l'architecture est décrite en détail. Finalement ce chapitre présente l'interaction entre les diverses composantes de l'architecture générale par le biais de diagramme de classes et de séquences.

Le cinquième chapitre présente certains essais de validations réalisés dans le but de prouver le bon fonctionnement des principales fonctionnalités du système. Seules les fonctionnalités relatives à la qualité de service on été validées dans ce chapitre.

Pour terminer, ce mémoire conclut en effectuant un survol de ce qui a été vu, puis des recommandations sont énumérées concernant l'évolution que pourrait prendre le système dans le futur. Cette évolution touche principalement l'amélioration des performances et des fonctionnalités du système. Finalement, l'ANNEXE 4 inclut un article soumis à la conférence ACM CoNEXT 2006 dont certains résultats, comme le débit de sortie de chaque classe par exemple, ont été obtenus à l'aide du système QMA.

CHAPITRE 1

PROBLÉMATIQUE

De nos jours, l'intégration de plusieurs services au sein d'une même infrastructure réseau est une pratique de plus en plus courante. Cependant, une telle intégration peut engendrer plusieurs problèmes au niveau de la qualité de service[1] (QdS) des services offerts. En effet, en passant d'une infrastructure dédiée pour chaque service à une autre, de type « meilleur effort », partagée par tous les services, des problèmes concernant la QdS peuvent survenir. Ces problèmes de qualité de service sont généralement engendrés, par exemple, par des pertes de paquets, un délai[2] ou une variation de délai trop élevée. Parfois la bande passante est vue comme un autre élément pouvant détériorer la qualité des communications. Cependant, lorsque la bande passante est insuffisante pour un service donné, la détérioration de la qualité de service sera perçue par une augmentation des délais, variation de délai et pertes de paquets. Il s'avère donc très important de mettre en place des mécanismes de QdS lorsqu'une infrastructure de type « meilleur effort » intègre plusieurs services ayant des critères de qualité spécifiques.

Il existe principalement 6 mécanismes de QdS soit la classification, le marquage, le lissage, le *policing*, l'ordonnancement et l'évitement de congestion (*Random Early Detection* - RED). Ces mécanismes sont brièvement décrits ci-dessous. Pour une description plus détaillée, veuillez vous référer à la section 2.2.

---

[1] La « qualité de service » signifie « aptitude d'un service à répondre adéquatement à des exigences, exprimées ou implicites, qui visent à satisfaire ses usagers ».
Cette définition est tirée de http://w3.granddictionnaire.com/.

[2] On entend par *délai* ou par *latence* le temps d'acheminement d'un paquet entre deux extrémités du réseau de télécommunications.

a. La classification consiste à diviser les différents types de trafic, circulant dans le réseau, et à les regrouper par classes. Ainsi, chaque classe recevra un traitement spécifique.
b. Le marquage consiste à identifier les différents types de trafic, qui circulent dans le réseau, en modifiant un champ des entêtes de certains protocoles afin de leur donner une valeur appropriée. De façon générale, le trafic d'une classe obtient une marque spécifique.
c. Le lissage est un mécanisme utilisé pour limiter le trafic en sortie à un certain débit. En fait, lorsqu'il y a beaucoup d'informations à transmettre, de sorte que le débit dépasse le débit spécifié, certains paquets sont conservés en mémoire pour être transmis ultérieurement.
d. Le *policing* est un autre mécanisme utilisé pour surveiller et limiter le trafic en entrée à un certain débit. Celui-ci est, par contre, plus radical puisqu'il rejette systématiquement les paquets en trop.
e. L'ordonnancement est un mécanisme utilisé pour gérer l'ordre de sortie de l'information sur l'interface. Il gère les priorités entre les différentes classes et garanti un service minimal à chacune d'elles.
f. L'évitement de congestion (RED) est un mécanisme qui utilise le mécanisme de contrôle de congestion du protocole *Transmission Control Protocol* (TCP) [1]. En fait, le mécanisme d'évitement de congestion anticipe celle-ci en jetant aléatoirement des paquets TCP afin de diminuer le débit des connexions.

Dépendamment de la nature des mécanismes de QdS, ceux-ci devront être implémentés à des endroits stratégiques dans le réseau. De plus, comme la dégradation de la QdS sera principalement ressentie par les entités communicantes, les mécanismes de QdS mis en place dans le réseau devront assurer la garantie du service entre les deux extrémités communicantes, c'est-à-dire de bout en bout. Ainsi, si la QdS est mal gérée dans une portion du réseau, une dégradation du service peut se faire ressentir. L'implémentation de la QdS dans un réseau peut donc être comparée à la fabrication d'une chaîne aussi

forte que son maillon le plus faible. C'est pourquoi il est très important de ne négliger aucun maillon de la chaîne. Typiquement, les configurations de QdS doivent s'exécuter routeur par routeur via un *Command Line Interface* (CLI) ou encore à l'aide d'un logiciel propriétaire à l'équipementier. Cependant, il peut s'avérer difficile, pour un administrateur réseau, de conserver une vision globale de la chaîne de QdS lorsque celui-ci doit se concentrer sur la configuration d'un seul équipement pour ensuite passer au suivant. Dans le cas d'un fournisseur de service, le réseau peut s'avérer être très grand et les configurations au sein d'un seul équipement peuvent être lourdes et complexes. Plus le réseau est large et complexe, plus il est facile de négliger un des maillons de la chaîne de QdS et ainsi compromettre la QdS des services offerts. De plus, plus une configuration est lourde et complexe, plus il est difficile, pour l'administrateur du réseau, d'obtenir l'information dont il a besoin. C'est pourquoi il est recommandé à l'administrateur d'utiliser un outil logiciel lui permettant d'obtenir uniquement l'information qu'il nécessite et lui permettant également d'avoir une vision globale des configurations de QdS dans son réseau. Un tel outil facilitera sans doute l'obtention d'une certaine homogénéité dans les configurations de QdS de bout en bout.

Un autre aspect très important et non négligeable à considérer par un administrateur de réseau est l'identification des sources de la dégradation de la QdS. En effet, il s'avère beaucoup plus difficile pour un administrateur d'identifier la source de la dégradation s'il doit entrer des commandes dans le CLI. En utilisant une telle méthode, il faut que le problème persiste au moment où l'administrateur exécute ses commandes. Dans le cas d'une détérioration de service qui survient de façon sporadique, l'utilisation d'une telle méthode n'est absolument pas adaptée et pratique. C'est pourquoi l'usage d'un outil permettant la surveillance des mécanismes de QdS en continu et permettant un affichage temporel est à considérer par l'administrateur. De plus, un tel outil permet généralement la détection d'anomalies pour en informer l'administrateur lorsque certains événements surviennent.

Finalement, en considérant l'hétérogénéité du réseau d'un fournisseur de service, l'usage d'outils de surveillance propriétaires aux équipementiers, présents dans le réseau, n'est pas à considérer. Dans une telle situation, l'administrateur du réseau aura à utiliser 2, 3 ou même plus d'outils de surveillance simultanément afin de conserver une vision globale du réseau. Une solution plus conviviale consiste à utiliser un outil générique de surveillance intégrant les différents équipementiers présents dans le réseau. Une telle solution permettrait d'avoir une vision globale des configurations de QdS établies dans le réseau ainsi que l'utilisation des ressources.

Le présent mémoire présente donc une proposition d'une architecture de base d'un système qui permet la gestion et la surveillance des mécanismes de QdS dans un réseau hétérogène. Par la suite, le développement de cette architecture sera présenté puis validé. Plus précisément, l'aspect « surveillance » sera développé puis, des propositions seront faites quant à l'évolution que pourra prendre le système. Ces propositions seront basées principalement sur les NGN présentés dans le forum multiservice (http://www.msforum.org). Le système développé dans le cadre de ce mémoire a été nommé *QoS and MPLS Assistant* (QMA). Ce nom est utilisé, tout au long de ce mémoire, pour référer à ce système. L'acronyme *MPLS* a été inséré puisque QMA est un système utilisé pour surveiller les mécanismes de qualité de service en plus de tous les aspects liés à MPLS et l'ingénierie de trafic. Cette dernière partie a été réalisée dans QMA par Olivier Truong, un étudiant de maîtrise du LAGRIT, qui en a documenté l'implémentation dans son propre mémoire [2]. Par conséquent le présent mémoire fait référence uniquement à l'aspect de qualité de service puisque l'aspect MPLS a été développé dans un autre document.

CHAPITRE 2

ÉTAT DE L'ART

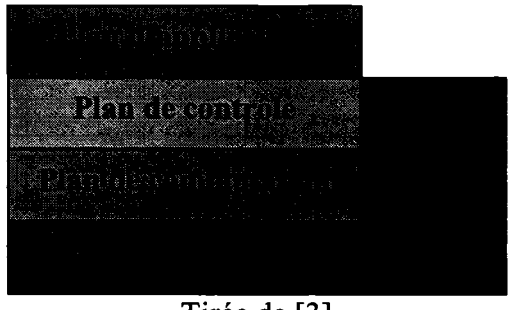
Afin de résoudre la problématique exprimée dans le CHAPITRE 1, qui consiste à faciliter la gestion et la surveillance des mécanismes de QdS dans un réseau hétérogène par le biais d'un outil logiciel, le présent chapitre présente le contexte dans lequel l'outil en question doit être inséré. Les NGN sont abordés en présentant l'approche développée dans le forum multiservices MSF. Par la suite, comme la QdS est un élément indispensable dans les NGN, une introduction technique des différents mécanismes de QdS est fournie. Par ailleurs, étant donné que le projet présenté dans ce mémoire se concentre principalement sur l'aspect « surveillance » des mécanismes de QdS, une section de ce chapitre est consacrée aux différents standards développés pour effectuer de la surveillance de réseaux. Finalement un survol des différents systèmes industriels de surveillance disponibles sur le marché est réalisé afin d'en faire ressortir les lacunes et, du fait même, de faire ressortir la nécessité de développer un système comme QMA.

2.1 Les réseaux de prochaine génération (NGN)

Le forum multiservice (voir http://www.msforum.org) a été fondé en 1998 et est composé principalement de fournisseurs de service et d'équipementiers du domaine des télécommunications. Son principal rôle consiste à développer une architecture ouverte pour les systèmes de commutation multiservice. Il est à noter que le développement réalisé jusqu'à maintenant, a été basé sur une infrastructure réseau pouvant offrir des services multimédia tels la voix ou vidéo sur IP.

Deux architectures ont été développées. L'architecture fonctionnelle présentée dans [3] puis l'architecture physique présentée dans [4]. Il est à noter que l'architecture fonctionnelle a été reconsidérée dans [5] et que les auteurs stipulent dans l'introduction :

« L'architecture de référence s'écarte des entités purement fonctionnelles; tous les éléments de l'architecture peuvent être réalisés en tant que composants physiques séparés; la plupart, sinon tous, sont présentement disponibles en tant que produits commerciaux ». Bien que les auteurs aient reconsidéré l'architecture fonctionnelle, le concept était tout de même intéressant. Cette architecture fonctionnelle est donc présentée dans la Figure 1 et dans le Tableau I. Dans ce dernier, les fonctions qui concernent le système considéré par ce mémoire sont mises en caractères gras.



Tirée de [3]

Figure 1 Plan de l'architecture fonctionnelle des NGN

Tableau I

Plan et fonctions de l'architecture fonctionnelle des NGN

PLAN	FONCTIONS
Plan d'adaptation	a. Formate les données de façon appropriée pour la transmission. b. Peut implémenter certains mécanismes de QdS tel la classification, l'ordonnancement, le *policing* et le lissage.
Plan de commutation	a. Fournit l'interconnexion de base entre ports logiques. b. Transmet les données utilisateur en utilisant des étiquettes. c. Supporte une multiplicité d'éléments de commutation dans le domaine d'un contrôleur.

Tableau I (suite)

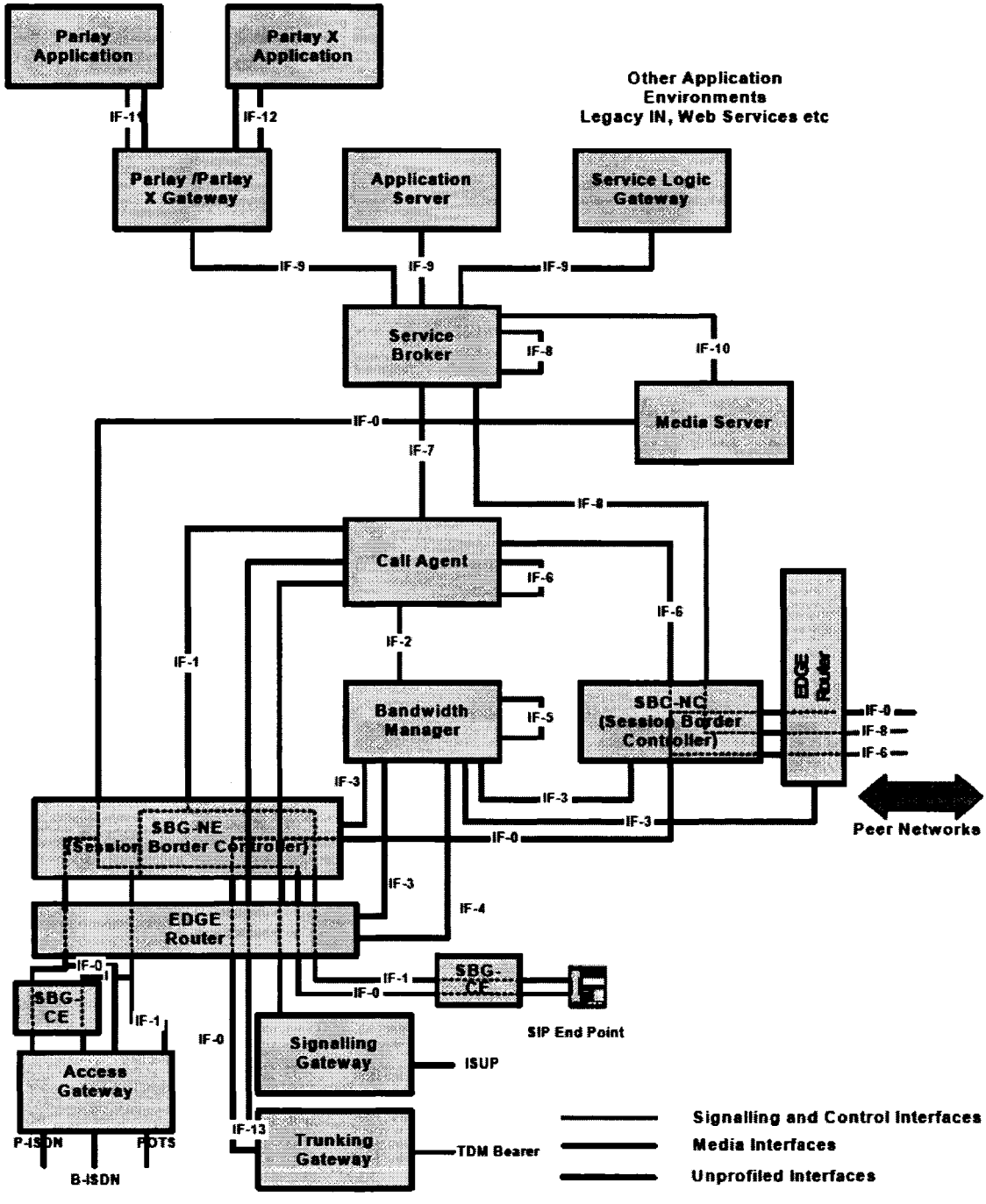
PLAN	FONCTIONS
Plan de commutation (suite)	a. Réplique l'information pour les connexions point à multipoints. b. Fournit une interface au plan d'adaptation. c. Partitionne et partage les ressources d'un commutateur.
Plan de contrôle	a. Achemine le trafic entre les plans d'application, de commutation et d'adaptation et il alloue les ressources du plan de commutation et d'adaptation. b. Contrôle les associations d'étiquettes entre les ports et interfaces via le plan de commutation. c. Commande l'établissement, la modification ou le relâchement des connexions. d. **Assigne les paramètres de QdS** pour chaque connexion ou flot. e. Contrôle les fonctions du plan d'adaptation. f. Fournit une interface au plan d'application. g. Fournit une variété de services accédés par des protocoles de signalisation pour la voix, la vidéo et les données. h. **Effectue le contrôle d'admission et de l'ingénierie de trafic.** i. **Fournit des statistiques et des alarmes.** j. Reçoit l'information de signalisation et l'achemine aux autres entités du plan de contrôle s'il y a lieu. k. Négocie les connexions et les paramètres d'adaptation tel le débit et le type de Codec.
Plan d'application	a. Fournit des services de messagerie tel le courriel et la boîte vocale. b. Fournit des services locaux de signalisation tels l'appel en attente, le transfert d'appel, etc. c. Fournit des services de traitement d'information tel le traitement des cartes de crédit et autres.

Tableau I (suite)

PLAN	FONCTIONS
Plan d'application (suite)	d. Fournit des services d'adressage IP et de noms de domaine tel DHCP, DNS et autres.
Plan de gestion	a. Effectue la gestion des fautes. b. **Effectue la gestion des configurations.** c. Effectue la gestion des comptes. d. **Effectue la gestion des performances.** e. Effectue la gestion de la sécurité.

Tiré de [3]

La Figure 2 présente la seconde version de l'architecture de référence des réseaux de nouvelle génération selon le forum multiservice. Les interfaces et les composantes de cette architecture sont respectivement présentées dans le Tableau II et le Tableau III.



Tirée de [5]

Figure 2 Architecture de référence des NGN selon MSF

Tableau II

Description des interfaces de l'architecture de référence des NGN

Interface	Description	Protocoles supportés
IF-0	Interface média	RTP
IF-1	*User Agent* ou *Media Gateway* vers *Call Agent*	SIP, MGCP, H.248
IF-2	*Call Agent* vers *Bandwidth Manager*	SIP, NRCP
IF-3	*Bandwidth Manager* vers *Edge Router* ou *Session Border Controller*	H.248, COPS-PR
IF-4	*Bandwidth Manager* vers *Core/Edge Router*	Sous étude
IF-5	*Bandwidth Manager* vers *Bandwidth Manager*	NRCP
IF-6	*Call Agent* vers *Call Agent*	SIP, SIP-T
IF-7	*Call Agent* vers *Service Broker*	SIP
IF-8	*Service Broker* vers *Service Broker*	SIP
IF-9	*Service Broker* vers *Application Server*	SIP
IF-10	*Service Broker* vers *Media Server*	SIP
IF-11	*Parlay Gateway* vers *Parlay Application*	PARLAY API
IF-12	*ParlayX Gateway* vers *ParlayX Application*	PARLAYX API
IF-13	*Trunking Gateway* vers *Call Agent*	MGCP, H.248

Tiré de [5]

Il est intéressant de constater, dans le Tableau II, que toutes les interfaces supportent des protocoles déjà existant excepté l'interface entre le *Bandwidth Manager* et les équipements réseau. Cette problématique peut être temporairement résolue en établissant une connexion sécurisée *Secure Shell* (SSH) [6] entre le *Bandwidth Manager* et l'équipement réseau en question. Une fois la connexion établie, le *Bandwidth Manager*

peut envoyer une série de commandes aux équipements en question. Il va de soi qu'il est absolument nécessaire que les équipements supportent SSH.

Tableau III

Descriptions des composantes de l'architecture de référence des NGN

Composantes	Descriptions
Signaling Gateway	Il sert de médiateur entre la signalisation SS#7 du PSTN et le *Call Agent*.
Trunking Gateway	Fournit le transcodage du média entre le réseau TDM externe et le réseau *Internet Protocol* (IP) du fournisseur de service. Il est sous le contrôle du *Call Agent* via MGCP et H.248.
Access Gateway	Supporte les téléphones standards POTS afin qu'ils puissent communiquer dans un réseau IP. Il se situe habituellement chez le client, soit à l'extérieur du réseau du fournisseur de service.
Edge Router	Il achemine le trafic IP dans la dorsale du fournisseur de service. Sous la charge du *Bandwidth Manager*, il est responsable d'effectuer le contrôle d'admission et le *policing*.
Session Border Gateway – Network Edge (SBG-NE)	Il fournit certaines fonctions de bordure telle la translation d'adresse (NAT), la détection et la prévention d'intrusion et il gère également l'association entre les flots RTP et la signalisation.
Session Border Gateway – Network Core (SBG-NC)	Il agit d'une façon similaire au SBG-NE mais il est déployé plutôt à l'interconnexion entre deux fournisseurs de service. Il effectue entre autre du NAT afin de pouvoir mieux cacher la topologie réseau d'un fournisseur.

Tableau III (suite)

Composantes	Descriptions
Session Border Gateway – Customer Edge (SBG-CE)	Il agit d'une façon similaire au SBG-NE mais il est situé chez le client afin de permettre une complémentarité.
Call Agent	Il gère les sessions (établissement, relâchement, etc…) et génère des rapports pour la facturation pour toutes les sessions sous son contrôle. Il demande les services du *Service Broker* et conserve un registre des abonnés de façon statique ou dynamique. Dans le cas où le registre est dynamique, le *Call Agent* doit pouvoir découvrir les autres *Call Agent* où un de ses abonnés s'est connecté.
Bandwidth Manager	**Il gère la QdS dans le réseau.** Il s'occupe d'allouer et de retirer la bande passante à certains flots en plus de gérer l'accès à cette bande passante. Il communique avec les *Edge Router* afin de leur imposer des politiques de service.
Media Server	Il fournit des fonctions aux autres composants réseaux. Ces fonctions peuvent être entre autre de jouer des annonces publicitaires, de détecter et de générer une tonalité, de traiter les fax, d'effectuer du mixage audio, pour les appels conférence par exemple, ainsi que d'autres fonctions.
Service Broker	Il gère l'interaction entre différentes applications, de technologies différentes, au sein d'une simple session.
Application Server	Il fournit l'exécution d'un ou plusieurs services et est orchestré par le *Service Broker*.

Tableau III (suite)

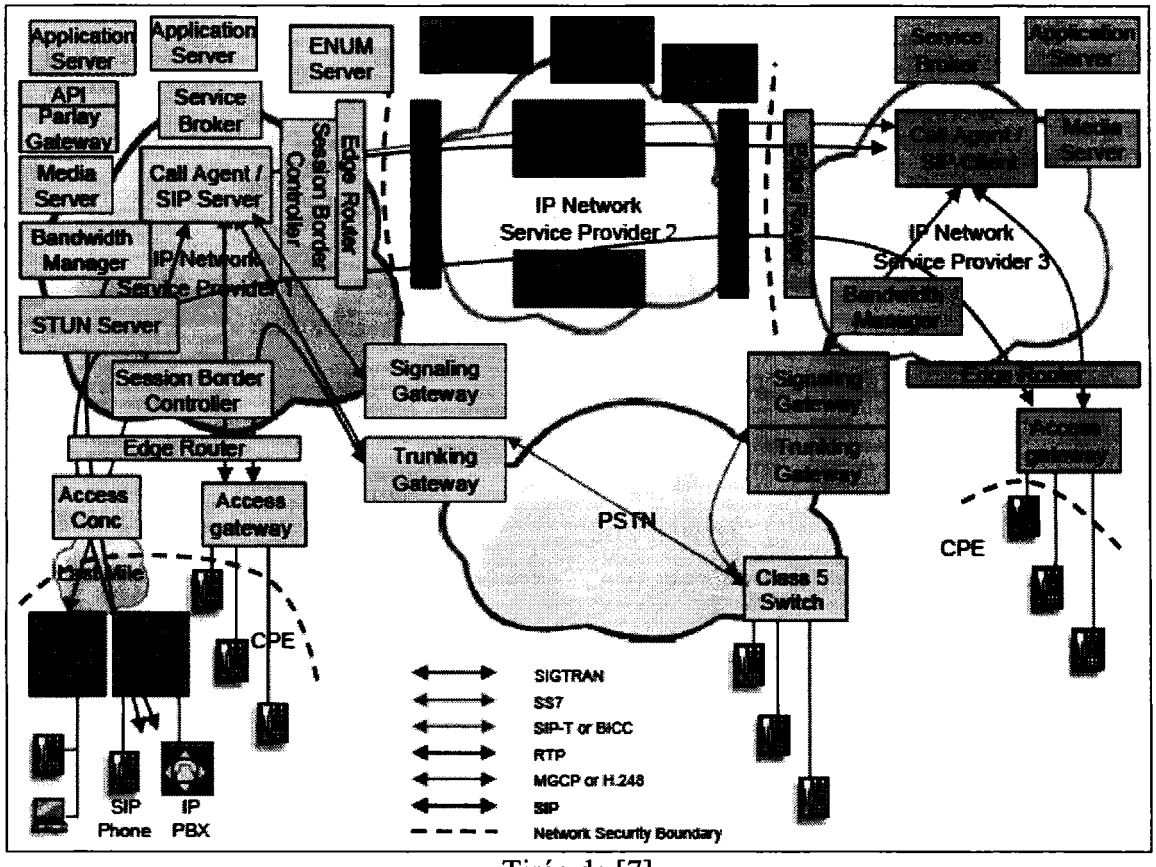
Composantes	Descriptions
Service Logic Gateway	Il permet à une application non-SIP de demander accès aux ressources réseau par le biais du *Service Broker.*
Parlay/ParlayX Gateway[1]	Il s'agit d'un type de *Service Logic Gateway* qui exporte les API Parlay/ParlayX vers les applications.
Parlay/ParlayX Application	Fournit la logique et l'exécution des services et est engagé via le *Parlay/ParlayX Gateway* sous l'orchestration du *Service Broker.*

Tiré de [5]

La Figure 3 présente un scénario de test qui permet de mieux situer les diverses composantes présentées dans la Figure 2. On peut entre autre voir qu'il y a trois fournisseurs de services et que deux d'entre eux ont une connexion au réseau *Public Switched Telephone Network* (PSTN) via un *Signaling* et un *Trunking Gateway.* Par la suite, on remarque que tous les domaines IP distincts sont séparés par un routeur de bordure (ou *Edge Router*). Par conséquent, chaque domaine est en mesure d'effectuer le contrôle d'admission et d'appliquer les politiques de QdS qu'il a définies. Notez que le fournisseur de service 1 fournit un accès IP ainsi qu'un accès aux téléphones *Plain Old Telephone Service* (POTS). Le fournisseur de service 2 est plutôt utilisé comme transporteur d'information entre deux fournisseurs. Quant au fournisseur de service 3, il fournit un accès local à des téléphones POTS uniquement, mais permet l'usage de la signalisation *Session Initiation Protocol* SIP jusqu'au serveur SIP. Les trois fournisseurs de service ont un *Bandwidth Manager* au sein de leur réseau afin de pouvoir gérer la QdS selon les requêtes obtenues. Il agira principalement sur les routeurs de bordure afin de permettre l'admission au réseau des nouvelles connexions multi-médias.

---

[1] Brièvement, il s'agit d'une « API, indépendante aux technologies, qui permet le développement d'applications qui opèrent dans un réseau multiservice. Parlay fournit une interface sécuritaire, mesurable et facturable et a été largement déployé dans les réseaux de télécommunications » (http://www.parlay.org/en/index.asp).



Tirée de [7]

Figure 3 Réseau de VoIP de prochaine génération

Étant donné que la qualité de service est un élément indissociable des NGN, la section qui suit présente une introduction technique aux différents mécanismes de QdS.

2.2 Introduction technique aux mécanismes de QdS

Lorsqu'on parle de QdS, il est essentiel de parler d'abord de l'usager puisque c'est lui qui subit les effets de la bonne ou de la mauvaise qualité du service. Tel que spécifié précédemment dans le CHAPITRE 1, les principaux critères de QdS sont le délai, la variation de délai et la perte de paquets. Dépendamment de l'application utilisée par l'usager, les critères de QdS seront différents. En effet, pour des applications temps réel

comme la voix ou la vidéo, les critères sont beaucoup plus stricts que pour une application de transfert de fichiers comme le *File Transfer Protocol* (FTP) ou la navigation sur le Web par exemple. Ceci vient du fait que certaines applications ont un profil de trafic plus élastique. C'est d'ailleurs le cas avec la majorité des applications qui utilisent TCP comme protocole de transport. Certaines autres applications ont un profil de trafic plutôt constant. C'est d'ailleurs le cas de la voix sur IP (VoIP).

Pour certaines applications, il est plutôt difficile de définir, de façon précise, des paramètres de QdS adéquats. Dans le cas de la VoIP, la perception de la QdS est subjective et varie selon l'interlocuteur. Dans [8], les auteurs se sont basés sur la recommandation P.85 de l'*Internationnal Telecommunication Union, Telecommunication Standardization Sector* (ITU-T) afin de déterminer, de façon plus objective, les paramètres de QdS pouvant représenter, au mieux, l'appréciation subjective d'un groupe test d'interlocuteurs. De façon générale, on peut considérer qu'une communication de voix sur IP doit subir moins de 150 msec de délais de bout-en-bout, une variation de délai maximale de 20 msec ainsi qu'un pourcentage de pertes moyen de 0.25%.

Afin de répondre aux critères de QdS imposés par les usagers, certains mécanismes doivent être mis en place dans le réseau où chaque nœud doit être considéré. Cette section présente donc la perspective des nœuds en décrivant les services et mécanismes de QdS pouvant être adoptés.

2.2.1 Les services IntServ et DiffServ

Cette section présente les deux types de services pouvant être déployés dans un réseau IP afin d'offrir une garantie de services aux différents flots qui y circulent. Dans un premier temps, *Integrated Services* (IntServ) sera présenté puis *Differentiated Services* (DiffServ).

2.2.1.1 IntServ

Le principe fondamental de ce type de service consiste à réserver une bande passante pour chaque flot individuel qui requiert une garantie de service [9]. Ainsi, tout le long du chemin emprunté par le flot, une partie des ressources de chaque équipement est allouée à ce flot. Il va s'en dire qu'un mécanisme de contrôle d'admission est absolument nécessaire afin que les ressources réservées pour un flot soient utilisées uniquement par celui-ci. De plus, la réservation de la bande passante nécessite inévitablement l'usage d'un protocole de réservation de ressources tel que *Resource Reservation Protocol* (RSVP) [10]. Un équipement réseau qui implémente RSVP doit conserver en mémoire l'état de chaque réservation et transmettre périodiquement des messages de rafraîchissement afin de maintenir ces états actifs. Notez que RSVP est conçu pour fonctionner uniquement dans des routeurs mais qu'un autre protocole nommé *Subnetwork Bandwidth Manager* (SBM), proposé dans [11], modélisé dans [12] et amélioré dans [13], a été développé afin de permettre une extension de RSVP dans les réseaux locaux (LAN). Il est intéressant de noter qu'en utilisant un tel type de service, le réseau ressemble de plus en plus à un réseau à commutation de circuits puisque le service est garanti pour un flot donné et que ce dernier a une portion des ressources réservées à son usage uniquement.

À première vue, ce type de service semble adapté et efficace mais il en est tout autrement. En fait, il nécessite une réservation de ressources par flots ce qui n'est pas souhaitable pour un déploiement à grande échelle. En effet, pour un équipement au cœur du réseau par lequel peut transiter des milliers de conversations téléphoniques par exemple, la réservation de ressource pour chacune d'elles ferait chuter les performances de l'équipement. En plus de transmettre ces conversations, l'équipement devrait alors maintenir les milliers d'états en plus de transmettre périodiquement les milliers de messages de rafraîchissement ce qui ferait grandement diminuer les performances de

l'équipement. Pour ces raisons, l'usage de services intégrés n'est pas recommandé dans un large réseau.

2.2.1.2 DiffServ

DiffServ a été principalement développé afin d'éliminer la contrainte présente dans IntServ [14]. En effet, le problème de déploiement d'IntServ est résolu avec DiffServ puisque ce dernier traite les trafics par agrégats au lieu de les traiter individuellement. Son principe de fonctionnement consiste à définir d'abord plusieurs classes (un maximum de 64 classes) dans lesquels seront agrégés plusieurs flots. Notez que les différentes classes servent à différentier les types de flots. Le fonctionnement consiste à classer et marquer les différents flots dès l'entrée dans le réseau. Ainsi la marque appliquée à un flot identifie le comportement que les routeurs suivants doivent appliquer à ce flot. Par conséquent, les équipements au cœur du réseau se baseront sur cette marque pour classer les différents flots de façon appropriée et appliqueront un certain comportement à chaque classe.

Cette façon de procéder est beaucoup plus efficace que l'approche IntServ et ne dégrade pas les performances comme le fait la précédente approche. Cependant, l'usage de DiffServ nécessite l'emploi de divers mécanismes de QdS tel la classification, le marquage, l'ordonnancement, l'évitement de congestion, le lissage ainsi que le *policing* du trafic qui seront tous présentés dans les sections suivantes.

2.2.2 La classification

Afin de pouvoir différentier un type de trafic par rapport aux autres, il est tout d'abord indispensable de le classer. Cette classification est basée sur des critères plutôt variés tel les numéros de port TCP ou UDP, la valeur du champ *DiffServ code Point* (DSCP) de l'entête IP, la valeur du champ *Class of Service* (CoS) de l'entête *Virtual LAN* (VLAN)

ou encore la valeur du champ EXP de l'entête protocole *Multi Protocol Label Switching* (MPLS). Dépendamment de l'endroit où la classification est effectuée dans le réseau, les critères utilisés varieront. Comme par exemple, en bordure du réseau, la classification sera effectuée sur l'information obtenue dans les entêtes des protocoles de couche transport (UDP/TCP) du modèle *Open System Interconnection* (OSI). Dans le cœur du réseau, lorsque les paquets seront routés, la classification sera plutôt basée sur l'entête de couche réseau du modèle OSI (IP/DSCP) tandis que lorsque les paquets seront commutés au niveau 2 (liaison) du modèle OSI, la classification devrait plutôt s'effectuer sur les entêtes de couche 2 tel que MPLS/EXP ou VLAN/CoS. Bien que ces critères soient ceux typiquement utilisés, il est tout de même possible d'utiliser d'autres critères de classification telle que les adresses IP, le type de protocole utilisé ou encore une adresse URL spécifique.

Une fois le trafic classé, il est possible d'y appliquer une action quelconque. Le genre d'action pouvant être effectué est également varié. Il peut entre autre s'agir d'effectuer un marquage bien spécifique, d'effectuer de l'ordonnancement, du lissage, du WRED, etc... La classification peut s'effectuer un peu partout dans le réseau (là où nécessaire) et ce, tant sur le trafic entrant que sur le trafic sortant d'une interface.

2.2.3 Le marquage

Le marquage est une des actions pouvant être appliquée aux différentes classes de trafic. Il consiste à marquer un paquet afin de lui définir un type de service offert. Le type de traitement qui sera attribué à ce paquet, ultérieurement dans le réseau, sera basé sur la marque qui lui a été attribuée lorsqu'il est entré dans celui-ci. Il est donc primordial de donner des marques aux paquets en fonction de ce qu'ils transportent de sorte que ces paquets soient traités selon leurs exigences. Les champs utilisés pour le marquage peuvent être le champ DSCP de l'en-tête IP, le champ CoS pour les VLAN, ou encore le champ EXP de l'en-tête MPLS.

Notez que les champs MPLS/EXP et VLAN/CoS ne sont constitués que de trois bits et peuvent par conséquent définir jusqu'à 8 niveaux de priorité (0 à 7). Le champ IP/DSCP quant à lui utilise 6 bits et peut définir jusqu'à 64 niveaux de priorité (0 à 63). Notez que la RFC 2597 [15] et 2598 [16] ont catégorisé l'usage des bits DSCP en définissant une classe prioritaire *Expedited Forwarding* (EF) ainsi que 4 classes *Assured Forwarding* (AF), puis en définissant 3 priorités de rejet pour chacune des classes AF. Le Tableau IV présente un résumé des noms des *Per Hop Behavior* (PHB) avec la valeur binaire du *Class Selector* et du D*rop Precedence* associés. Cependant, bien que les RFCs [15] et [16] définissent l'usage de ces bits, il revient à l'administrateur du réseau de définir le vrai comportement à adopter pour chacune de ces classes en configurant les mécanismes de QdS de façon appropriée.

Tableau IV

PHB DiffServ avec *class selector* et *drop precedence* associés

PHB	**Class Selector**	**Drop Precedence**	
EF	101	110	
AF41	100	010	Low
AF42	100	100	Medium
AF43	100	110	High
AF31	011	010	Low
AF32	011	100	Medium
AF33	011	110	High
AF21	010	010	Low
AF22	010	100	Medium
AF23	010	110	High
AF11	001	010	Low
AF12	001	100	Medium
AF13	001	110	High

Un dernier point extrêmement important lorsqu'il s'agit de marquage est la définition d'une frontière de confiance. C'est-à-dire définir un équipement à partir duquel les paquets sont marqués de manière fiable. Puisqu'un client pourrait volontairement marquer tous ses paquets à haute priorité afin d'avoir un meilleur service, il est donc nécessaire de remarquer les paquets de manière correcte dès leur entrée dans le réseau et ce, même si ceux-ci possèdent déjà une marque.

2.2.4 L'ordonnancement

Il a été vu précédemment que la classification consiste à trier le trafic et à le regrouper par classes selon le type de chacun. Le mécanisme d'ordonnancement est utilisé afin de gérer les différentes files d'attente (ou classe) du routeur en cas de congestion. Cette technique est principalement utilisée afin de traiter le niveau de priorité de chaque classe et de gérer la bande passante qui leur est allouée. Elle réorganise en fait l'ordre de sortie des paquets afin de leur donner la qualité de service à laquelle ils ont droit. Cette section présentera brièvement les principaux mécanismes d'ordonnancement ainsi que d'autres plus spécifiques à l'équipementier Cisco Systems.

2.2.4.1 Priority Queueing (PQ)

Cet algorithme est l'un des plus simples algorithmes d'ordonnancement [17]. Le principe est illustré à la Figure 4 et consiste à classer d'abord le trafic et à traiter en premier lieu la file d'attente la plus prioritaire. Lorsque aucun paquet n'est présent dans cette file, l'ordonnanceur passe à la file suivante. Si un paquet est présent, il le traite et retourne traiter la file la plus prioritaire. Si aucun paquet n'est présent, il passe à la file suivante. Cet algorithme a l'avantage de pouvoir traiter en priorité les paquets les plus prioritaires et en second lieu les autres. Cependant il comporte comme principal inconvénient le simple fait que si la file d'attente la plus prioritaire n'est jamais vide, les

files moins prioritaires ne seront jamais traitées ce qui peut compromettre la qualité de service des applications moins prioritaires.

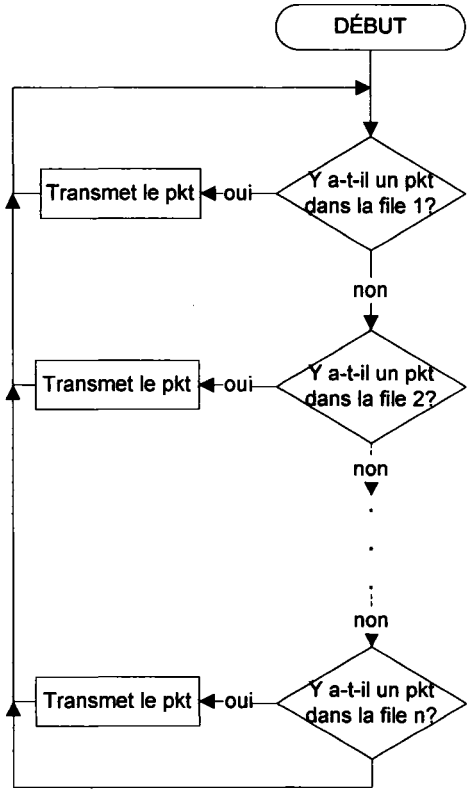


Figure 4 Fonctionnement de l'algorithme *Priority Queueing*

2.2.4.2 Round Robin (RR) et Fair Queueing (FQ)

Ces algorithmes ont été développés afin d'offrir un service équitable entre toutes les files d'attentes. Le principe de l'algorithme RR est simple, il consiste à parcourir chaque classe en boucle, les unes à la suite des autres, et à traiter un seul paquet par classe à chaque itération. Par conséquent, une classe moins occupée peut être traitée même si d'autres classes sont surchargées. Le problème avec ce mécanisme est qu'il ne fait pas de distinction entre les différentes tailles de paquets. Ainsi une classe où les paquets sont

plus grands recevra plus de capacité qu'une classe avec des paquets plus petits. (Voir aussi [18])

L'algorithme *Fair Queuing* [17] fonctionne de façon très similaire mais il a principalement été développé pour l'usage au sein d'un réseau informatique. Il classifie d'abord les paquets en fonction des flots auxquels ils appartiennent. Par la suite, il insère chaque flot dans une file d'attente propre à ce flot. Ensuite chaque file est parcourue octets par octets. Lorsqu'un paquet a fini d'être parcouru, celui-ci est inséré dans la file de sortie (Figure 5). Cette façon de faire est beaucoup plus équitable que le *Round Robin* puisque les classes ayant des plus petits paquets ne sont pas désavantagées. Cependant cet algorithme a le désavantage de ne pas pouvoir définir des poids (ou une bande passante) par classe.

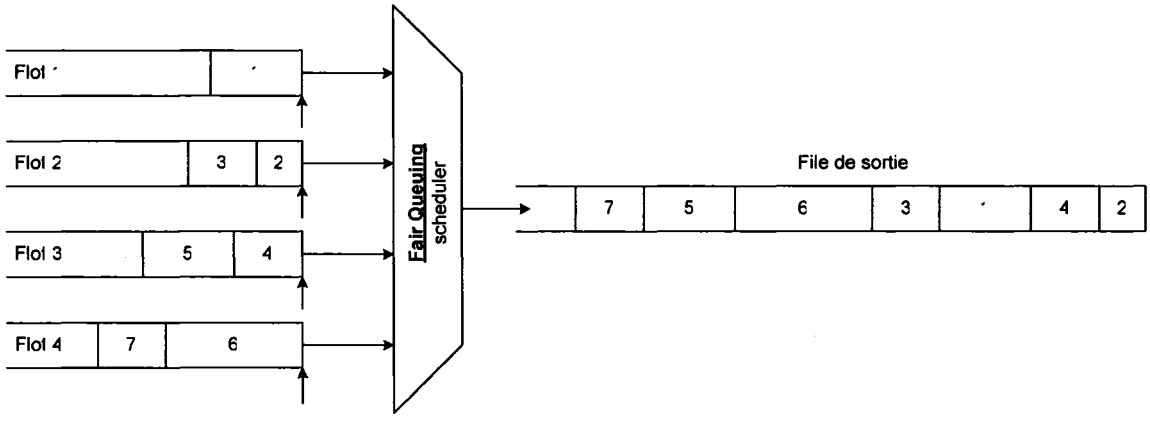


Figure 5 Fonctionnement de l'algorithme *Fair Queueing*

2.2.4.3 Class-Based Queueing (CBQ)

Cet algorithme d'ordonnancement est beaucoup plus adapté aux besoins des fournisseurs de service que les algorithmes précédemment décrits. D'abord il permet à l'administrateur de définir les types de trafic qui vont dans chacune des classes. Par la suite il lui permet de définir un poids pour chaque file d'attente (voir [19]). Par

conséquent, lorsque chaque file d'attente est pleine, la file d'attente j recevra une capacité égale à

$$C_j = \frac{W_j}{\sum_{\forall i} W_i} \times C \tag{2.1}$$

où W_j correspond au poids de la classe j, $\sum_{\forall i} W_i$ représente la somme des poids de toutes les files d'attente et C correspond à la capacité de l'interface. Le principal inconvénient avec cet algorithme est qu'il ne permet pas de prioriser un trafic dont les critères de QdS sont élevés. Cet inconvénient peut être réglé en ajoutant à cet algorithme une file d'attente de type *Priority Queue*. Cet ajout peut cependant comporter les mêmes problèmes soulevés dans la section 2.2.4.1. Cisco a défini un autre type de file d'attente qui sera vu à la section suivante.

2.2.4.4 Low Latency Queue (LLQ)

Une file d'attente de type LLQ (voir [17]) est utilisée pour diminuer autant que possible les délais des paquets mis dans cette file. En utilisant ce type de file, conjointement avec le CBQ, une file ayant une priorité supérieure à toutes les autres est créée. La LLQ utilise le principe du *Priority Queueing* afin que cette file soit vérifiée en premier à chaque fois qu'une place se libère dans la file de sortie. La différence avec le PQ consiste à imposer une limite de bande passante pour cette classe. Ainsi cette file d'attente ne pourra monopoliser toute la file de sortie au détriment des autres files d'attentes.

Tel que représenté dans la Figure 6, la file d'attente contenant les paquets prioritaires (LLQ) sera servie immédiatement puis, lorsqu'elle est vide ou qu'elle a atteint la bande passante maximale allouée, le routeur servira les files suivantes. Les informations contenues dans cette figure ont été prélevée implicitement de [17] mais leur véritable

algorithme n'est pas publié. Cependant, cette figure présente sans doute une bonne approximation du véritable algorithme.

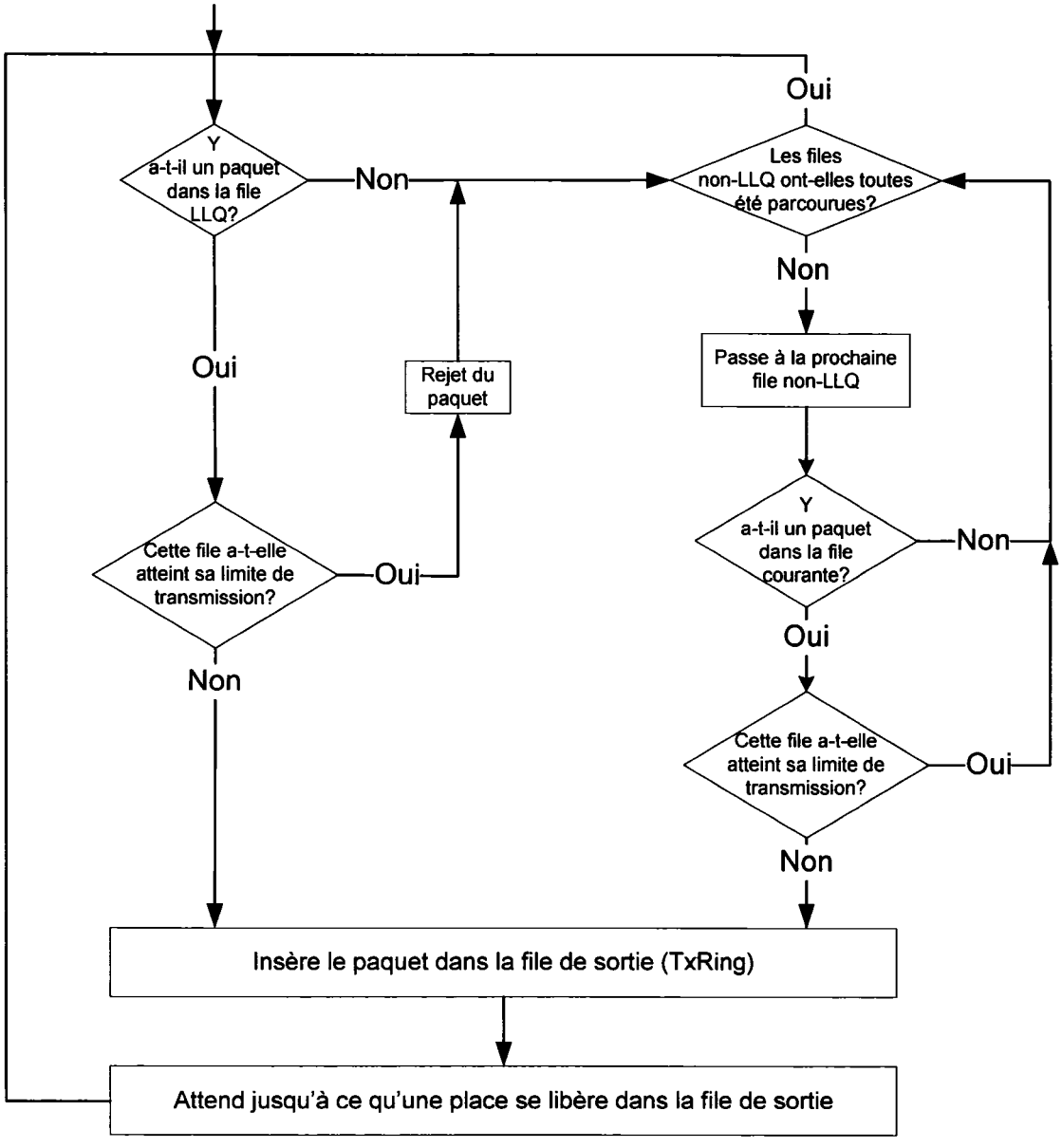


Figure 6 Inter fonctionnement entre une file LLQ et les files CBQ

2.2.5 Le lissage

Cette section présente le mécanisme de lissage utilisé pour réguler le trafic à un débit préalablement fixé. Avant d'expliquer le fonctionnement de ce mécanisme, il est important de connaître le rôle de certains paramètres de configuration.

Tableau V

Définition des paramètres de *shaping* et de *policing*

Terme	Définition
Tc	Intervalle de temps (en millisecondes) auquel la rafale Bc est transmise. De façon générale Tc = Bc/CIR
Bc	Nombre de bits contenu dans chaque rafale.
CIR	Débit défini dans le contrat de service (en bits/seconde).
Be	Nombre de bits, supérieur à Bc, pouvant être transmis ou reçus après une certaine période d'inactivité.

Concernant la rafale Bc, lorsque celle-ci n'est pas configurée, elle est calculée en multipliant Tc par CIR. Il est toutefois possible de modifier l'intervalle de temps Tc implicitement en configurant la rafale Bc. Ainsi, Tc devient égal à Bc/CIR. Notez que de façon générale, une rafale arrive ou sort de l'équipement à un taux égal au taux physique de l'interface. Par conséquent, la rafale ne dure pas nécessairement la totalité de l'intervalle Tc. Un exemple a été réalisé à la Figure 7 pour pouvoir mieux comprendre le fonctionnement du lissage et le calcul des paramètres. Dans cet exemple, le taux de lissage est de 96 kbps sur un lien ayant une capacité de 128 kbps. Notez que seul le paramètre CIR a été configuré. Par conséquent le paramètre Bc a été calculé à partir du CIR et de la valeur par défaut de Tc.

Idéalement le lissage devrait être activé sur le trafic sortant d'une interface afin de limiter le trafic à un débit bien précis. Ce débit pourrait avoir été préalablement établi dans un contrat entre le client et le fournisseur de service. Par conséquent, le routeur émet par intervalle de temps et utilise des tampons afin de réguler le trafic au débit de sortie moyen souhaité. Des délais sont occasionnés aux paquets qui ont été insérés dans le tampon. De ce fait, ce mécanisme peut être problématique pour des applications temps réel comme la voix. Veuillez noter que ce mécanisme est actif uniquement lorsque le débit, devant être transmis, dépasse le débit défini dans le contrat. Dans le cas contraire, les paquets seront transmis directement et le mécanisme restera inactif.

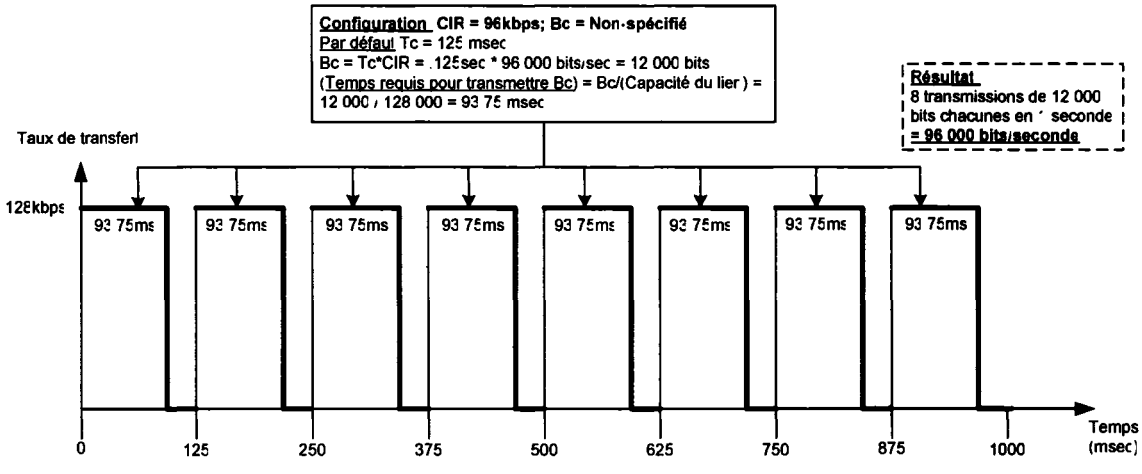


Figure 7 Exemple de calcul des paramètres de lissage

2.2.6 Le policing

Le *policing*, quant à lui, est appliqué sur le trafic entrant une interface afin de s'assurer que ce trafic est conforme au contrat établi. Le fonctionnement des divers paramètres décrits dans le Tableau V (CIR, Bc, Be) est identique dans le cas du *policing*. La façon de les configurer diffère légèrement puisque le *burst-normal* correspond à Bc tandis que le *burst-max* correspond à l'addition de Bc et Be. La principale différence avec le *shaping* est que pour le *policing*, il est possible de poser des actions sur le trafic.

La Figure 8 présente le fonctionnement du *policing* en utilisant l'algorithme *Token Bucket*. Notez que cette méthode de fonctionnement est tirée de [17]. Sachant qu'un jeton équivaut à un octet, à chaque fois qu'un paquet est reçu, un certain nombre de jetons sont ajoutés dans le *bucket* Bc. Si ce *bucket* est plein lorsque les jetons y sont ajoutés, les jetons excédentaires iront dans le *bucket* Be. De façon similaire, si le *bucket* Be est rempli de jetons et qu'il reste toujours des jetons excédentaires, ces jetons seront perdus. Le nombre de jetons à insérer dans le *bucket* est calculé de la façon suivante :

$$\frac{(Current\_pkt\_arrival\_time - Previous\_pkt\_arrival\_time) \times Police\_Rate}{8}$$

où les temps d'arrivés du paquet nouvellement reçu (*Current*) et du précédent (*Previous*) sont en seconde et le taux de *policing* est en bits par seconde. Par conséquent, le tout est divisé par huit afin d'obtenir un nombre de jetons.

À chaque fois qu'un paquet entre dans un *bucket*, un nombre de jetons égal au nombre d'octets dans le paquet est retiré du *bucket*. Ainsi les deux *buckets* peuvent accepter la venue de paquets pourvu que la taille des paquets n'excède pas le nombre de jetons dans le *bucket*. Si un paquet entre dans l'équipement et qu'il y a suffisamment de jetons dans le *bucket* Bc, alors le paquet sera conforme et sera traité comme tel (*Conform Action*). S'il n'y a pas suffisamment de jetons dans le *bucket* Bc mais qu'il y en a suffisamment dans le Be pour accepter le paquet, alors celui-ci est considéré excédentaire et subira l'action destinée à ces paquets (*Exceed Action*). S'il n'y a pas assez de jetons ni dans le *bucket* Bc ni dans le Be, alors ce paquet sera considéré comme un paquet dépassant le contrat et subira l'action adéquate (*Violate Action*). Le genre d'actions portées aux paquets peut être de le transmettre inchangé, le remarquer et le transmettre ou carrément le rejeter. Un scénario adéquat pourrait être de transmettre inchangés les paquets conformes, de remarquer avec une priorité de rejet plus élevée les paquets excédentaires et rejeter les paquets dépassant le contrat.

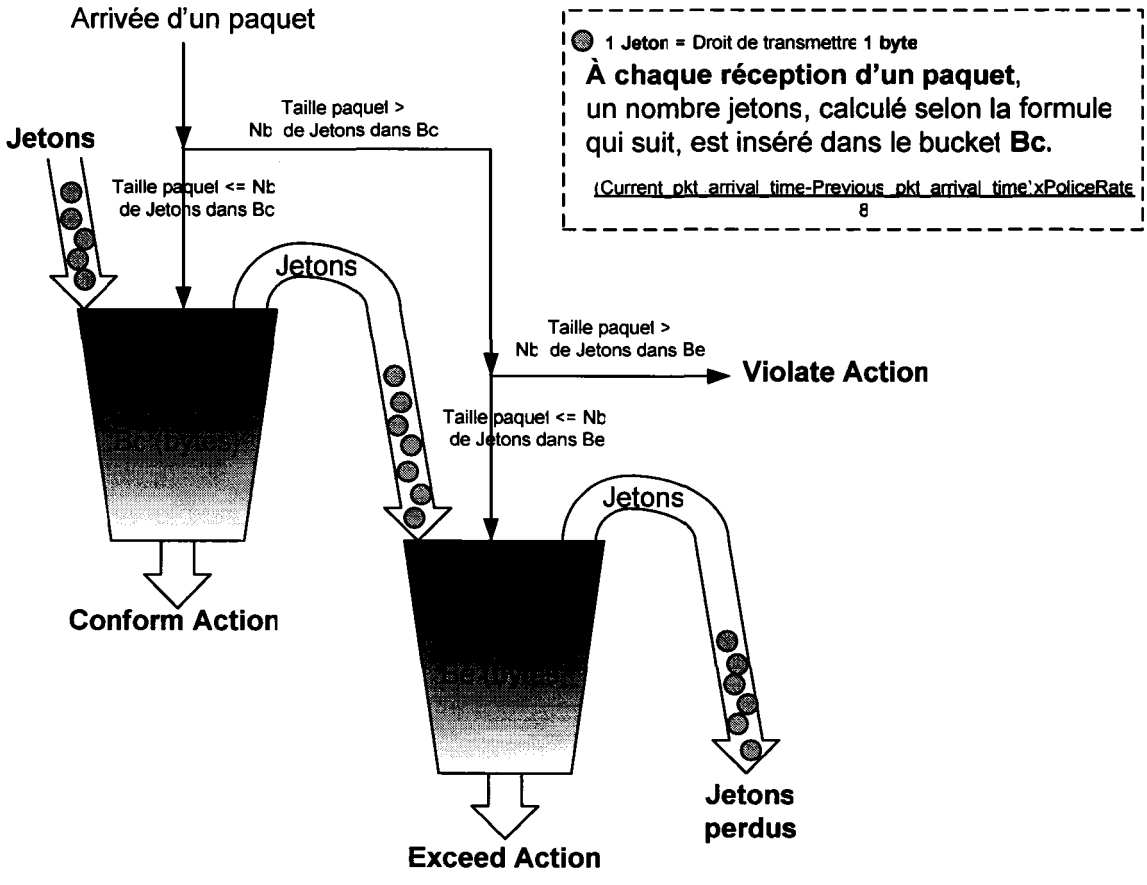


Figure 8 Gestion du *policing* par l'algorithme *Token Bucket*

2.2.7 L'évitement de congestion

Un bon moyen d'éviter la congestion dans un routeur est de l'anticiper. Le RED [17] profite du mécanisme de contrôle de congestion de TCP en rejetant des paquets selon un certain pourcentage. L'émetteur, n'ayant pas reçu d'acquittement durant un certain temps, réduira de moitié son taux de transmission puis recommencera à l'augmenter progressivement. De ce fait, l'application de RED sur du trafic UDP n'a aucun effet significatif sur le niveau de congestion. Il y a trois principaux paramètres à configurer pour le RED. Premièrement ce mécanisme permet de définir un premier seuil (Minimum Threshold), à partir duquel l'équipement commence à rejeter des paquets avec un certain

taux. Deuxièmement, le taux de rejet maximal peut être configuré en modifiant le paramètre *Mark Probability Denominator* (MPD). Ce taux de rejet maximal correspond en fait à 1/MPD. Finalement il est possible de définir le seuil maximal (Maximum Threshold) à partir duquel l'équipement rejettera 100% des paquets. Il est donc logique de configurer ce paramètre à la taille maximale de la file d'attente. Ainsi lorsque la file est pleine, 100% des paquets sont rejetés. Notez que lorsque le niveau de remplissage de la file se situe entre les deux seuils (Min et Max Threshold), le taux de rejet augmentera proportionnellement dépendamment d'où le niveau de remplissage se situe entre les deux seuils. Il devrait normalement atteindre un taux égal à 1/MPD une fois rendu à la limite du seuil maximal.

Par conséquent, le remplissage total des files d'attente est retardé et la congestion est prévenue. La différence entre le RED et le WRED [17] est que le WRED peut s'appliquer sur des classes de trafic afin de rejeter les paquets en fonction de leur niveau de priorité.

Concernant le WRED, il est clair que les paquets qui ont une priorité de rejet (*Drop Precedence*) plus élevée doivent être jetés les premiers. Le seuil minimum associé à une priorité de rejet élevée sera donc inférieur aux seuils minimums pour les paquets avec une priorité de rejet moins élevée et ce, afin que le mécanisme se déclenche plus tôt pour les paquets moins prioritaires. De plus un pourcentage de rejet plus élevé peut également être défini pour les paquets ayant une priorité de rejet élevée. Par conséquent, une classe pouvant contenir des paquets avec le même niveau de priorité mais avec des priorités de rejet différentes, pourrait définir des seuils minimums pour chaque priorité de rejet afin de rejeter les paquets dont la priorité de rejet est la plus élevée en premier.

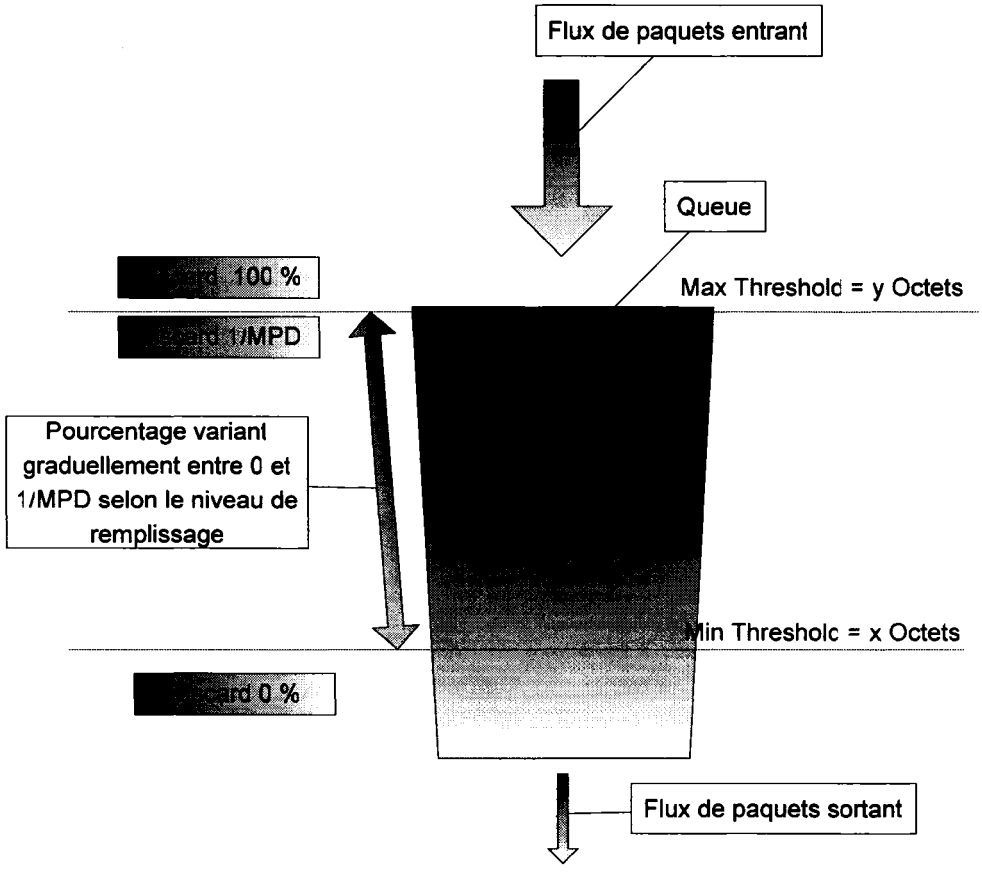


Figure 9 Fonctionnement du RED

2.3 La surveillance d'un réseau

Cette section présente un survol des différents protocoles et/ou des différentes architectures proposées pour effectuer la surveillance d'un réseau de télécommunications, plus particulièrement un réseau IP. On dénote principalement deux standards développés pour effectuer la surveillance d'équipements de télécommunications soit *Telecommunication Management Network* (TMN) et *Simple Network Management Protocol* (SNMP). Dans les deux cas, le principe est basé sur l'architecture Client/Serveur. Dans un couple *Gestionnaire/Agent*, le *gestionnaire* est le système qui interroge les agents situés sur les équipements réseau afin d'obtenir une information quelconque. Le *gestionnaire* est également nommé un *Network*

Management System (NMS). L'*agent* est, quant à lui, situé dans les équipements réseau et est en charge de répondre aux requêtes du NMS. De plus, peu importe le standard utilisé, les agents organisent leurs informations en suivant une structure arborescente nommée *Management Information Base* (MIB). Notez que [20] introduit un filtre permettant l'accélération de la recherche dans une MIB. Plus la MIB est grande, plus l'usage de leur filtre permet une diminution significative des délais de recherche.

Les sous-sections qui suivent présentent une introduction à ces deux standards largement utilisés dans les réseaux de télécommunications. Notez que [21] effectue une comparaison de ces deux standards tout en faisant ressortir les défis de la gestion de réseau dans les réseaux de prochaine génération.

2.3.1 TMN

TMN est un standard défini par l'ITU-T dans la série de recommandations M-3000. Il est basé sur le modèle OSI et a été développé pour gérer les réseaux de télécommunications en général. Bien qu'il ait été développé pour gérer tout type de réseau, son implémentation au sein d'un réseau IP peu s'avérer relativement complexe. Par exemple, la majorité des équipements IP supportent SNMP et non TMN, une passerelle de gestion (également nommée Q *Adapter*) est donc nécessaire afin de faire le lien entre le protocole *Common Management Information Protocol* (CMIP - [22]) et SNMP (voir [23]). Le protocole CMIP est l'équivalent du protocole SNMP dans l'architecture TMN. TMN comporte cinq fonctionnalités soit :

a. La gestion des configurations
b. La gestion des fautes
c. La gestion des performances
d. La gestion comptable
e. La gestion de la sécurité

En référant au Tableau I, on peut s'apercevoir que TMN supporte les mêmes fonctionnalités que le plan de gestion. Le Tableau VI présente, quant à lui, la liste des services *Common Management Information Service* (CMIS - [24]) offerts par TMN. On y retrouve également une courte description de ces services. On y retrouve, entre autres, des services permettant d'obtenir ou de modifier la valeur d'un objet, des services permettant d'obtenir des notifications lorsque certains événements surviennent mais également, des services permettant de créer ou de supprimer des objets.

Tableau VI

Définition des services offerts par TMN

Services	**Descriptions**
M-GET	Ce service est utilisé pour obtenir les valeurs des attributs d'un objet géré. Les paramètres de la portée et de filtrage peuvent être employés afin de sélectionner un groupe d'objets gérés pour lesquels on demande les valeurs des attributs.
M-CANCEL-GET	Ce service est utilisé pour arrêter un service M-GET en cours.
M-SET	Ce service est utilisé pour modifier les valeurs d'un attribut d'un objet géré.
M-EVENT-REPORT	Ce service est utilisé par un objet pour signaler un changement d'état ou une erreur. L'objet génère la notification qui est envoyée par l'agent au *manager*.
M-ACTION	Ce service est utilisé pour effectuer des actions sur un objet géré. Il a été introduit pour permettre à un mécanisme d'effectuer des opérations autres que M-GET, M-SET, M-CREATE. Quand un objet est créé, on définit aussi les opérations qu'on peut effectuer sur lui.
M-CREATE	Ce service est utilisé pour créer une nouvelle instance d'un objet géré. On spécifie aussi les valeurs d'information gérée.

Tableau VI (suite)

Services	Descriptions
M-DELETE	Ce service est utilisé pour effacer une instance d'un objet géré. Les paramètres de la portée et de filtrage peuvent être employés afin d'effacer un groupe d'objets gérés.

2.3.2 SNMP

SNMP est un standard défini par *Internet Engineering Task Force* (IETF), développé pour fonctionner au sein d'un réseau IP en utilisant la pile de protocole du modèle TCP/IP. Il utilise UDP comme protocole de couche transport. Il est le standard le plus largement déployé dans les réseaux IP dû à sa simplicité d'implémentation et d'utilisation. Trois versions ont vu le jour jusqu'à maintenant soit SNMPv1 [25], SNMPv2 [26] et SNMPv3 [27]. Le Tableau VII présente l'évolution des services offerts par SNMP selon la version de protocole utilisée.

Tableau VII

Définition des services offerts par les trois versions du protocole SNMP

Services	Version	Description
GET/GET_NEXT	1	Ces services sont utilisés pour demander à un agent de retourner la valeur d'un attribut d'un objet géré.
SET	1	Ce service est utilisé pour demander à un agent de modifier la valeur d'un attribut d'un objet géré.
TRAP	1	Ce service est utilisé par un objet géré pour signaler un changement d'état ou une erreur. L'agent envoie alors la notification au NMS.

Tableau VII (suite)

Services	Version	Description
GET_BULK	2	Ce service permet le transfert de plusieurs valeurs d'attributs d'objets gérés en un seul transfert.
Sécurité	3	Permet l'authentification et l'encryption de données.

Tel que stipulé dans [28], la première version offre des services de base utilisés pour aller chercher ou modifier les attributs d'un objet ou encore pour obtenir une notification lorsque certains événements surviennent. La seconde version, quant à elle, permet l'obtention d'une plus grande quantité d'information à l'aide d'une seule commande (GET_BULK). Une autre fonctionnalité importante de la version 2 est la gestion décentralisée qui fera ses preuves surtout dans la gestion des réseaux étendus. En fait, la communication NMS à NMS est dorénavant possible avec cette version, permettant ainsi l'établissement d'une structure de gestion hiérarchique. Par conséquent, des NMS pourraient être répartis de façon géographique où chacun gèrera sa portion du réseau et chacun d'eux pourraient communiquer avec un autre NMS père qui aura une vue plus globale du réseau. Après avoir constaté un manque flagrant de sécurité dans les deux premières versions, la troisième version a vu le jour offrant alors un service d'authentification et d'encryption des données.

2.3.3 TMN vs. SNMP

De façon générale, SNMP est beaucoup plus adapté à la gestion des réseaux IP que TMN [21]. Comme le font ressortir les auteurs, d'abord parce que TMN est beaucoup plus complexe que SNMP qui offre une architecture simple et une plus grande facilité d'utilisation. Du point de vue des fonctionnalités, les deux approches sont plutôt similaires à l'exception que TMN est considéré plus sécuritaire que SNMP. Cette différence réside principalement dans le fait que TMN utilise un réseau de gestion physiquement séparé du réseau géré ce qui le rend physiquement sécuritaire

comparativement à un réseau partagé comme IP. En considérant l'aspect multi-vendeur, les deux approches permettent de supporter cet aspect pourvu que les vendeurs implémentent les interfaces adéquates au standard choisi. Globalement, SNMP est préféré à TMN par les vendeurs, dû à la complexité de ce dernier et à la popularité du premier. Du point de vu des communications, TMN utilise la pile de protocoles du modèle OSI tandis que SNMP utilise la pile de protocoles du modèle TCP/IP. En considérant que les protocoles du modèle OSI sont rarement supportés dans les LAN et les réseaux étendus (WAN), SNMP est le choix privilégié. Du point de vue de l'implémentation, SNMP est beaucoup plus simple à implémenter dû à la simplicité du standard. Cependant, [29] fait ressortir qu'il y a eu plusieurs problèmes de vulnérabilité du protocole SNMP qui sont principalement dus à une mauvaise implémentation du standard pour la majorité des fabricants. De plus, ils notent que certaines de ces vulnérabilités persistent dans les dernières versions du standard et que celles-ci peuvent permettre à un attaquant de gagner des privilèges dans le système ou encore qu'ils peuvent causer des conditions de déni de service.

La section qui suit présente un survol des différents systèmes de surveillance présents dans le marché.

2.3.4 Survol du marché

Cette sous-section a été réalisée dans le but de faire un survol du marché des systèmes de gestion de la QdS. Ce marché est plutôt restreint et il n'existe pas encore, à ce jour, de système permettant la surveillance des mécanismes de QdS dans un environnement réseau hétérogène.

2.3.4.1 *QoS Policy Manager* (QPM)

Cet outil a été développé par l'équipementier Cisco System Inc. et permet principalement de définir des politiques de service et de configurer un ou plusieurs équipements selon la politique définie. Il permet également une surveillance des configurations afin d'informer l'administrateur du réseau d'un changement de configuration. De plus, ce système permet la visualisation de statistiques pour les différents mécanismes de QdS. Les statistiques sont récoltées par le biais de la MIB CBQoSMIB qui est propriétaire Cisco. Son architecture est présentée à la Figure 10 où on peut voir qu'il utilise le protocole SNMP pour interroger les équipements du réseau et Telnet et *Common Open Policy Service* (COPS) [30] pour configurer les équipements lorsque requis. QPM fait partie de Cisco Works et permet une gestion centralisée de la QdS.

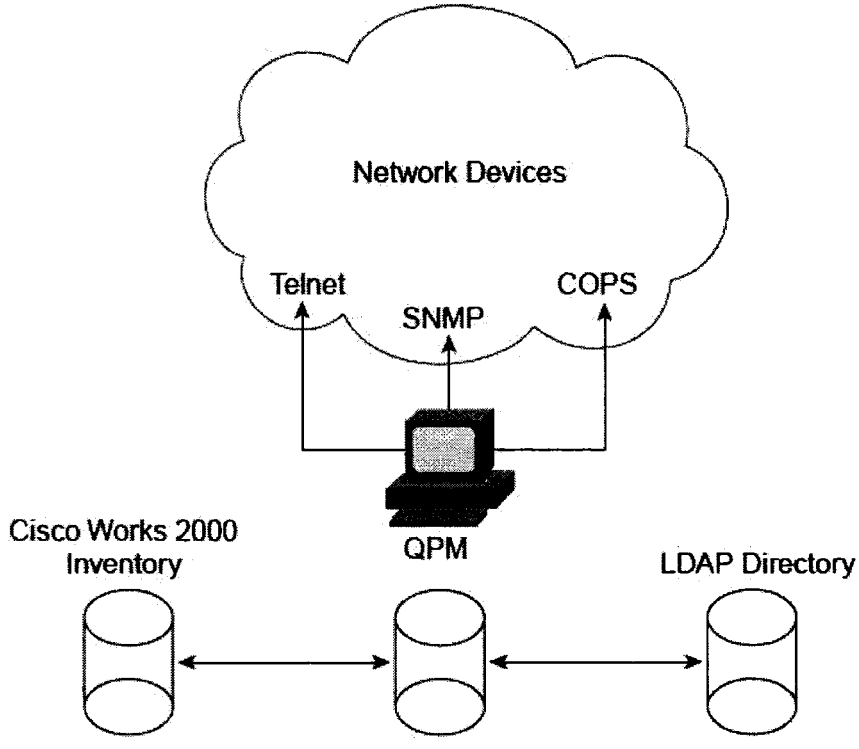


Figure 10 Architecture de QPM (tiré de [17])

Bien que ce système puisse répondre aux critères élaborés dans la problématique, le principal inconvénient avec les outils propriétaires réside dans le fait qu'ils ont été conçus pour un seul et unique équipementier. Par conséquent, ce type d'outil n'est pas tout à fait approprié dans un réseau hétérogène, ce qui est très souvent le cas chez les fournisseurs de service.

2.3.4.2 Netscout nGenious

Cette section présente un système de gestion développé par l'entreprise *NetScout* (http://www.netscout.com/) qui se spécialise dans le développement de système de surveillance réseau. Par conséquent, leur système n'est pas limité à un seul équipementier et peut donc offrir la possibilité de surveiller un réseau hétérogène. *Netscout nGenious* est un système permettant la surveillance des performances d'un réseau de la couche 2 à la couche 7 du modèle OSI. Son principe de fonctionnement est représenté à la Figure 11. Pour l'expliquer brièvement, des sondes sont insérées dans le réseau afin de récolter le trafic qui y circule et en dériver des statistiques sur la performance des liens. Le serveur local récolte les informations afin que le serveur global puisse les afficher, au client, dans un format approprié. De plus, les serveurs locaux peuvent interroger les routeurs et les commutateurs afin de récolter des informations générales par le biais du protocole SNMP.

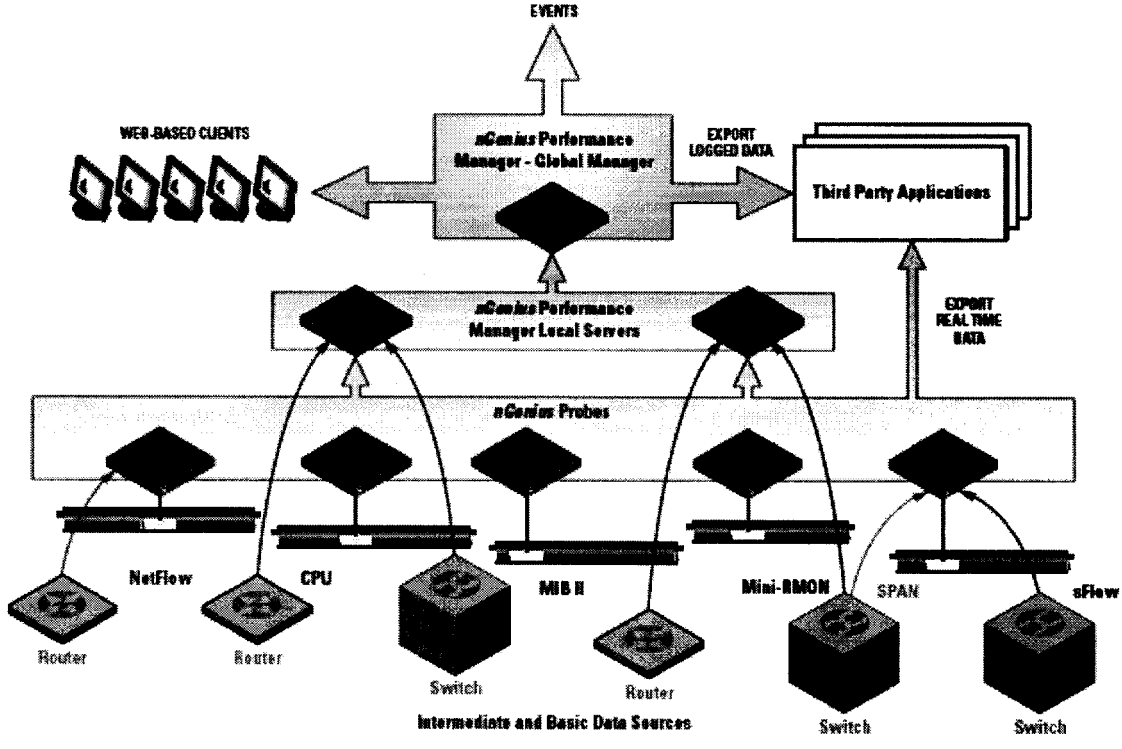


Figure 11 Structure générale de *Netscout nGenious* (http://www.netscout.com/)

Ce système permet de visualiser l'utilisation de la capacité des liens par type d'application et même par utilisateur. De plus, il est en mesure de générer des rapports sur mesure journaliers, hebdomadaire ou mensuels. Cependant, le seul moyen possible pour surveiller la QdS est de créer des filtres IP/DSCP appropriés. Le système présentera alors les statistiques d'utilisation en fonction des filtres et, pour un filtre spécifique, la répartition de la charge entre les diverses applications. Par conséquent, ce système est plutôt mal adapté à la surveillance des mécanismes de QdS puisqu'il ne permet que de connaître le débit associé à chaque classe de service et n'est pas en mesure de donner des statistiques sur les autres mécanismes de QdS tel le lissage, le *policing*, ou l'évitement de congestion. Cet inconvénient est également valable pour d'autres modèles de gestionnaire de performance similaire à celui présenté dans la présente section. C'est d'ailleurs le cas du modèle SCE1000 de Cisco System.

2.3.4.3 WhatsUp et HP OpenView

Les systèmes de gestion WhatsUp de IPswitch (http://www.ipswitch.com/) et OpenView de HP (http://www.openview.hp.com/products/nnm/) sont deux exemples de systèmes complets de gestion. Tous les deux permettent une découverte de la topologie du réseau et permettent une surveillance des performances générales des équipements. De plus, des alarmes sont générées lorsque certains événements se produisent dans le réseau ou encore, lorsque l'utilisation des ressources dépasse un certain seuil.

Concernant la surveillance des mécanismes de QdS, WhatsUp n'intègre absolument pas cet aspect alors que HP OpenView permet la surveillance de la téléphonie IP. Ce dernier surveille les performances des équipements de VoIP en interrogeant principalement le *Call Manager* de Cisco. Ainsi, il peut retirer des statistiques sur les appels ainsi que donner des recommandations pour de meilleures performances et une plus grande disponibilité.

Ces systèmes de gestion permettent tous deux le développement de scripts[1]. Ainsi un utilisateur pourrait construire son propre script afin d'obtenir les informations qu'il désire. Cependant, les opérations pouvant être effectuées avec ces scripts étant plutôt limitées, le développement de scripts pour obtenir des informations concernant les mécanismes de qualité de service n'est pas à considérer. Après une plus grande investigation auprès de ces produits, on peut conclure que l'aspect de qualité de service n'a pas été abordé par ceux-ci.

---

[1] Série d'instructions servant à exécuter une tâche particulière (tiré de http://w3.granddictionnaire.com/).

2.4 Contexte du projet par rapport à l'état de l'art

Comme les NGN consistent en une architecture permettant, entre autre, une gestion de réseau dans un environnement multi-services et qu'un des rôles clé de QMA s'insère dans cette architecture, une part importante de ce chapitre a été consacrée à l'expliquer.

Par ailleurs, comme la qualité de service est un élément essentiel des réseaux multi-services et que ce mémoire se consacre spécifiquement à la surveillance de ces mécanismes, une introduction technique aux mécanismes de QdS ainsi qu'un survol des protocoles et systèmes de surveillance existants ont été réalisés.

Le chapitre qui suit présente la proposition d'un système permettant une surveillance de ces mécanismes. La solution proposée s'insère dans le contexte des NGN. Par conséquent, la place qu'elle y occupe sera décrite.

CHAPITRE 3

PROPOSITION

Les chapitres précédents ont d'abord fait ressortir les difficultés auxquelles font face les administrateurs de réseau quant aux configurations et à la surveillance des mécanismes de QdS dans un réseau hétérogène. Ces difficultés ont alors entraîné la nécessité, pour un administrateur, d'avoir un outil logiciel dont le but est de faciliter la gestion et la surveillance de ces mécanismes. De plus, l'incapacité des systèmes de gestion industriels de répondre à ces besoins, ont engendré la proposition d'un système dont les buts principaux sont de faciliter l'homogénéité des configurations des mécanismes de QdS dans un réseau hétérogène tout en permettant une surveillance de ceux-ci. Par conséquent, le présent chapitre décrit en détail la proposition de ce système nommé QMA.

Ce chapitre se divise en deux principales parties. D'abord le système développé dans le présent mémoire est situé dans le contexte des NGN selon les approches présentées à la section 2.1. Par la suite, les spécifications fonctionnelles et non fonctionnelles de QMA, concernant la QdS, sont présentées. Les spécifications concernant MPLS et l'ingénierie de trafic, qui sont parties intégrantes de QMA, ne seront pas abordées dans ce mémoire puisque cela fait partie du travail d'un autre membre de l'équipe.

3.1 QMA dans le contexte des NGNs

Cette section a pour but de situer le système QMA dans le contexte des réseaux de prochaine génération. Mais avant d'y parvenir, il est important de se rappeler les principales raisons d'être du système QMA. D'abord il faut préciser que QMA est un système dont le but est d'assister l'administrateur du réseau quant aux configurations et à la surveillance des divers mécanismes de QdS ainsi que tout ce qui est relié à MPLS,

plus particulièrement, l'ingénierie de trafic. Bien que la gestion et la surveillance des LSP est une partie importante de QMA et que celles-ci permettent d'avoir une vision encore plus globale du réseau, seul ce qui a trait à la QdS sera abordé et développé dans ce mémoire. Par conséquent, les principaux rôles de QMA, à l'égard de la QdS, sont résumés dans les paragraphes qui suivent.

Pour commencer, QMA doit faciliter la visualisation et la compréhension des configurations de QdS pour un administrateur réseau. Ainsi, ce dernier n'aura pas besoin de trier les informations de configuration de l'équipement et pourra voir directement ce qui l'intéresse.

Par ailleurs, QMA doit faciliter la surveillance des mécanismes de QdS. De plus cette surveillance doit pouvoir s'effectuer sur tout type d'équipement étant donné que les réseaux des fournisseurs de service sont généralement hétérogènes. Cette surveillance permettra de valider le bon fonctionnement des mécanismes ainsi que de détecter la ou les sources d'une dégradation de la QdS.

Par la suite, un des rôles de QMA est de faciliter la prise de décision de l'administrateur de réseau quant aux mécanismes de QdS. L'aspect de surveillance doit grandement aider à cette tâche puisque les statistiques présentées à l'administrateur pourront lui indiquer des classes ou mécanismes qui ont besoin d'être redimensionnés. De plus, étant donné que QMA est un outil logiciel, ce dernier peut dresser une liste de recommandations en se basant sur les statistiques récoltées. Il peut également assister l'administrateur du réseau, lors de la configuration de mécanisme, en lui fournissant de l'information générale sur ceux-ci et en lui fournissant des exemples typiques de configuration.

Un autre rôle de QMA est de faciliter la configuration de la QdS, de façon homogène, dans tout le réseau. Ainsi l'administrateur pourra s'assurer qu'il n'y a pas de maillon faible dans la chaîne de QdS. Étant donné que QMA permet la visualisation des

mécanismes de QdS pour chaque équipement du réseau, il devient plus aisé pour l'administrateur de valider la cohérence des configurations. De plus, comme QMA est un outil logiciel, il peut reproduire une configuration sur plusieurs équipements préalablement sélectionnés par l'administrateur. Ainsi l'homogénéité des configurations est plus facilement accessible pour les équipements sélectionnés.

Par ailleurs, l'administrateur peut définir des alertes lorsque certains événements surviennent. Ces alertes peuvent, par exemple, être générées par QMA lorsqu'une statistique dépasse un seuil défini par l'administrateur. Après avoir défini les alertes, l'administrateur peut, par la suite, définir certaines actions que le système QMA doit prendre lorsque ces événements surviennent. Ainsi le nombre de manipulations effectuées par l'administrateur sur les équipements réseau sera grandement diminué.

Étant donné que plusieurs systèmes QMA peuvent être déployés dans un réseau, QMA doit être capable de communiquer avec une entité paire distante. Ainsi un administrateur peut gérer l'ensemble du réseau en interagissant avec une seule entité QMA.

Finalement, étant donné que QMA fournit des statistiques sur l'utilisation des mécanismes de QdS, il est possible pour lui de déterminer des endroits, dans le réseau, où l'utilisation atteint un seuil critique susceptible de dégrader les performances du réseau. Par conséquent, il peut dresser une liste d'améliorations ou de modifications physiques susceptibles de préserver un niveau de performance acceptable. Les performances du réseau étant un facteur important dans la qualité de service, cette liste de modifications permettra, du même coup, de préserver une qualité de service acceptable.

3.1.1 Selon l'approche fonctionnelle

En se basant sur les plans de l'approche fonctionnelle présentés dans le Tableau I, QMA pourrait se situer au niveau du plan de gestion tout en travaillant étroitement avec le plan de contrôle. La Figure 12 présente la place que QMA prend dans les plans de l'approche fonctionnelle.

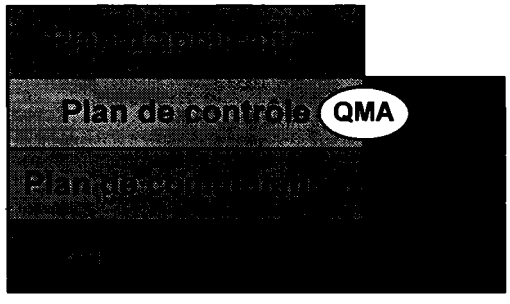


Figure 12 QMA dans le contexte des NGN, selon l'approche fonctionnelle

Parmi les fonctions du plan de gestion que QMA est en charge de supporter, nous retrouvons la gestion des configurations, la gestion des performances ainsi que la gestion des fautes.

QMA implémente la fonction de gestion des configurations puisqu'un administrateur réseau utilise QMA pour visualiser ou modifier les configurations de QdS dans le réseau. De plus, QMA peut prendre des décisions dynamiquement et ainsi modifier certaines configurations.

Par la suite, il implémente la fonction de gestion des performances puisque QMA est en charge de surveiller les divers mécanismes de QdS et de rapporter les problèmes de performances susceptibles de dégrader la qualité de service offerte.

Finalement, il implémente la fonction de gestion des fautes puisque QMA doit réagir dynamiquement à certains événements qui surviennent dans le réseau. Un exemple de problèmes susceptibles de survenir est un bris quelconque (équipement ou interface défectueux, bris de lien, etc...) et l'action que QMA pourrait adopter, dans cette situation, serait de rediriger le trafic vers un nouveau chemin (ingénierie de trafic ou TE) et reconfigurer les paramètres de QdS en conséquence.

Par ailleurs, QMA doit travailler étroitement avec le plan de contrôle. En effet, QMA doit s'occuper des configurations des mécanismes de QdS dans le réseau, du contrôle d'admission et de l'ingénierie de trafic. Ces configurations sont commandées par le plan de gestion. De plus, le plan de contrôle est en charge d'obtenir et de retourner les statistiques et les alarmes dont le plan de gestion a besoin pour effectuer la gestion des performances et des fautes respectivement.

3.1.2 Selon l'architecture de référence

En se basant sur l'architecture de référence présentée dans la Figure 2 et le Tableau III, il peut être conclu que QMA répond bien aux spécifications du *Bandwidth Manager*. En effet, un *Call Agent* SIP peut communiquer avec QMA lorsqu'il reçoit une nouvelle requête de connexion. Ainsi, il demande à QMA de s'assurer de la disponibilité de la bande passante, en reconfigurant les équipements si nécessaire, pour sa connexion.

Cette communication entre *Call Agent* et QMA (*Bandwidth Manager*) s'effectue par le biais de l'interface IF-2 telle que définie dans [31]. Si la connexion doit s'étendre à l'extérieur du domaine géré par le *Bandwidth Manager*, ce dernier communiquera avec une entité paire dans un autre domaine via l'interface IF-5 définie dans [32]. Afin de communiquer avec les équipements réseau, les interfaces IF-3 et IF-4 doivent être utilisées.

Cependant QMA a d'autres fonctionnalités que celles définies dans les spécifications du *Bandwidth Manager*. Nous retrouvons entre autre la surveillance des performances du réseau ainsi que la gestion des fautes.

L'application visée par l'approche NGN est de dynamiser les configurations statiques de QdS en se basant sur les nouvelles requêtes de connexion multi-media. Mais QMA pourrait aller plus loin, en faisant de la prédiction de trafic, par exemple, et en configurant les équipements avant même d'obtenir les requêtes de connexion du *Call Agent*.

3.2 Les spécifications du système QMA

Dans cette section, les fonctionnalités que doit supporter le système QMA sont définies. Les fonctionnalités générales de l'interface utilisateur sont d'abord présentées. Comme ces fonctionnalités ne sont pas spécifiques à la QdS, elles ont été regroupées dans une seule section.

Par la suite, les fonctionnalités spécifiques à la QdS sont introduites. On y retrouve entre autre la visualisation des configurations et des statistiques ainsi que la configuration semi-automatique puis la génération d'alertes et de rapports de performance.

Finalement, les spécifications non fonctionnelles sont développées. On y retrouve entre autre les spécifications par rapport à l'utilisation même du système, à la fiabilité, aux performances, aux supports technologiques ainsi qu'aux capacités d'intégration et de maintenance.

3.2.1 Les spécifications fonctionnelles générales de l'interface utilisateur

Cette section présente certaines fonctionnalités générales de l'interface utilisateur. Elles correspondent à des fonctionnalités relatives à la manipulation de l'interface utilisateur. Il est important de savoir que QMA n'est pas implémenté directement sur l'ordinateur de l'administrateur réseau mais qu'il permet à ce dernier de se connecter à l'entité de QMA en charge de la surveillance (voir Figure 13). Ainsi plusieurs administrateurs peuvent accéder au système QMA simultanément.

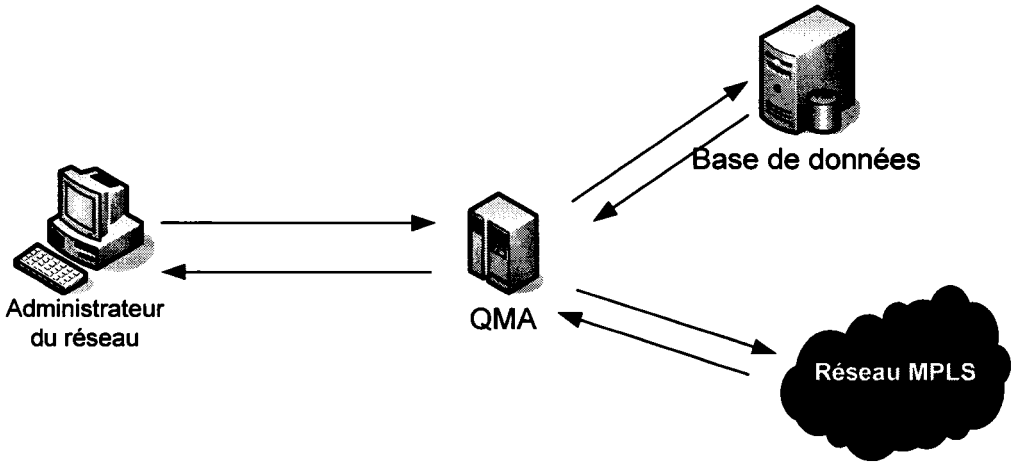


Figure 13 Architecture simplifiée du système QMA

Concernant les fonctionnalités générales de l'interface utilisateur, l'usager doit être en mesure de visualiser la topologie de tout le réseau. C'est-à-dire qu'il doit pouvoir voir les divers équipements ainsi que leurs interconnexions. Par ailleurs, l'usager doit visualiser facilement des informations générales d'un équipement ou d'un lien sélectionné. Ces informations peuvent inclure le nom ou le type d'équipement, la bande passante disponible, etc.

Par la suite, l'utilisateur doit avoir accès à des fonctionnalités générales et d'autres spécifiques à un équipement sélectionné comme par exemple, des options de QdS. De

plus, l'usager doit pouvoir rafraîchir les informations relatives à un équipement ou à tous les équipements du réseau s'il le désire.

Pour terminer, lorsque l'usager choisit une option, il peut décider ou non de faire apparaître une aide (sous forme de compagnon par exemple) dont le but est de guider l'usager pas à pas à travers les diverses actions qu'il doit poser pour obtenir les renseignements qu'il désire.

3.2.2 Les spécifications fonctionnelles pour la QdS

Cette section englobe les fonctionnalités relatives à la qualité de service. On y retrouve deux thèmes principaux soit la visualisation des configurations et la visualisation des statistiques.

Pour l'un ou l'autre des thèmes, un usager doit pouvoir visualiser, pour un équipement donné, les interfaces sur lesquelles une configuration de QdS est appliquée. De plus, il doit également pouvoir visualiser, pour un équipement et une interface donnée, les politiques de services appliquées sur cette interface.

3.2.2.1 Visualisation des configurations de QdS

Un usager peut visualiser, pour tout type d'équipement, les configurations des mécanismes de QdS appliquées sur une interface et un équipement donné. Pour une politique de service donnée, l'usager doit être en mesure de visualiser le nom de chaque classe présente dans cette politique. De plus, pour une classe donnée, l'usager peut visualiser si cette classe fait appel à une autre politique de service dite *enfant* (voir le *hierarchical policy* dans [17]). Finalement, pour une classe donnée, l'usager peut visualiser les informations pertinentes des configurations de QdS tel que précisées dans le Tableau VIII.

Tableau VIII

Configurations de QdS pouvant être visualisées par l'usager

Type de configuration	Description
Classification	L'usager peut visualiser les critères de classification de cette classe.
Marquage	L'usager peut connaître le type de marque ainsi que la valeur appliquée.
Ordonnancement	L'usager peut connaître le débit de l'interface alloué à cette classe ainsi que le type de file d'attente utilisé.
Lissage	L'usager peut connaître les valeurs des principaux paramètres de lissage tel CIR, Bc et Be.
Policing	L'usager peut connaître les principaux paramètres de configuration tel CIR, Bc, Be ainsi que les actions prises pour le trafic conforme, en excès ou violant les paramètres du contrat de service. De plus, il peut connaître le type et les valeurs de remarquage.
Évitement de congestion	L'usager peut connaître les valeurs DSCP ou *Precedence* pour lesquelles le WRED a été configuré. De plus, il peut connaître les principaux paramètres tel les seuils minimaux et maximaux ainsi que la probabilité de rejet maximal et ce, pour chacune des valeurs DSCP ou *Precedence* configurée.

3.2.2.2 Visualisation des statistiques des mécanismes de QdS

Un usager peut visualiser les statistiques des divers mécanismes de QdS d'une politique de service appliquée sur une interface et un équipement donné. Ainsi pour chacune des classes de la politique de service, l'usager peut visualiser des statistiques sur tous les mécanismes de QdS activés dans cette classe.

Les statistiques pouvant être prélevées pour chacun des mécanismes sont résumées dans le Tableau IX. Les statistiques peuvent être présentées à l'usager dans deux principaux formats. Dans un premier temps, l'usager peut utiliser le mode *surveillance* pour pouvoir voir uniquement les dernières valeurs récoltées. Dans un deuxième temps, l'usager peut utiliser le mode *historique* pour pouvoir voir, en totalité ou en partie, les données récoltées depuis le début de la récolte d'information.

Concernant l'interface graphique, l'usager peut sélectionner la ou les courbes qu'il désire visualiser sur un même graphe. Par ailleurs, l'usager peut visualiser plusieurs graphes à la fois et les disposer comme bon lui semble dans son écran (par le biais de fenêtres *Pop Up* par exemple ou autre).

Tableau IX

Aspects de la QdS pouvant être surveillés par l'usager

Mécanismes ou aspects à surveiller	**Descriptions**
Classification	Pour chaque classe, l'usager peut visualiser le débit entrant dans la classe.
File d'attente	Pour chaque classe, l'usager peut visualiser : a. Le nombre de paquets présents dans la file d'attente. b. Le nombre de paquets rejetés.
Ordonnancement	Pour chaque classe, l'usager peut visualiser : a. Le débit sortant de la classe. b. Le débit des informations rejetées.
Marquage	Pour chaque classe qui effectue du marquage et pour chaque type et valeur de marquage, l'usager peut visualiser le nombre de paquets marqués.

Tableau IX(suite)

Mécanismes ou aspects à surveiller	**Descriptions**
Lissage	Pour chaque classe où le lissage est configuré, l'usager peut visualiser : a. Si le lissage est en action ou non. b. Le nombre de paquets dans le tampon utilisé pour le lissage. c. Le nombre de paquets qui ont subi un délai dû à ce mécanisme. d. Le nombre de paquets rejetés dû à un débordement du tampon utilisé pour le lissage.
Policing	Pour chaque classe où le *policing* est activé : a. L'usager peut voir le débit du trafic conforme aux spécifications du contrat. b. L'usager peut voir le débit du trafic en excès. c. L'usager peut voir le débit du trafic violant le contrat de service.
Évitement de congestion	Pour chaque valeur DSCP ou *Precedence*, l'usager peut visualiser : a. Le nombre de paquets rejetés dû au mécanisme d'évitement de congestion. b. Le nombre de paquets rejetés dû au débordement de la file d'attente de cette classe.

3.2.2.3 Configurations semi-automatiques

Afin d'éviter à un usager d'avoir à connaître toutes les commandes relatives à la QdS, cette fonctionnalité permet à un usager de pouvoir configurer des politiques de service par le biais d'une interface ergonomique.

Cette interface permet entre autre à l'usager de pouvoir sélectionner l'équipement sur lequel il désire appliquer cette politique de service. Par la suite, l'usager peut sélectionner l'interface de l'équipement sur lequel il désire appliquer cette politique de service. Il peut définir un nom à la politique de service puis à chaque classe de cette politique. Ensuite, il doit définir tous les critères de classification de la classe afin de spécifier les types de trafic qui seront traités par celle-ci. Finalement, pour chaque classe, l'usager peut définir les mécanismes de QdS ainsi que leurs paramètres.

Pour terminer, l'usager peut enregistrer sa configuration dans le but de l'appliquer sur plusieurs équipements. Lorsqu'il est près, l'usager peut configurer le ou les équipements concernés avec la politique de service qu'il vient de créer.

3.2.2.4 Génération d'alertes

Cette fonctionnalité permet à l'usager d'obtenir des notifications lorsque certains événements relatifs à la QdS se produisent. Par conséquent, cette fonctionnalité permet à l'usager de définir des seuils pour tous les aspects énumérés dans le Tableau IX. C'est-à-dire qu'une fois ce seuil dépassé, une notification sera envoyée à l'usager.

Une autre fonctionnalité forte intéressante, pour un administrateur de réseau, est d'obtenir une notification lorsqu'une configuration de QdS a été modifiée. Ainsi il pourra rester informé des modifications apportées aux configurations.

3.2.2.5 Génération de rapports de performance

Cette fonctionnalité permet à l'usager de définir ses propres rapports de performance. De façon plus précise, le système permet à l'usager d'établir d'abord le format dans lequel le rapport sera envoyé (.html, .pdf, .doc). Par la suite, il peut préciser la fréquence de génération des rapports (journaliers, hebdomadaires, mensuels ou annuels) ainsi que la ou les adresses de messagerie auxquelles les rapports seront envoyés.

Une fois ces informations générales complétées, l'usager peut définir le formatage de son rapport. Par exemple, il peut indiquer le nombre de graphiques à inclure dans le rapport et définir un titre pour chacun d'eux. Par la suite, il doit détailler les statistiques qu'il désire visualiser au sein de chacun des graphes. Finalement il peut dresser la liste des alertes qu'il désire voir apparaître dans le rapport.

Un tel rapport le tiendra au courant de l'usage des ressources réseau tout en lui permettant de voir si les configurations de QdS sont adéquate ou nécessitent quelques modifications. Finalement, il sera informé des événements importants susceptibles de brimer la QdS.

3.2.3 Les spécifications non fonctionnelles

Ces spécifications définissent les aspects non fonctionnels du système. C'est-à-dire des aspects qui ne constituent pas des fonctionnalités du système mais qui sont très importants lors du développement de celui-ci. Veuillez noter que certains des éléments de spécifications ont été tirés de [33].

3.2.3.1 Utilisation

Cette section regroupe les principales spécifications relatives aux capacités d'utilisation du système. Elle regroupe des points concernant l'utilisation de l'interface graphique ainsi que l'apprentissage de l'utilisation du système.

D'abord l'interface utilisateur doit permettre de répondre aux spécifications fonctionnelles sans quoi, le système n'aurait pas lieu d'être. De plus, les fonctionnalités offertes doivent être facilement accessibles (quelques clics seulement) et intuitives. Par conséquent, le système sera plus simple d'utilisation et d'apprentissage.

Par ailleurs, le système doit être accessible de n'importe où sur la planète pourvu que l'usager bénéficie d'une connexion à Internet. Ainsi l'administrateur pourra, en tout lieu, pouvoir superviser son réseau.

Par la suite, l'interface utilisateur doit être ergonomique étant donné que le système a pour objectif de faciliter le travail à l'utilisateur et de répondre à ses besoins. De plus, les résultats obtenus à l'écran, par l'utilisateur, doivent être disponibles dans un format imprimable de sorte qu'il puisse conserver les résultats qu'il obtient. Finalement, l'interface doit être disponible dans plusieurs langues afin de faciliter sa capacité d'internationalisation. La langue par défaut du système sera l'anglais afin de pouvoir faire face à une plus grande part de marché.

Pour terminer, une aide et un manuel d'utilisateur doivent être disponibles en ligne, expliquant les principaux menus et fonctionnalités du système. De plus, une aide à la saisie doit être disponible (sous forme de compagnon par exemple) afin d'assister l'usager dans les diverses tâches qu'il exécute. Puis, une liste *Frequently Asked Questions* (FAQ) et un support technique doivent être disponibles en ligne afin de pouvoir répondre efficacement aux éventuels problèmes techniques.

3.2.3.2 Fiabilité

Cette section présente les exigences non fonctionnelles qui concernent la fiabilité du système. Il s'agit d'exigences générales qui concernent l'ensemble du système.

D'abord la sécurité est une exigence primordiale afin que seuls les usagers autorisés puissent bénéficier du système. De plus, les données doivent être protégées de sorte qu'elles ne puissent être interceptées, altérées ou détruites par un utilisateur non autorisé.

Par ailleurs, les pannes ne doivent survenir que très rarement et le temps de récupération doit être minime. Par exemple, afin de préserver un bon niveau de fiabilité, un maximum d'une panne par année est permis dont le temps de récupération ne peut s'élever à plus de 5 minutes. De plus, aucunes données ne doivent être perdues en cas de panne et le système doit être en mesure de poursuivre ses opérations là où elles ont été interrompues au moment de la panne.

Pour terminer, les valeurs, présentées à l'usager, doivent être suffisamment précises pour refléter le comportement des équipements réseau.

3.2.3.3 Performance

Ces spécifications concernent un ensemble d'exigences qui doivent être respectées pour ne pas compromettre les performances générales du système. Elles concernent l'ensemble de l'architecture technique du système et sont résumées dans les paragraphes qui suivent.

Étant donné que ce système est voué à la surveillance de réseau, il s'avère impératif qu'il soit disponible en tout temps afin qu'un administrateur puisse réagir rapidement face à un éventuel problème technique.

Afin que l'utilisateur puisse être au courant si le système de surveillance est en défaut, chaque traitement lancé par l'utilisateur doit être validé lorsqu'il est terminé. De plus, l'utilisateur doit être informé de l'état de progression du traitement en question. Finalement, il doit pouvoir annuler tout traitement et ce, en tout temps. Ainsi il pourra annuler des traitements lancés par erreur ou jugés trop long.

Par ailleurs, les échanges SNMP doivent être priorisés dans le réseau de sorte que les paquets subissent peu de latence et de perte. De plus, la latence entre le serveur Web et le client doit être le seul facteur pouvant faire augmenter significativement le temps de réponse général du système.

Pour terminer, le nombre maximal d'équipements pouvant être surveillés en concurrence doit être évalué. Ainsi les performances d'un système QMA ne pourront être affectées dû à une surcharge de ce dernier. Similairement, le nombre maximal de sessions concurrentes avec le serveur Web doit être évalué.

3.2.3.4 Sécurité

Étant donné que les informations récoltées par le système QMA sont de nature confidentielle, il s'avère très important pour les entreprises qui utilisent QMA que l'aspect de sécurité soit considéré par ce dernier. Par conséquent, tout échange d'information entre les diverses composantes de l'architecture doit être sécurisé de sorte que l'information en transit ne puisse être interceptée et utilisée par des personnes non autorisées. Il est donc recommandé d'utiliser le protocole HTTPS entre le fureteur de l'utilisateur et le service Web afin de sécuriser l'échange d'informations entre ceux-ci.

Par ailleurs, le protocole SNMPv3 doit être utilisé pour interroger les équipements réseau de façon sécuritaire. Tous les équipements réseau doivent avoir un compte SNMPv3 configuré, afin de permettre au système de surveillance d'interroger l'agent

SNMP. Par conséquent, une base de données doit être créée pour conserver la liste des noms d'usager et mots de passe SNMPv3. Cette base de données doit régulièrement être tenue à jour afin que tous les équipements soient accessibles en tout temps.

Finalement, le protocole SSH doit être utilisé entre les modules et les équipements réseau pour que les modules puissent envoyer des commandes aux équipements de façon sécurisée.

3.2.3.5 Capacité d'intégration et de maintenance

« Il s'agit de toutes les contraintes liées à la facilité d'accueil et de maintenance du nouveau système dans son environnement »[33].

D'abord le système doit être facile à installer et à paramétrer. Par exemple, un client qui désire installer le système QMA dans son réseau, ne doit exécuter que quelques manipulations physiques (exemple : connecter le système dans le réseau) et quelques paramétrages (exemple : lui spécifier une adresse IP pour qu'il puisse avoir une connectivité avec son équipement). De plus, les manipulations physiques et le paramétrage doivent être clairement détaillés dans un document d'installation et de paramétrage fournit avec le système. Par la suite, le système doit être facile à distribuer aux utilisateurs. Dans le cas du système QMA, il doit être accessible par une interface Web.

Par ailleurs, dans le but de faciliter l'intégration du système aux équipes de développement du système QMA, un document de conception doit être réalisé et régulièrement tenu à jour. De plus, le système doit être facile à tester et à valider. Pour terminer, les mises à jour du système doivent être faciles à réaliser et à distribuer. Ce dernier aspect concerne tant les développeurs du système que les administrateurs qui l'ont déployé dans leur réseau.

Pour conclure, le présent chapitre a d'abord décrit la place que pouvait prendre QMA dans le contexte des NGN. Par la suite, les diverses spécifications fonctionnelles et non-fonctionnelles ont été énumérées. Le prochain chapitre partira de cette liste de spécifications afin de décrire une architecture de base dont l'objectif vise à supporter toutes les fonctionnalités citées, en plus de permettre son évolution vers le *Bandwidth Manager* de l'architecture des NGN.

CHAPITRE 4

CONCEPTION

Ce chapitre décrit la conception d'une architecture de base dont le but tend vers les diverses spécifications énumérées dans la section 3.2. Étant donné que toutes les spécifications n'ont pu être implémentées, ce chapitre décrit d'abord les spécifications fonctionnelles qui ont été développées. Par la suite, l'architecture générale de QMA est suite présentée puis, chaques composantes de cette architecture sont décrites avec plus de précision. Ensuite, des diagrammes de séquences sont présentés afin de montrer l'interaction entre les diverses composantes lorsque l'utilisateur exécute des opérations particulières à la QdS. Finalement, comme QMA a été conçu, jusqu'à ce jour, pour interroger des routeurs Cisco uniquement, la dernière section du chapitre présente une approche pouvant être utilisée pour que QMA puisse interragir avec des équipements d'autres équipementiers que Cisco Systems.

4.1 Spécifications mises en oeuvre

Cette section énumère les spécifications fonctionnelles qui ont été implémentées dans QMA. QMA permet de visualiser la topologie du réseau et d'accéder à des fonctionnalités générales à l'ensemble du réseau et à d'autres plus spécifiques à un équipement précis. Par exemple, les options de QdS sont accessibles à l'utilisateur par le biais d'un menu contextuel obtenu lorsque l'utilisateur sélectionne un équipement spécifique.

Concernant les spécifications fonctionnelles, seules les fonctionnalités de visualisation des configurations et des statistiques des mécanismes de QdS ont été réalisées. Par conséquent, les fonctionnalités de configuration semi-automatique, de génération d'alertes et de génération de rapports de performance n'ont pu être réalisées et ont été

laissées pour le développement futur. Finalement, plus de la moitié des spécifications non-fonctionnelles énumérées dans la section 3.2.3 ont été atteintes.

4.2 Architecture générale

Cette section présente l'architecture générale de QMA (voir Figure 14). Elle commence tout d'abord par présenter brièvement le rôle des différentes composantes ainsi que l'interaction entre chacune d'elle. Cependant le choix des plateformes technologiques utilisées n'est pas abordé dans ce document puisqu'il a été expliqué plus en détail dans [34]. Par conséquent, ce mémoire prend pour acquis que la plateforme .Net a été utilisée pour réaliser les diverses composantes de l'architecture. De plus, pour les raisons évoquées dans la section 2.3, le protocole SNMP a été choisi pour l'interaction entre les composantes de l'architecture et les équipements réseaux.

Dans la Figure 14, on peut voir que *ModulesQMA* a deux interfaces : une pour interagir avec le réseau, l'autre avec la base de données. En fait, son rôle principal est d'interroger les équipements réseau à l'aide du protocole SNMP et de remplir la base de données en conséquence. Bien que ces deux fonctions soient ses principales, *ModulesQMA* est également utilisé à d'autres fins comme l'authentification, la gestion de la surveillance des tunnels MPLS, la gestion des Trap SNMP, etc. Il est généralement invoqué par le service Web *WS_QMA* ou par le contrôleur de serveur (*Server Controller*) et ce, de façon sporadique.

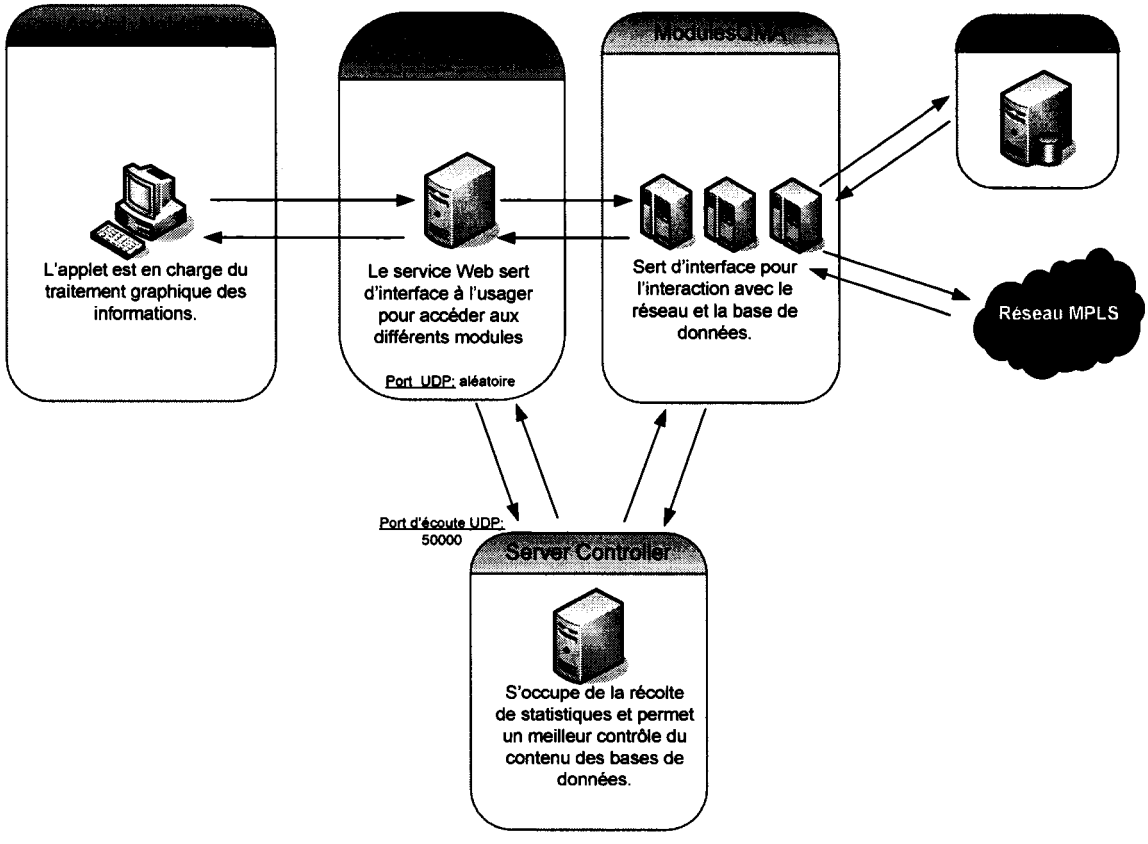


Figure 14 Architecture générale de QMA

La composante *Server Controller* est quant à elle utilisée afin de contrôler la base de données. Cette composante offre une interface utilisateur permettant de vider et/ou remplir une ou des tables de la base de données d'un équipement spécifique. Cette composante interagit avec *ModulesQMA* pour demander d'interroger le réseau et de remplir la base de données en conséquence ou tout simplement pour obtenir certaines informations de cette dernière. En plus de contrôler la base de données de configuration, cette composante est utilisée afin de récolter les statistiques des mécanismes de QdS. Par conséquent, *Server Controller* doit être créé qu'une seule fois et rester actif indéfiniment ou jusqu'à ce qu'un usager le termine. Les fonctionnalités de cette composante peuvent être invoquées soit par l'interface utilisateur ou encore par le biais du service Web.

L'applet constitue l'interface utilisateur typique du client et est en charge du traitement graphique des informations. Il communique avec le service Web (*WS_QMA*) de deux différentes façons. Il peut envoyer des commandes à *ModulesQMA* pour interroger les équipements réseau ou au contrôleur de serveur pour obtenir une configuration ou encore pour démarrer la collecte de statistiques. Ces commandes peuvent être lancées par l'utilisateur par le biais de boutons et/ou de menus. Par ailleurs, il peut communiquer avec le service Web pour obtenir des informations contenues dans la base de données. Ces informations peuvent dans certains cas subir un traitement par l'applet avant d'être affichées à l'utilisateur dans des *comboBox*, *listBox*, *textBox*, graphique ou autres composantes visuelles.

Finalement, le service Web *WS_QMA* est utilisé comme interface entre l'utilisateur de l'applet et le reste du système. Par conséquent, il peut invoquer des méthodes disponibles dans *ModulesQMA* pour interagir avec le réseau et/ou la base de données lorsque requis. Similairement, il communique avec le contrôleur de serveur afin d'obtenir une configuration de QdS ou encore afin d'ajouter et/ou de supprimer une instance d'un objet chargé de récolter les statistiques des mécanismes de QdS.

4.3 Descriptions des composantes de l'architecture générale

Cette section présente les diverses composantes du système QMA, présentées à la Figure 14, mais de façon plus détaillé. Elles seront en fait décrites par le biais de diagrammes de classes. Avant de commencer la lecture de cette section, il est recommandé d'aller voir l'ANNEXE 1 pour connaître les divers OIDs utilisés pour obtenir les configurations et les statistiques de QdS sur les équipements Cisco. Étant donné qu'il n'existe pas actuellement de MIB standard pour l'obtention de telles informations, QMA a donc dû être développé de façon à s'adapter à toutes les MIBs propriétaires. Étant donné que l'équipementier Cisco System est le plus gros vendeur mondial d'équipements IP, le système QMA a d'abord été développé pour supporter la MIB de cet équipementier.

L'ANNEXE 2 présente, quant à elle, la base de données de configurations et de statistiques de QdS. Cette base de données est principalement adaptée à la MIB de Cisco. Cependant, l'intégration d'autres équipementiers sera possible soit en adaptant la base de données afin de la rendre plus générique ou encore en créant une base de données différente dépendamment de l'équipementier utilisé. Pour plus de détails référez vous à la section 4.5. Ainsi l'aspect multi-vendeur de QMA est assuré.

4.3.1 ModulesQMA

Cette section présente les principales classes de *ModulesQMA* dont les classes de base *Snmp* et *Database* ainsi que deux classes de contrôle soit *Control_DB* et *Control_Auth*.

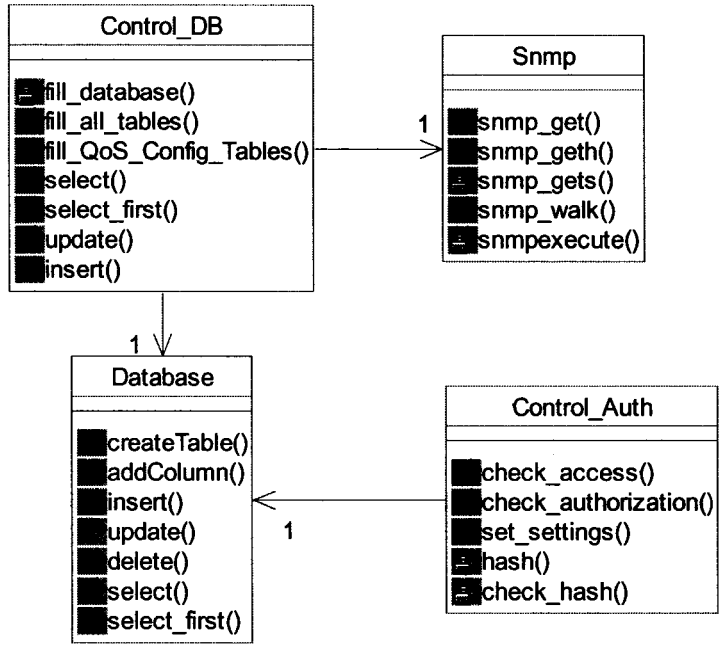


Figure 15 Diagramme de classes de ModulesQMA

4.3.1.1 Classe Snmp

Avant de comprendre le rôle des diverses méthodes, il est essentiel de savoir que la librairie Net-Snmp (http://net-snmp.sourceforge.net/) est utilisée afin d'effectuer les requêtes SNMP vers les équipements. Notez que les fonctions utilisées par cette librairie nous retournent les réponses en format *string* avec la structure qui suit :

OID.Feuille = Type: Valeur

Dans cette structure, *OID* représente l'objet demandé tandis que *Feuille* représente l'instance de l'objet demandé. Par la suite, *Type* correspond au type de la valeur retournée qui est représentée ici par *Valeur*.

La Figure 16 présente le détail de la classe *Snmp* qui ne sert en fait que d'interface d'accès à la librairie Net-Snmp. Par ailleurs, les diverses méthodes de la classe sont décrites ci-dessous.

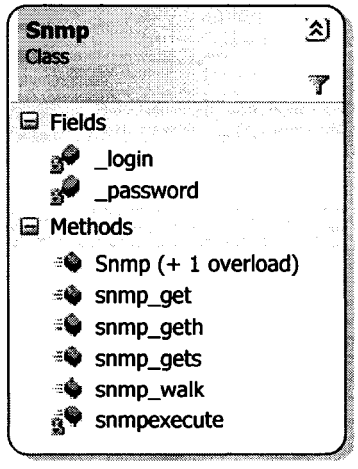


Figure 16 Détail de la classe Snmp de ModulesQMA

Méthode snmp_get

Cette méthode utilise la fonctionnalité GET de SNMP afin d'obtenir la valeur d'un seul *Object ID* (OID). La particularité de cette méthode consiste à ne retourner que la valeur de l'OID. De façon plus précise, elle retourne la valeur située à la fin du *string*, c'est-à-dire après le *Type:*. Dans l'exemple ci-dessous, l'équipement est interrogé pour connaître le nombre d'interfaces présentes dans l'équipement. En utilisant cette méthode, seule la valeur 10 sera retournée.

IF-MIB::ifNumber.0 = INTEGER: 10

Méthode snmp_gets

Cette méthode fonctionne de façon similaire à la précédente. Sa particularité consiste à traiter les OID dont la valeur de retour est de type *STRING*. En considérant l'exemple ci-dessous, cette méthode supprime les guillemets « " » pour ne retourner que la chaîne de caractère *out-ets1-parent*.

enterprises.9.9.166.1.6.1.1.1.1129 = STRING: "out-ets1-parent"

Méthode snmp_geth

Cette méthode fonctionne de façon similaire aux précédentes. Sa particularité consiste à traiter les OID dont la valeur de retour est de type *Hex-STRING*. Cette méthode a été créée spécifiquement pour convertir une chaîne hexadécimale en chaîne de caractère correspondant à une adresse IP. Par exemple, dans le cas où la valeur de retour est « AC 15 FD 0E », cette chaîne est reconvertie en adresse IP « 172.21.253.14 ».

Méthode snmp_walk

Cette méthode utilise la fonctionnalité BULKWALK de SNMP. En considérant que la MIB est structurée en arbre, cette fonctionnalité consiste à spécifier un noeud de l'arbre (ou OID) auquel toute la structure sous-jacente sera retournée. Cette méthode est

pratique dans le cas où un OID a plusieurs feuilles qui s'y rattache. Par exemple, si un usager désire obtenir la liste des noms d'interfaces, il peut demander l'OID *ifDescr* (ou 1.3.6.1.2.1.2.2.1.2) qui retournera ifDescr.ifIndex pour tous les index d'interface (ifIndex). Dans une telle situation, il est donc utile d'obtenir tout le retour de la fonction SNMP-BULKWALK et non seulement la liste des valeurs. Il devient alors plus simple, pour un programmeur, d'associer un index d'interface avec son nom.

Méthode snmp_execute

Cette méthode est une méthode générique qui prend comme paramètre le nom de la fonction SNMP à appeler. Ainsi elle peut être utilisée dans plusieurs situations, comme par exemple par les méthodes snmp_walk, snmp_get, etc. Elle retourne un *string* contenant la totalité de la réponse obtenue.

4.3.1.2 Classe Database

Cette classe est utilisée pour interagir avec la base de données. Elle comprend des méthodes lui permettant de manipuler la base de données de toute sorte de façon. La Figure 17 présente le détail de cette classe tandis que la liste des méthodes ainsi que leurs descriptions sont résumées dans le Tableau X.

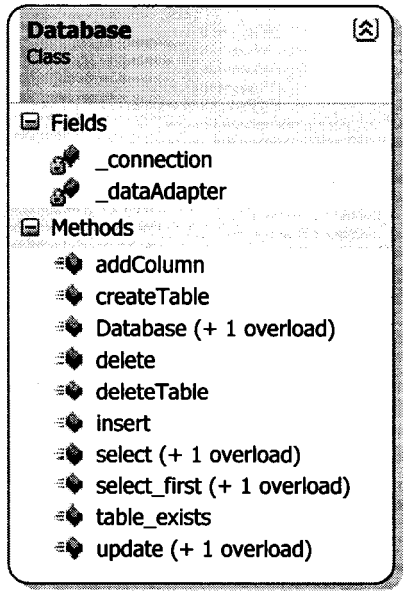


Figure 17 Détail de la classe Database de ModulesQMA

Tableau X

Description des méthodes de la classe Database dans ModulesQMA

Méthodes	**Descriptions**
createTable	Utilisée pour créer une nouvelle table avec les colonnes spécifiées dans les paramètres de la méthode.
deleteTable	Utilisée pour supprimer une table de la base de données.
table_exists	Retourne un booléen indiquant si la table spécifiée existe ou non.
addColumn	Ajoute une ou des colonnes à une table.
insert	Insère une nouvelle entrée dans une table.
update	Modifie une ou plusieurs entrées d'une table selon les conditions spécifiées dans les paramètres de la méthode.
delete	Supprime une ou plusieurs entrées d'une table selon les conditions spécifiées dans les paramètres de la méthode.

Tableau X (suite)

Méthodes	Descriptions
select	Sélectionne une ou plusieurs entrées d'une table suivant les conditions spécifiées dans les paramètres de la méthode. Cette méthode retourne une structure de donnée de type DataSet.
select_first	Retourne la première occurrence des critères de sélection spécifiés dans les paramètres de la méthode.

4.3.1.3 Control_DB

Cette classe est utilisée pour remplir la base de données selon les informations récoltées des équipements. Elle utilise principalement la classe *Snmp* pour interroger les équipements, puis la classe *Database* pour interagir avec la base de données. Elle comporte une foule de méthodes pour remplir ou vider les tables de façon individuelle ou en groupe. Toutes les méthodes (58 au total) n'ont pas été insérées dans le diagramme de classe dans le but de l'alléger. Il est cependant important de noter que la table Router doit être remplie au préalable avant d'utiliser les méthodes de remplissage des tables relatives à la QdS. Certaines méthodes ont été définies pour remplir et vider la table Router (*fill_Router_Table* et *empty_Router_Table*). Elles sont accessibles à un usager via deux boutons tel que présenté dans l'ANNEXE 3. De plus, la classe *Control_DB* possède certaines méthodes utilisées pour manipuler la base de données. Elles possèdent le même nom et syntaxe que les méthodes de la classe *Database* et sont utilisées comme interface à cette dernière classe. En fait, ces méthodes on été créées afin qu'un objet qui possède une instance de *Control_DB* n'ait pas besoin d'instance de *Database* pour obtenir ou modifier des informations de la base de données. La Figure 18 présente le détail de la classe tandis que les principales méthodes de *Control_DB* sont résumées dans le Tableau XI.

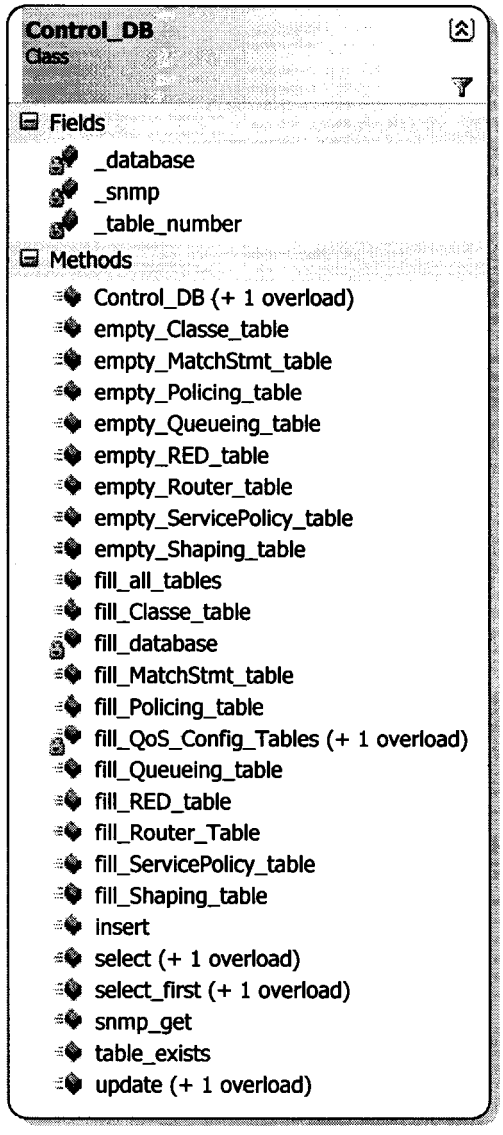


Figure 18 Détail de la classe Control_DB de ModulesQMA

Tableau XI

Description des méthodes de la classe Control_DB dans ModulesQMA

Méthodes	Descriptions
fill_database	Utilisée pour remplir toute les tables relatives à MPLS et à la QdS. Elle appelle entre autre la méthode fill_QoS_Config_Tables.
fill_QoS_Config_Tables	Remplit, pour un ou tous les routeurs, toutes les tables relatives aux configurations de la QdS. Ces tables sont présentées à l'ANNEXE 2 et doivent être remplit dans l'ordre suivant : ServicePolicy, Class, MatchStmt, Queueing, Shaping, Policing puis RED.
select	Appelle la méthode *select* de la classe *Database.*
select_first	Appelle la méthode *select_first* de la classe *Database.*
update	Appelle la méthode *update* de la classe *Database.*
insert	Appelle la méthode *insert* de la classe *Database.*

4.3.1.4 Control_Auth

La Figure 19 présente le détail de la classe *Control_Auth*. Cette classe a deux principaux rôles. Elle est utilisée, dans un premier temps, afin de valider si un usager est autorisé à faire quelque opération que ce soit dans le système ou non. Dans un deuxième temps, lorsque l'usager a été authentifié, cette méthode retourne des informations générales au bon fonctionnement du système. Ces informations sont composées comme suit :

a. Nom du serveur SQL
b. Nom de la base de données de configuration
c. Nom de la base de données de statistiques
d. Nom d'usager et mot de passe de la base de données
e. Nom d'usager et mot de passe SNMP

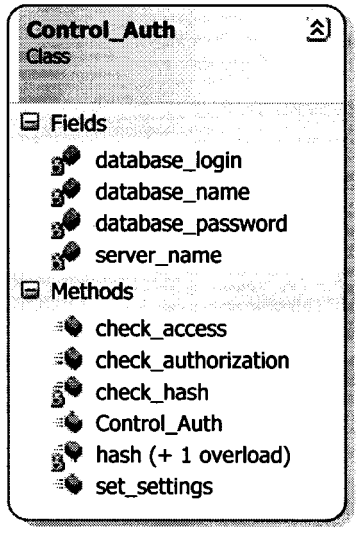


Figure 19 Détail de la classe Control_Auth de ModulesQMA

Le Tableau XII présente une description des méthodes de cette classe.

Tableau XII

Description des méthodes de la classe Control_Auth dans ModulesQMA

Méthodes	Descriptions
check_access	Retourne un booléen indiquant si l'usager est autorisé ou non dans le système.
check_authorization	Une fois que l'usager a été authentifié, cette méthode est utilisée pour retourner les informations générales relatives à la base de données utilisée ainsi qu'aux configurations SNMP. Notez que l'utilisateur peut spécifier les informations qu'il désire recevoir.
set_settings	Configure les informations générales pour un utilisateur spécifique.
hash	Cette méthode prend en paramètre le texte à hacher et retourne une chaîne de caractère contenant le texte haché.

Tableau XII (suite)

Méthodes	Descriptions
check_hash	Cette méthode prend en paramètre le mot de passe non haché (donné par l'usager) ainsi que le mot de passe haché (relevé de la base de données) et retourne un booléen indiquant s'ils correspondent. Cette méthode est principalement utilisée afin de savoir si un usager a entré le bon mot de passe.

4.3.2 Applet

Cette section présente une description de l'implémentation de l'applet, qui est l'interface utilisateur de QMA. La Figure 20 présente un diagramme de classe qui illustre l'interaction entre les diverses classes de l'applet. Dans ce diagramme, seules les classes relatives à la QdS sont affichées. De plus, aucun attribut ni aucune méthode ne sont affichés afin d'alléger la complexité de ce diagramme. Notez cependant que chacune de ces classes sera discutée plus en détail dans les sections qui suivent.

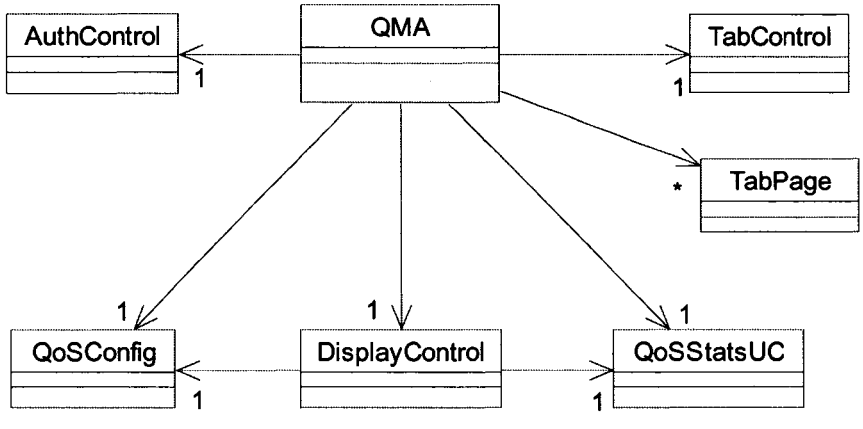


Figure 20 Diagramme de classe de l'applet QMA

Dans la Figure 20, on peut voir que la classe *QMA* est la classe principale. Elle possède les instances des autres classes et possède, du même coup, un certain contrôle sur ces dernières. Dans ce diagramme, les classes *QMA*, *DisplayControl*, *AuthControl*, *QoSConfig* et *QoSStatsUC* sont toutes des contrôles utilisateur (*User Control*). C'est-à-dire qu'elles sont « des blocs de code réutilisables dont le but cible un interface utilisateur »[35]. Tous ces contrôles seront décris dans les lignes qui suivent.

4.3.2.1 Le contrôle utilisateur QMA

Tel que spécifié précédemment, le contrôle utilisateur *QMA* est une classe principale qui détient les instances des autres contrôles utilisateurs. La Figure 21 illustre la composition de cette classe. Notez que seul ce qui a trait à la QdS y a été affiché, le reste étant caché afin de ne pas nuire à la compréhension.

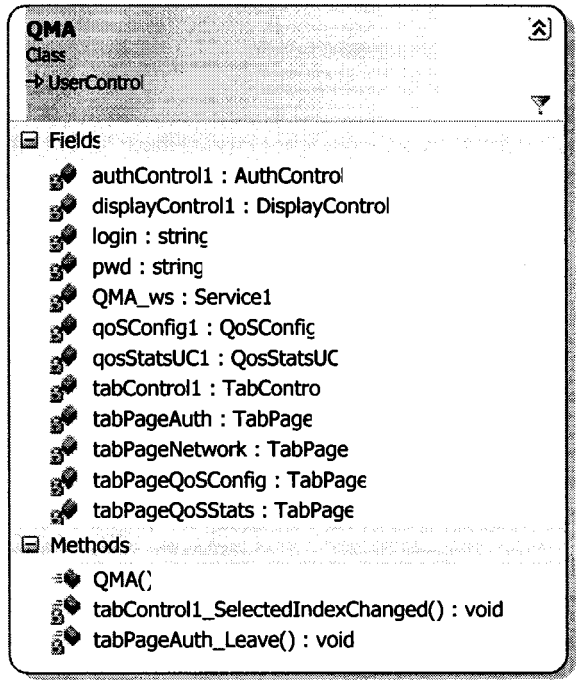


Figure 21 Détails du contrôle utilisateur QMA

On peut voir dans cette figure que le nombre de méthodes et d'attributs est plutôt restreint puisque la majorité du traitement se fait dans les autres contrôles utilisateur. Le contrôle utilisateur QMA est plutôt utilisé afin d'assembler tous les contrôles utilisateurs dans une seule fenêtre. Ce contrôle utilisateur possède la référence au service Web qu'il référera aux autres contrôles utilisateurs qui le nécessite.

Afin de pouvoir afficher plusieurs contrôles utilisateurs dans une seule fenêtre, des onglets ont été réalisés. Par conséquent, *tabControl1* est utilisé afin de gérer les différents onglets tandis que *tabPageAuth*, *tabPageNetwork*, *tabPageQoSConfig* et *tabPageQoSStats* constituent en fait les différents onglets. Il suffit d'insérer les contrôles utilisateur *authControl1*, *displayCotnrol1*, *qoSConfig1* et *qosStatsUC1* dans chaque onglet respectif.

Dans la liste des méthodes on voit bien sûr le constructeur de la classe. La méthode *tabControl1_SelectedIndexChanged* est appelée lorsque l'utilisateur tente de changer d'onglet. Son nom d'utilisateur et son mot de passe sont alors vérifiés afin de s'assurer qu'il s'agit bien d'un utilisateur autorisé. Si ce n'est pas le cas, l'onglet retourne à l'onglet d'authentification. Lorsque l'usager est dans l'onglet d'authentification et clique sur un autre onglet, la méthode *tabPageAuth_Leave* est appelée. Cette méthode sert à valider les paramètres d'authentification entrés par l'usager en appelant une méthode du service Web. Une fois validés, les paramètres d'authentification sont alors configurés dans les autres contrôles utilisateurs qui les nécessitent.

4.3.2.2 Le contrôle utilisateur AuthControl

Le contrôle utilisateur présenté dans cette section est utilisé pour qu'un usager puisse entrer ses paramètres d'authentification. C'est-à-dire qu'il doit entrer son nom d'usager et son mot de passe dans *login_textBox* et *password_textBox* respectivement. S'il désire supprimer le contenu de ces deux *textBox*, l'utilisateur n'a qu'à appuyer sur *button1* afin

que la méthode *button1_Click* soit appelée pour exécuter cette tâche. Quant aux méthodes *get_login_textBox* et *get_password_textbox*, elles sont utilisée par le contrôle utilisateur *QMA* afin qu'il puisse aller chercher le contenu de *login_textBox* et *password_textBox* respectivement (voir Figure 22).

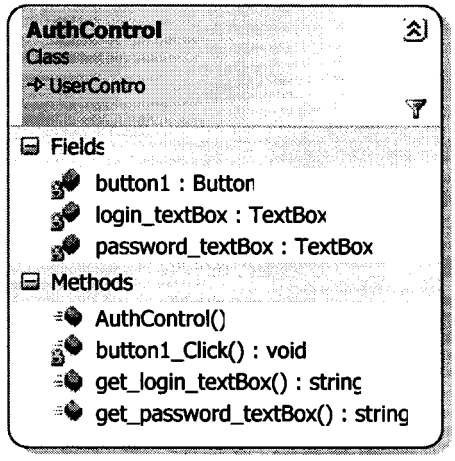


Figure 22 Détails du contrôle utilisateur AuthControl

4.3.2.3 Le contrôle utilisateur DisplayControl

Le contrôle utilisateur *DisplayControl* est celui utilisé pour visualiser les éléments du réseau et pour passer des commandes. Il est cependant important de mentionner que le schéma du réseau ne permet présentement pas d'afficher une grande quantité d'équipements et que du travail reste à faire afin de pouvoir afficher la totalité du réseau d'un fournisseur de service. Veuillez prendre note qu'afin de rendre la figure lisible, la Figure 23 présente une vue sommaire de la classe et que les méthodes et attributs présentés correspondent à ceux liés à la QdS. Dans un même ordre d'idée, les arguments des méthodes ont été omis. Par exemple, les composantes graphiques utilisées pour représenter le réseau (*GraphRouter*) ne sont pas affichés.

Comme *DisplayControl* doit interagir avec le service Web, une référence à celui-ci a été insérée par le contrôle utilisateur *QMA*. Ensuite, celui-ci a également configuré les noms d'usager (*_login*) et mot de passe (*_password*) requis, et ce par le biais des méthodes *set_login* et *set_password*. De plus, *DisplayControl* doit posséder une référence vers certains objets du contrôle utilisateur *QMA* afin de pouvoir interagir sur eux lors de la sélection de certaines options (*qosConfigObject* et *qosStatsObject*), ou encore afin d'imposer un changement d'onglet (*tabControlQMA*, *qosConfigPage* et *qosStatsPage*). Par conséquent la méthode *setQoSObjects* a été créée de sorte que le contrôle utilisateur QMA puisse fournir la référence vers ces instances à *DisplayControl*.

Étant donné que les options de QdS sont accessibles par le biais d'un menu contextuel obtenu en cliquant avec le bouton droit sur un objet *GraphRouter*, l'objet *contextMenuRouter1* a été créé. Ce menu contextuel détient plusieurs items de menu, dont les plus importants pour la QdS sont :

a. menuItemShowPolicy : correspond à l'option de QdS *Show Service Policy*.
b. menuItemRenewQos : correspond à l'option de QdS *ReNew QoS Configurations*.
c. menuItemQosStats : correspond à l'option de QdS *Begin Statistic Collection*.
d. menuItemTelnet : correspond à l'option *Start Telnet*.

Lorsqu'un des *menuItem* est sélectionné, la méthode *menuItemX_Click* correspondante est appelée. Lorsqu'un des *menuItem* correspondant à une option de QdS est sélectionné trois opérations générales à ces options sont réalisées. D'abord le nom du routeur sélectionné est conservé, dans l'attribut *selectedRouter*, pour un usage futur. Par la suite l'attribut *qosSelected* est mis à *true* afin de signaler qu'une option de QdS est sélectionnée puis l'option sélectionnée est conservée temporairement dans l'attribut *qosOptionSelected*.

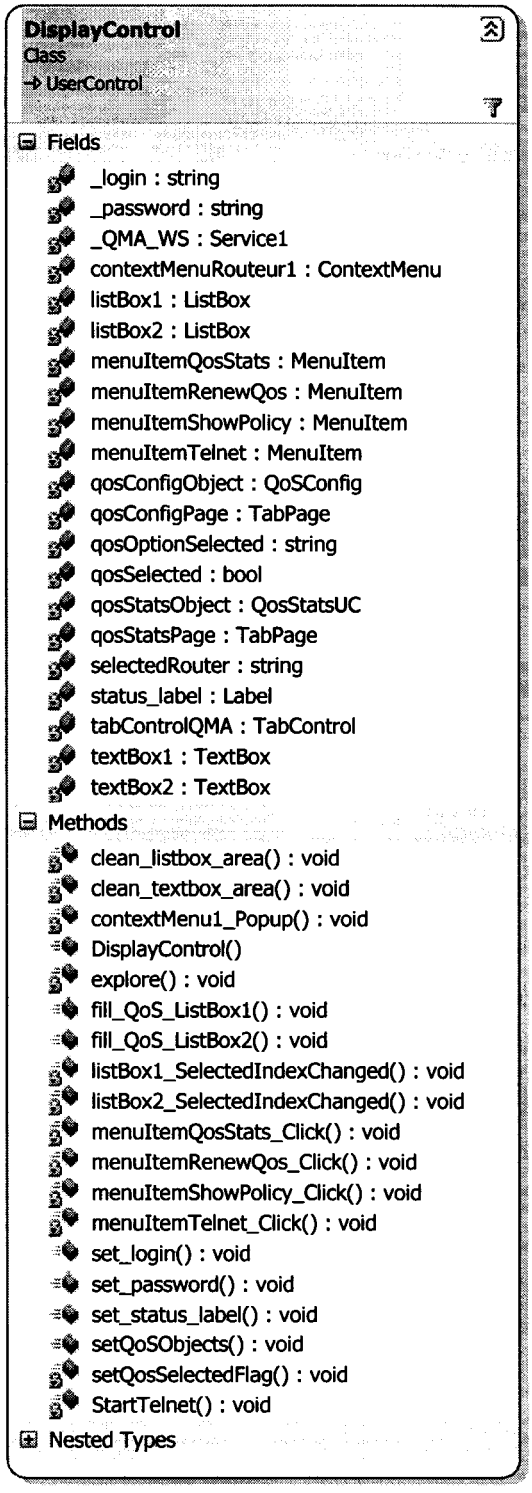


Figure 23 Détail du contrôle utilisateur DisplayControl

Lorsque l'une des options *Show Service Policy* ou *Begin Statistic Collection* est sélectionnée, les méthodes *clean_listbox_area* et *clean_textbox_area* sont appelées afin de vider le contenu des *listBox* et *textBox* respectivement. Par la suite la méthode *fill_QoS_ListBox1* est appelée afin de lister, dans le *listBox1*, les interfaces du routeur sélectionné (*selectedRouter*) sur lesquelles une politique de service est appliquée. Lorsqu'un usager sélectionne alors une interface, la méthode *listBox1_SelectedIndexChanged* est appelée afin qu'elle puisse elle-même lancer la méthode *fill_QoS_ListBox2*. Ainsi le *listBox2* dresse la liste des politiques de service présentes sur l'interface sélectionnée. Une fois que l'usager sélectionne une de ces politiques, la méthode *listBox2_SelectedIndexChanged* est lancée. Cette méthode vérifie d'abord si une option de QdS a été sélectionnée (*qosSelected*). Si tel est le cas, la méthode vérifie quel type d'option a été sélectionné (*qosOptionSelected*) afin d'appeler la méthode correspondante du service Web. Dans le cas de l'option *Show Service Policy*, une réponse, contenant la configuration demandée, est attendue du service Web. *DisplayControl* appelle alors la méthode *addConfig* de l'objet *qosConfigObject* puis change d'onglet afin de diriger l'utilisateur directement à l'onglet *qosConfigPage*. Dans le cas de l'option *Begin Statistic Collection*, seule la méthode *addStats*, de l'objet *qosStatsObject*, a été appelée. Par la suite, l'utilisateur est redirigé vers l'onglet *qosStatsPage*.

Dans le cas où l'utilisateur aurait choisi l'option *ReNew QoS Configurations*, une méthode correspondante est appelée sur le service Web.

Finalement, lorsque l'utilisateur sélectionne l'option *Start Telnet*, le service Web est interrogé afin de retrouver l'adresse IP du routeur sélectionné. Par la suite, la méthode *StartTelnet* est appelée. Cette méthode prend en paramètre l'adresse IP à laquelle la session Telnet doit être établie. Pour terminer, l'application Telnet est lancée à l'adresse IP spécifiée.

4.3.2.4 Le contrôle utilisateur QoSConfig

Le contrôle utilisateur présenté dans cette section (Figure 24), permet de lister et d'afficher les configurations demandées par l'utilisateur. L'un des principaux attributs de cette classe est *configList* qui est en fait une liste triée contenant la liste des configurations, en format texte, que l'utilisateur a demandé de visualiser. Ces configurations sont indexées par une chaîne de caractères identifiant, de façon unique, une politique de service. En fait, la chaîne de caractère est structurée comme suit : *NomRouteur-NomInterface-Direction-NomDeLaPolitique*. Le *comboBox1* dresse la liste des index de la liste triée. Lorsque l'utilisateur sélectionne un de ces éléments de la liste, la méthode *comboBox1_SelectedIndexChanged* est appelée, laquelle retrouve la configuration correspondante de la liste et l'affiche dans le *textBox1*.

L'utilisateur peut supprimer les configurations de la liste à sa guise en utilisant le bouton *buttonRemove* qui appelle la méthode *buttonRemove_Click* qui supprime alors l'entrée correspondante de la liste triée et du *comboBox1*.

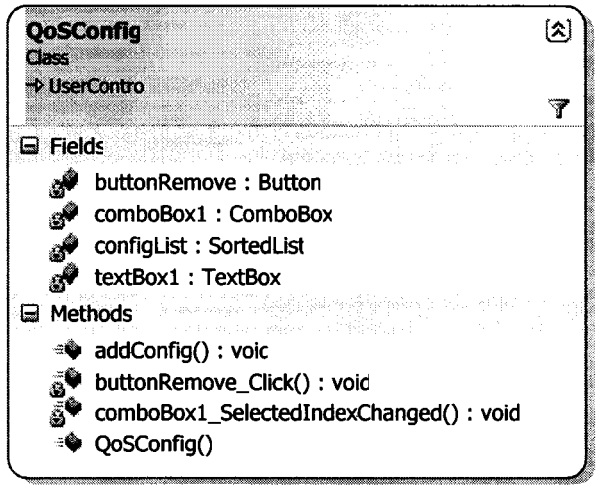


Figure 24 Détail du contrôle utilisateur QoSConfig

4.3.2.5 Le contrôle utilisateur QoSStatsUC

Le principal but de ce contrôle utilisateur est d'afficher les statistiques de QdS à l'usager. Le détail de sa classe est présenté à la Figure 25 à l'exception de certains *Label* qui ont été omis afin d'alléger la figure sans toutefois compromettre la compréhension de la classe. Comme ce contrôle doit interroger la base de données régulièrement, la référence au service Web *QMA_ws* a été insérée et configurée par le contrôle utilisateur *QMA*. Par ailleurs, afin que le service Web puisse authentifier l'utilisateur, le contrôle utilisateur *QoSStatsUC* doit avoir connaissance du nom d'usager (*login*) et du mot de passe (*password*) qui sont tous deux configurés par le contrôle utilisateur *QMA* via les méthodes *setLogin* et *setPassword*.

Tel que vu dans la description du contrôle utilisateur *DisplayControl*, la méthode *AddStats* est utilisée pour ajouter une politique de service à la liste des politiques surveillées si elle n'y est pas déjà. Cette méthode appelle la méthode *addQoSStats* du service Web.

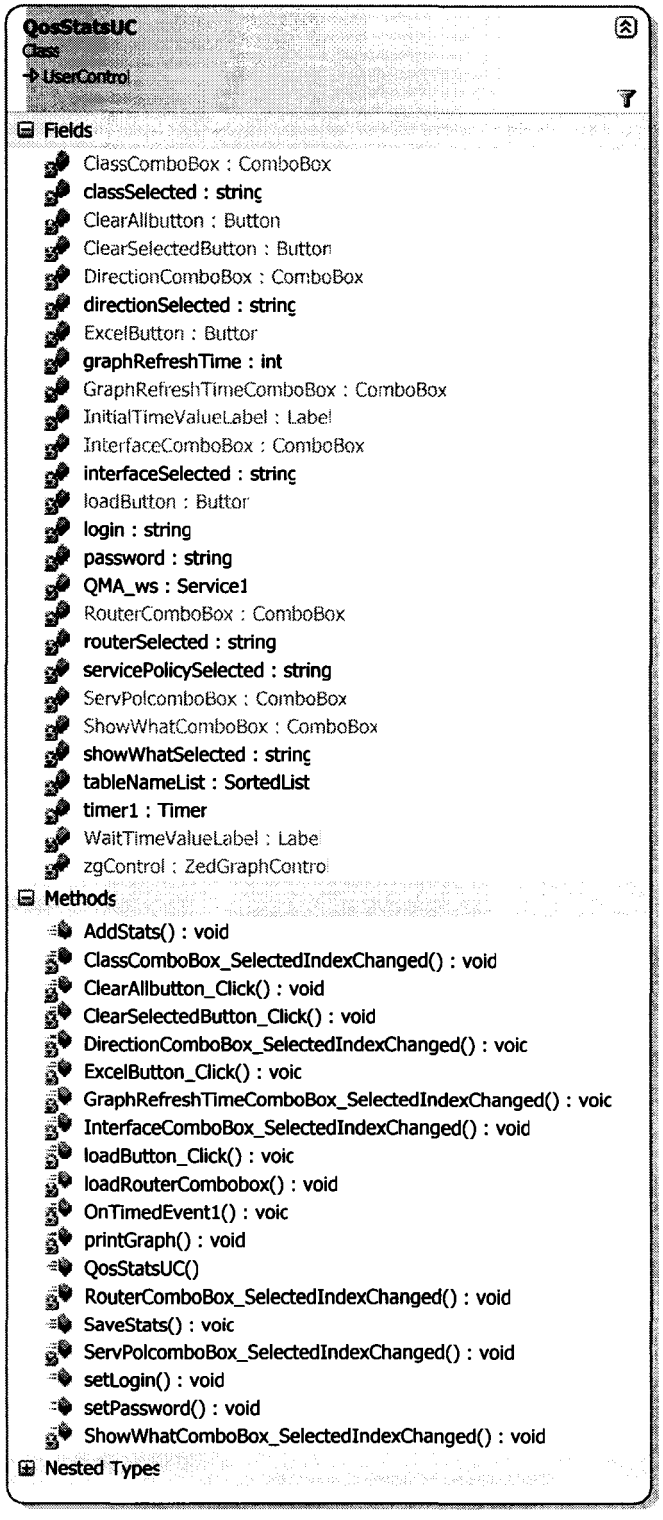


Figure 25 Détail du contrôle utilisateur QosStatsUC

Les *comboBox*

Plusieurs *comboBox* ont été insérés afin de permettre à l'utilisateur de sélectionner les statistiques à visualiser. C'est *comboBox* sont listés et décrits brièvement ci-dessous.

a. routerComboBox :
   - Permet la sélection de l'équipement sur lequel la politique désirée est appliquée.

b. InterfaceComboBox :
   - Permet la sélection de l'interface de l'équipement sur laquelle la politique de service désirée est appliquée.

c. DirectionComboBox :
   - Permet de sélectionner la direction de l'interface et du routeur sélectionné sur laquelle la politique de service désirée est appliquée.

d. ServPolcomboBox :
   - Permet de sélectionner la politique de service désirée.

e. ClassComboBox :
   - Permet de sélectionner la classe de la politique de service dont l'usager désire voir des statistiques. Notez que le nom de la classe représenté dans ce *comboBox* a été structuré en insérant le nom de la politique et le nom de la classe, tous deux séparés d'un « _ ». Cette structure a été adoptée afin de différencier les classe pouvant porter le même nom dans une politique parent et enfant.

f. ShowWhatComboBox :
   - Permet de sélectionner les statistiques de la classe que l'usager désire visualiser.

Les objets des *comboBox* doivent être sélectionnés dans la séquence suivante :

Router→Interface→Direction→Politique de service→Classe→ShowWhat

Lorsqu'un un objet est sélectionné dans l'une ou l'autre des listes précédentes, la méthode *xyzComboBox_SelectedIndexChanged* correspondante est appelée. Par conséquent, lorsqu'un objet est sélectionné dans un *comboBox*, la méthode *xyzComboBox_SelectedIndexChanged* correspondante est utilisée pour remplir le *comboBox* suivant de la séquence à l'exception de *ShowWhatComboBox*. En effet lorsqu'un objet de la liste de ce *comboBox* est sélectionné, la méthode *printGraph* est appelée afin d'afficher les statistiques correspondantes.

Étant donné qu'une table est créée dans la base de données pour chacune des classes, la liste triée *tableNameList* conserve la liste des noms de table indexé avec le nom de la politique de service et le nom de la classe. Par conséquent, cette liste est remplie lorsque l'usager sélectionne une politique de service dans *ServPolcomboBox*.

De plus, des attributs ont été créés afin de conserver la sélection réalisée par l'utilisateur, soit les attributs :

a. routerSelected : Conserve l'identification du routeur sélectionné.
b. interfaceSelected : Conserve l'identification de l'interface sélectionnée.
c. directionSelected : Conserve l'identification de la direction sélectionnée.
d. servicePolicySelected : Conserve l'identification de la politique de service sélectionnée.
e. classSelected : Conserve l'identification de la classe sélectionnée.
f. showWhatSelected : Conserve l'identification de la statistique sélectionnée.

L'affichage graphique

Une fois que l'usager a sélectionné la statistique qu'il désire visualiser, la méthode *printGraph* est appelée. Cette méthode utilise le nom de la table, prise dans *tableNameList*, afin de savoir quelle table interroger. Par la suite elle recherche, dans la table *StatsTablesInfo* (voir ANNEXE 2), les divers noms de colonnes, à aller chercher, qui corresponds aux statistiques demandées. Par la suite elle crée une liste de points pour

chacune de ces colonnes et l'ajoute à l'objet *zgControl* (voir http://zedgraph.org/ pour plus d'informations sur cette librairie) puis défini l'élément de la légende correspondant. Ainsi il est possible d'avoir plusieurs courbes sur un seul graphique. Finalement elle recalcule l'échelle de l'abscisse, rafraîchit le graphique, le rend visible s'il ne l'était pas déjà et démarre *timer1* qui est utilisé pour le rafraîchissement graphique. En fait, à chaque fois que le minuteur *timer1* expire, la méthode *OnTimedEvent1* est lancée. Cette méthode lance à son tour la méthode *printgraph* afin de rafraîchir le graphique. Notez que l'intervalle de rafraîchissement graphique est défini par l'attribut *graphRefreshTime* qui peut être modifié par l'utilisateur à l'aide du *comboBox* *GraphRefreshTimeComboBox*. Si l'usager sélectionne un nouveau temps de rafraîchissement, la méthode *GraphRefreshTimeComboBox_SelectedIndexChanged* est lancée afin de modifier la valeur *graphRefreshTime* et de réinitialiser le minuteur.

Les boutons

Quatre boutons ont été mis à la disposition de l'utilisateur afin de lui permettre un plus grand nombre de fonctionnalités. D'abord le bouton *loadButton* est probablement un des plus utiles puisqu'il permet à un usager d'accéder aux statistiques de toutes les politiques de service surveillées. Lorsque ce bouton est appuyé, la méthode *loadButton_Click* est alors appelée. Cette méthode lance à son tour la méthode *loadRouterComboBox* afin de remplir *RouterComboBox* avec la liste des routeurs du réseau sur lesquels une politique de service est surveillée.

Le bouton *ClearSelectedButton* est quant à lui utilisé pour stopper la collecte d'information de la politique sélectionnée. Lorsque ce bouton est appuyé, la méthode *ClearSelectedButton_Click* est lancée afin de supprimer toutes les entrées de la base de données correspondant à cette politique. Elle fait appel à la méthode *clearQoSStats* du service Web.

Le bouton *ClearAllButton* est quant à lui utilisé pour stopper la collecte de statistiques de toutes les politiques de service surveillées. Elle vide également la base de données des statistiques de ces politiques.

Finalement le bouton *ExcelButton* est utilisé pour sauvegarder toutes les statistiques dans un fichier Microsoft Excel. Lorsque ce bouton est appuyé, la méthode *ExcelButton_Click* est lancée. Cette méthode appelle à son tour la méthode *SaveStats* qui se charge du formatage et de la sauvegarde des données. Cette méthode enregistre toutes les tables listées dans *tableNameList* une à la fois. Pour chacune de ces tables, les colonnes sont ajoutées une à une.

4.3.3 Service Web

Tel que mentionné précédemment, cette composante a été créée afin de servir d'interface entre l'applet du client et le reste du système. De plus, elle offre une connexion sécurisée (avec HTTPS) afin de protéger les données échangées. La Figure 26 présente le diagramme de classe du service Web. Bien qu'on ne voit qu'une seule classe, une classe (Global.asax) supplémentaire est ajoutée par défaut, afin de gérer les connexions TCP. Quant à la classe *QMA_WebService.asmx*, les diverses méthodes présentées dans la figure peuvent se séparer en trois catégories soit l'authentification et vérification d'accès, l'interaction avec la base de données puis l'interaction avec le contrôleur du serveur. De plus, certaines classes de *ModulesQMA* sont parfois instanciés directement dans les méthodes de *QMA_WebService*.

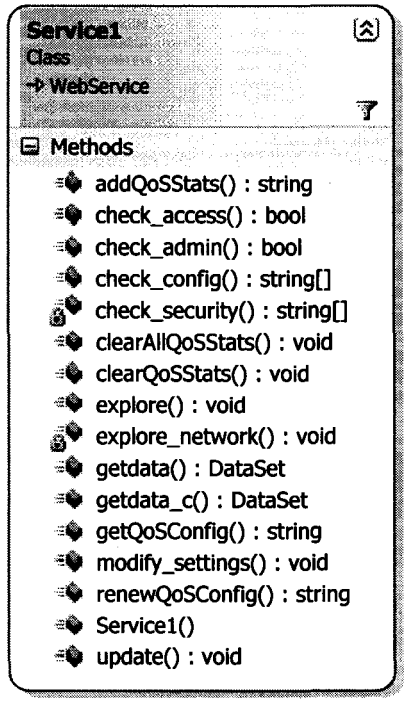


Figure 26 Diagramme de classe du service Web

4.3.3.1 Authentification et obtention des informations générales

Cette catégorie englobe les cinq premières méthodes présentées dans la Figure 26 soit : *check_access*, *check_admin*, *check_security*, *check_config* et *modify_config*. Elles sont entre autre utilisées afin de valider l'authentification d'un utilisateur et d'obtenir ou de modifier ses paramètres généraux. Ces méthodes sont résumées dans le Tableau XIII.

Tableau XIII

Méthodes du service Web utilisée pour l'authentification ou l'obtention/modification des informations générales

Méthodes	Descriptions
check_access	Appelle la méthode *check_access* de la classe *Control_Auth* de *ModulesQMA*.
check_admin	Vérifie si l'utilisateur est de type administrateur ou non et retourne un booléen indiquant s'il en est un ou non.
check_security	Appelle la méthode *check_authorization* de la classe *Control_Auth* de *ModulesQMA* et retourne les paramètres demandés par l'utilisateur.
check_config	Appelle la méthode *check_security* et retourne la liste complète des informations générales.
modify_settings	Appelle la méthode *set_settings* de la classe *Control_Auth* de *ModulesQMA* afin de configurer les informations générales pour un utilisateur particulier.

4.3.3.2 Interaction avec la base de données

Les méthodes de cette catégorie sont utilisées par l'utilisateur de l'applet afin de demander ou de modifier des entrées dans la base de données. Elles sont également utilisées par l'utilisateur de l'applet afin de remplir les bases de données de configuration. Le Tableau XIV résume ces méthodes.

Tableau XIV

Méthodes du service Web utilisées pour l'interaction avec la base de données

Méthodes	Descriptions
getdata_c	Crée une instance de *Control_DB* puis appelle la méthode *select* de cette instance afin de pouvoir retourner à l'utilisateur un *Dataset* correspondant à la table, aux colonnes et aux conditions demandées.
getdata	Appelle la méthode *getdata_c* mais sans spécifier de condition.
update	Crée une instance de *Control_DB* et appelle la méthode *update* de cette instance afin de modifier une entrée de la base de données.
explore	Crée une instance de *Control_DB* et démarre un *thread* qui exécute la méthode *explore_network.*
explore_network	Appelle la méthode *fille_all_tables* de l'instance de *Control_DB* afin de remplir les tables de configurations pour l'ensemble du réseau.
renewQoSConfig	Crée une instance de *Control_DB* et appelle la méthode *fill_QoS_Config_Tables* de cette instance afin de rafraîchir la base de données de configuration de QdS pour un équipement spécifique.

4.3.3.3 Interaction avec le contrôleur du serveur

Les méthodes de cette catégorie sont appelées par l'utilisateur de l'applet et sont exécutées par le contrôleur de serveur. Chacune de ces méthodes établi un *socket* UDP afin d'envoyer les commandes au contrôleur de serveur. Ce *socket* est détruit à la fin de la méthode.

Tableau XV

Méthodes du service Web utilisées pour l'interaction avec le contrôleur de serveur

Méthodes	Descriptions
getQoSConfig	Utilisée pour demander une configuration de QdS et la retourner à l'applet.
addQoSStats	Utilisée pour démarrer la collecte de statistiques pour une politique de service données.
clearQoSStats	Utilisée pour stopper la collecte de statistiques pour une politique de service donnée. De plus toutes les statistiques, précédemment récoltées, sont supprimées.
clearAllQoSStats	Utilisée pour stopper toutes collectes de statistiques ainsi que pour supprimer toutes les statistiques précédemment récoltées.

4.3.4 Contrôleur de serveur

Le contrôleur de serveur est une application qui est constamment active et qui fonctionne sur la même machine physique que le serveur web. Tel qu'illustré à la Figure 14, cette application utilise le protocole UDP pour communiquer avec le service Web. Afin de conserver une architecture sécuritaire, cette application doit obligatoirement fonctionner sur la même machine physique que le service Web afin que les échanges d'information ne puissent être interceptés. Cependant le protocole UDP peut éventuellement être remplacé par un autre, plus sécuritaire, afin de pouvoir séparer physiquement ces deux composantes de l'architecture (serveur Web et contrôleur de serveur).

Cette application est utilisée dans deux buts. Premièrement, elle permet une surveillance des politiques de QdS ainsi qu'une surveillance de certains aspects qui ont trait à MPLS partie traitée dans un autre mémoire [2]. Deuxièmement, elle permet de contrôler certains paramètres généraux.

Étant donné que les différentes instances des modules définis dans *ModulesQMA* sont créées uniquement lorsque le système en a besoin et détruites lorsqu'il en a finies, ces modules ne pouvaient être utilisés pour pouvoir effectuer la récolte de statistiques. Le contrôleur de serveur, qui est une application toujours en fonction, a donc été créé afin de pallier à cette problématique.

Il est important de mentionner que l'interface graphique offerte par le contrôleur de serveur est dédiée à l'administrateur du système QMA et non aux administrateurs réseau qui utilisent QMA pour surveiller le réseau. Ces derniers utiliseront plutôt l'interface offerte par l'applet.

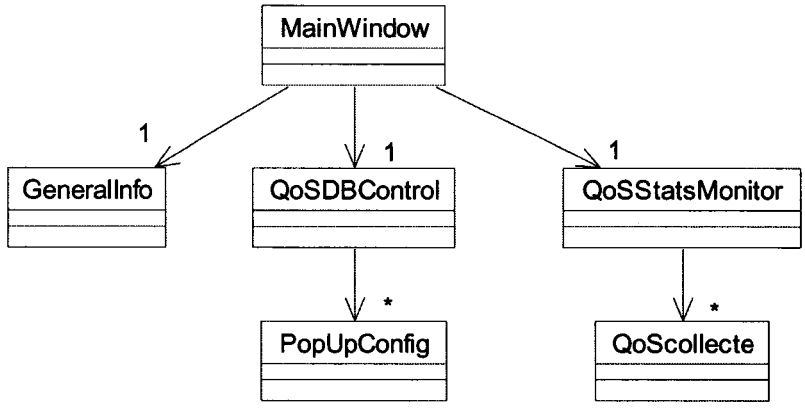


Figure 27 Diagramme de classe du contrôleur du serveur

La Figure 27 présente le diagramme de classe du contrôleur de serveur pour les options qui concernent la QdS uniquement. Cette figure montre que la forme Windows *MainWindow* possède une instance des contrôles utilisateurs *GeneralInfo*, *QoSDBControl* et *QoSStatsMonitor*. Le premier est utilisé pour modifier les paramètres généraux concernant les bases de données et les paramètres SNMP, le second est utilisé pour contrôler la base de données de configuration de la QdS ainsi que pour afficher une politique de service en utilisant la classe *PopUpConfig*, le troisième est utilisé pour gérer

les objets *QoScollecte* qui sont chargés d'effectuer la surveillance d'une politique de service. Les sections qui suivent présenteront plus en détail chacune de ces classes.

4.3.4.1 La forme Windows MainWindow

Cette forme est en quelque sorte la fenêtre principale de l'application. La Figure 28 représente le détail de cette classe. Cette fenêtre possède les instances des différents contrôles utilisateur (*generalInfo1*, *qoSDBControl1* et *qoSStatsMonitor1*). De plus, cette forme gère le changement d'onglet pour passer d'une page à une autre. Ce changement d'onglet est réalisé en utilisant les objets *tabControl1*, *tabPage1*, *tabPage2* et *tabPage3*. La méthode *setGeneralInfo* est utilisée par le contrôle utilisateur *generalInfo1* afin que les informations générales modifiées dans ce contrôle utilisateur soient modifiées également dans les autres contrôles utilisateurs.

Le rôle principal de *MainWindow* consiste à gérer la communication entre cette application et le service Web. Il exécute cette tâche en appelant les méthodes appropriées, du contrôle utilisateur adéquat, lors de la réception d'une commande du service Web. En fait, l'attribut *listener* constitue l'interface de connexion (*socket*) qui écoute sur le port UDP défini par la constante *listenPort* (50000). À la fin de l'initialisation, un *thread* est lancé afin d'exécuter la méthode *StartListener* qui est en charge d'attendre la réception de commandes provenant du service Web et d'appeler les méthodes appropriées.

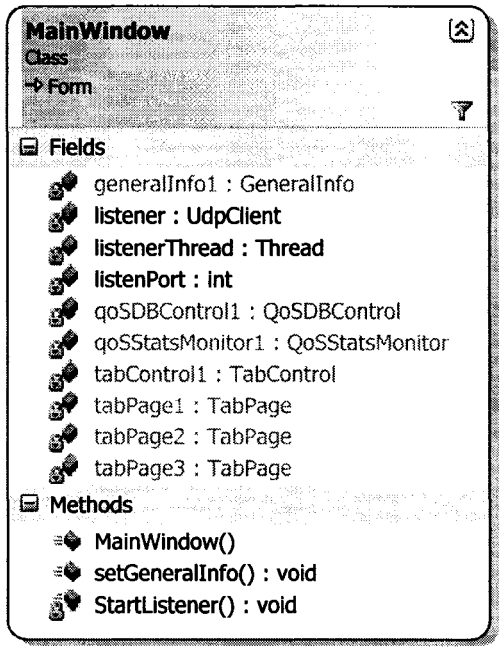


Figure 28 Détail du contrôle utilisateur MainWindow

L'organigramme de la méthode *StartListener* est représenté à la Figure 29. Dans cette figure, on voit qu'une boucle est exécutée indéfiniment. D'abord un message est attendu dont la structure est la suivante :

NomDeLaCommande Paramètre1 Paramètre2 Paramètre3 ...

Cette structure débute par le nom de la commande suivi des paramètres tous séparés par un espace. Le nombre de paramètres transmis dépend de la commande utilisée. Les messages sont de format chaîne de caractères. Une fois qu'un message est reçu, une vérification du type de message est faite puis la méthode adéquate est lancée. Il y a deux principaux types de message, ceux utilisés pour agir sur les statistiques et celui pour agir sur les configurations. Comme il y a des contrôles utilisateurs conçus spécifiquement pour agir sur les statistiques (*qoSStatsMonitor1*) et sur les configurations (*qoSDBControl1*), les méthodes de ces contrôles utilisateurs, correspondants aux commandes reçues dans les messages, seront lancées.

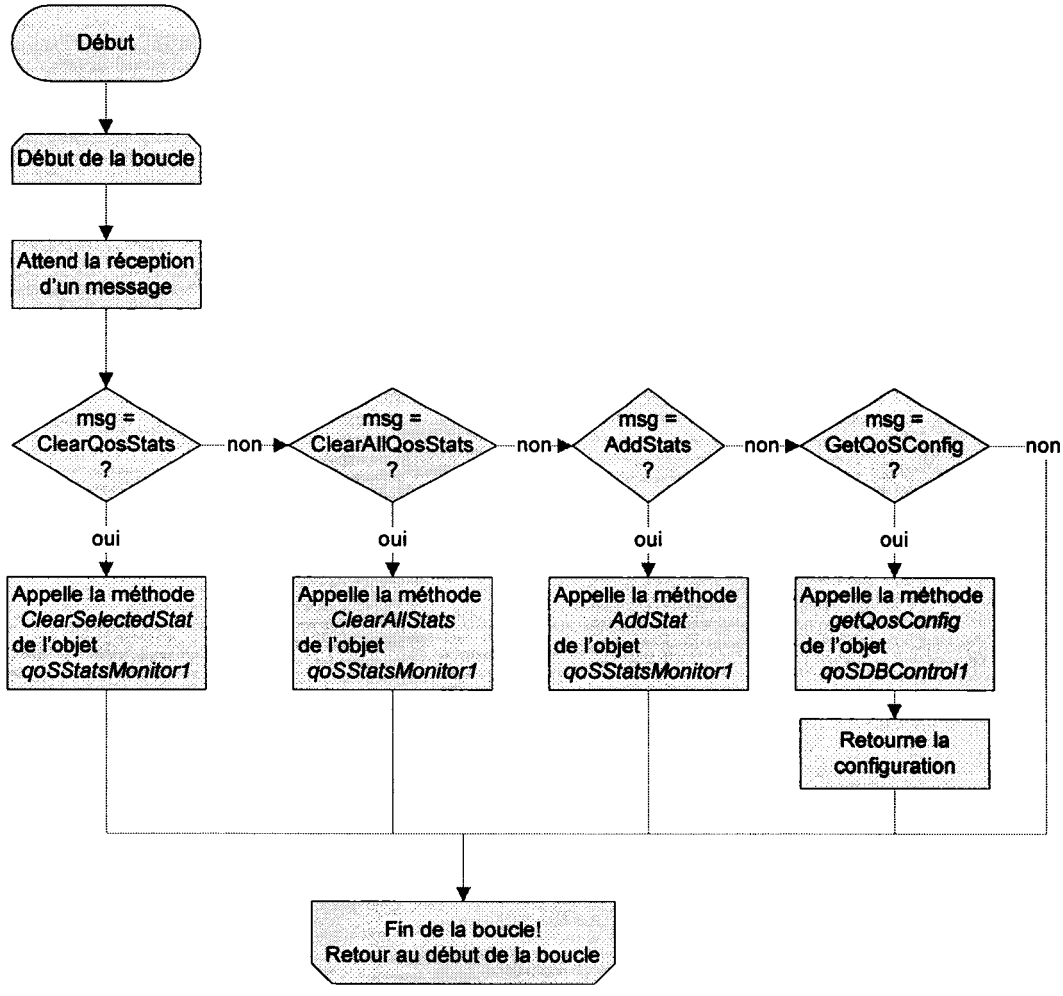


Figure 29 Organigramme de la méthode *StartListener*

En se basant sur les explications ci-dessus, la commande *ClearQosStats* est utilisée pour supprimer les statistiques d'une politique de service précise, identifiée par les divers paramètres. Par conséquent la méthode *ClearSelectedStat* de l'objet *qoSStatsMonitor1* est appelée. Similairement, lorsque la commande *ClearAllQosStats* est reçue, la méthode *ClearAllStats* du même objet est lancée. Cette commande n'inclut aucun paramètre. Lorsque la commande *AddStats* est reçue avec les paramètres qui identifient la politique de façon unique, la méthode *AddStat* du même objet est lancée. Finalement, lorsque la méthode *GetQoSConfig* est reçue, suivie des paramètres qui identifient la politique de

façon unique, la méthode *getQosConfig* de l'objet *qoSDBControl1* est lancée. Cette méthode retourne la configuration, en format chaîne de caractères, qui sera alors retournée à l'entité qui est à l'origine de la commande et ce, en utilisant la même interface de connexion, c'est-à-dire les mêmes adresses IP et ports UDP source et destination.

4.3.4.2 Le contrôle utilisateur GeneralInfo

Ce contrôle utilisateur (voir Figure 30) a été créé afin de centraliser les informations générales utilisées par la majorité des classes de cette application. Si des modifications s'imposent, celles-ci sont réalisées à un seul endroit et sont diffusées à toutes les entités qui les nécessitent. Ces informations générales, codées en dur via des chaînes de caractères, sont présentées à l'utilisateur dans des *textBox*. Le Tableau XVI présente la liste des informations générales ainsi que leurs variables et descriptions associées.

Tableau XVI

Descriptions des informations générales et de leurs variables associées

Nom des chaînes de caractères	**Nom des *textBox***	**Descriptions**
SQLServer	SQLServerTextBox	Nom du serveur SQL.
ConfigDBName	ConfigDBNameTextBox	Nom de la base de données utilisée pour les configurations.
StatsDBName	StatsDBNameTextBox	Nom de la base de données utilisée pour les statistiques.
DBUser	DBLoginTextBox	Nom d'utilisateur utilisé pour l'accès aux bases de données.

Tableau XVI (suite)

Nom des chaînes de caractères	**Nom des *textBox***	**Descriptions**
DBPwd	DBPassTextBox	Mot de passe utilisé pour l'accès aux bases de données.
snmpUser	SNMPLoginTextBox	Nom d'utilisateur utilisé pour les requêtes SNMP.
snmpPwd	SNMPPassTextBox	Mot de passe utilisé pour les requêtes SNMP.

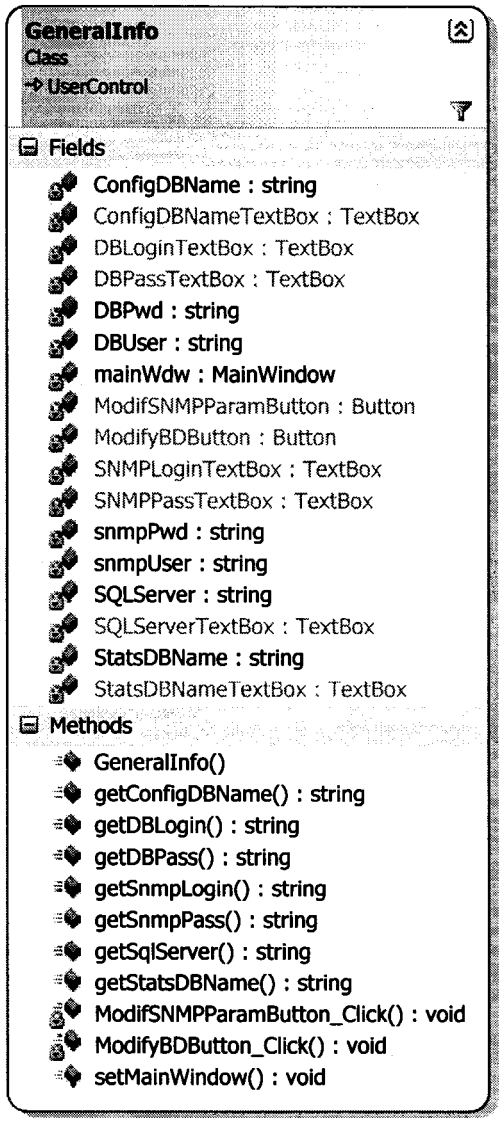


Figure 30 Détail du contrôle utilisateur GeneralInfo du contrôleur de serveur

Comme il peut être vu dans la Figure 30, ce contrôle utilisateur contient une référence vers le contrôle utilisateur *MainWindow*. Cette référence est configurée par le *MainWindow* lui-même, via la méthode *setMainWindow*, une fois que ce dernier a initialisé l'objet *generalInfo1*. Elle est utilisée lorsque l'usager modifie une des informations et clique sur le bouton *ModifyDBButton* ou sur le bouton *ModifSNMPParamButton*. La méthode *setGeneralInfo* du *MainWindow* est alors appelée

afin que les modifications soient diffusées aux autres contrôles utilisateurs. Cette méthode du *MainWindow* utilise d'abord les accesseurs *getSqlServer*, *getConfigDBName*, *getStatsDBName*, *getDBLogin*, *getDBPass*, *getSnmpLogin* et *getSnmpPass* pour accéder aux nouvelles informations puis les diffuse par la suite.

4.3.4.3 Le contrôle utilisateur QoSDBControl

Ce contrôle utilisateur a deux principaux rôles. D'abord il comporte une panoplie de boutons permettant de remplir les tables de configurations de QdS pour un équipement donné. Par ailleurs, il peut être utilisé pour afficher la configuration d'une politique de service pour un équipement donné. La Figure 31 présente le détail de la classe. Certains attributs n'y ont pas été affichés afin de rendre la figure plus lisible. On retrouve, parmi ces attributs, les informations générales qui peuvent être configurées par le *MainWindow* via la méthode *SetGeneralInfo*.

D'abord le *ListBox3* dresse la liste des équipements présents dans le réseau. Il est utilisé afin que l'usager puisse sélectionner un équipement pour lequel il désire effectuer une action. Lorsqu'il sélectionne un équipement, la méthode *listBox3_SelectedIndexChanged* est lancée. Cette méthode lance alors l'autre méthode *Fill_ListBox1_QoSConfig* afin que le *listBox1* dresse la liste des interfaces de cet équipement, pour lesquelles une politique de service a été appliquée. L'usager peut alors soit cliquer sur un des boutons pour effectuer une action sur l'équipement en question, soit cliquer sur une des interfaces affichées dans le *listBox1*. Lorsqu'il sélectionne une interface, la méthode *listBox1_SelectedIndexChanged* est lancée. Cette méthode lance par la suite l'autre méthode *Fill_ListBox2_ServicePolicy* afin d'afficher les politiques de services présentes sur l'interface sélectionnée. Lorsque l'usager sélectionne une des politiques affichées dans le *listBox2*, la méthode *listBox2_Selected_IndexChanged* est lancée. Cette méthode lance alors la méthode *getQosConfig* qui retourne la configuration. Cette méthode est décrite par l'organigramme présenté et décrit à la

Figure 65 de l'ANNEXE 2. Par la suite, un objet *PopUpConfig* est créé afin d'afficher la configuration dans une fenêtre de type *pop up*.

Les boutons

En tout temps, l'usager peut cliquer sur le bouton *Clear_All* afin que la méthode *Clear_All_Click* soit lancée. Cette méthode est utilisée afin de vider toutes les tables de configuration de QdS de tous les équipements listés dans le *listBox3*. Les autres boutons nécessitent la sélection d'un équipement spécifique dans le *listBox3*. Leur liste et descriptions sont présentées dans les points qui suivent. Notez qu'avant de remplir, pour un équipement donné, les tables relatives aux configurations de la QdS, la table *Router* doit impérativement être remplie. Cette table peut être remplie en cliquant sur le bouton *Fill_Router_Table.*

a. Le bouton *Fill_Router_Table* lance la méthode *Fill_Router_Table_Click.*
   - Remplit la table *Router* du routeur sélectionné.

b. Le bouton *Empty_Router_Table* lance la méthode *Empty_Router_Table_Click.*
   - Vide la table *Router* du routeur sélectionné.

c. Le bouton *Fill_All* lance la méthode *Fill_All_Click.*
   - Remplit toutes les tables de QdS du routeur sélectionné en appelant la méthode *Fill_All_QoS_Table.* Suit la séquence de remplissage suivante :

ServicePolicy → *Class* → *MatchStmt* → *Queueing* → *Shaping* → *Policing* → *RED*

d. Le bouton *Remove_All* lance la méthode *Remove_All_Click.*
   - Vide toutes les tables de configuration de QdS du routeur sélectionné. Suit la séquence inverse de celle citée dans le point précédent.

e. Le bouton *Fill_ServicePolicyTable* lance *Fill_ServicePolicyTable_Click.*
   - Remplit la table *ServicePolicy* du routeur sélectionné.

f. Le bouton *Empty_ServicePolicyTable* lance *Empty_ServicePolicyTable_Click.*
   - Vide la table *ServicePolicy* du routeur sélectionné.

g. Le bouton *Fill_ClasseTable* lance la méthode *Fill_ClasseTable_Click.*

- Remplit la table *Class* du routeur sélectionné.

h. Le bouton *Empty_ClasseTable* lance la méthode *Empty_ClasseTable_Click.*
   - Vide la table *Class* du routeur sélectionné.

i. Le bouton *Fill_MatchStmt_Table* lance la méthode *Fill_MatchStmt_Table_Click.*
   - Remplit la table *MatchStmt* du routeur sélectionné.

j. Le bouton *Empty_MatchStmt_Table* lance *Empty_MatchStmt_Table_Click.*
   - Vide la table *MatchStmt* du routeur sélectionné.

k. Le bouton *Fill_Queueing_Table* lance la méthode *Fill_Queueing_Table_Click.*
   - Remplit la table *Queueing* du routeur sélectionné.

l. Le bouton *Empty_Queueing_Table* lance *Empty_Queueing_Table_Click.*
   - Vide la table *Queueing* du routeur sélectionné.

m. Le bouton *Fill_Shaping_Table* lance la méthode *Fill_Shaping_Table_Click.*
   - Remplit la table *Shaping* du routeur sélectionné.

n. Le bouton *Empty_Shaping_Table* lance *Empty_Shaping_Table_Click.*
   - Vide la table *Shaping* du routeur sélectionné.

o. Le bouton *Fill_Policing_Table* lance la méthode *Fill_Policing_Table_Click.*
   - Remplit la table *Policing* du routeur sélectionné.

p. Le bouton *Empty_Policing_table* lance la méthode *Empty_Policing_table_Click.*
   - Vide la table *Policing* du routeur sélectionné.

q. Le bouton *Fill_RED_table* lance la méthode *Fill_RED_table_Click.*
   - Remplit la table *RED* du routeur sélectionné.

r. Le bouton *Empty_RED_Table* lance la méthode *Empty_RED_Table_Click.*
   - Vide la table *RED* du routeur sélectionné.

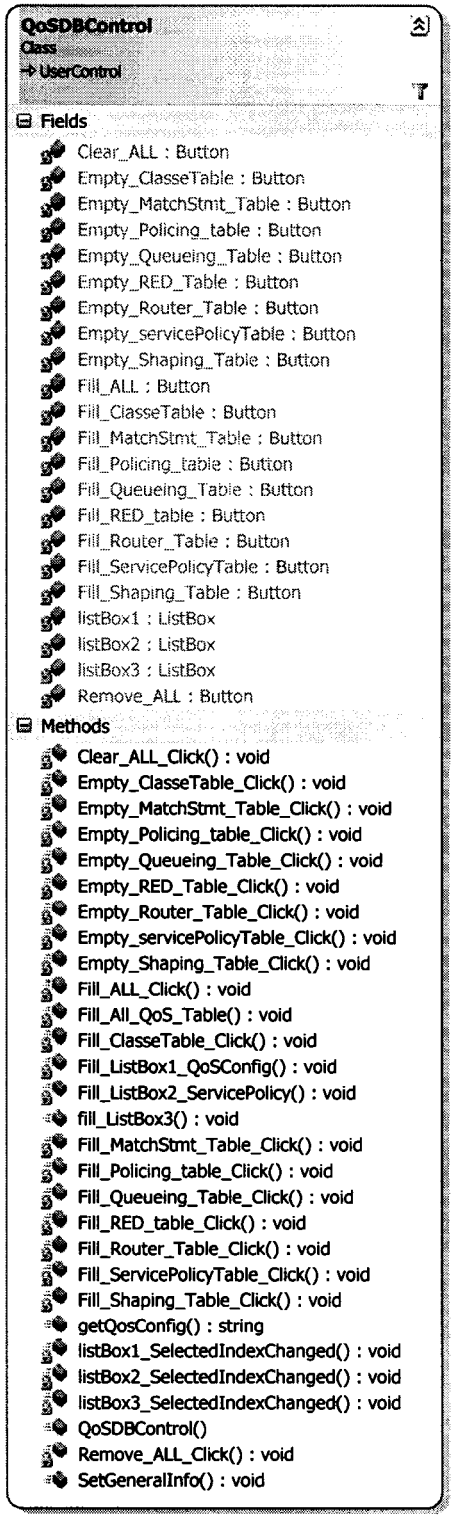


Figure 31 Détail du contrôle utilisateur QoSDBControl du contrôleur de serveur

4.3.4.4 La classe PopUpConfig

Cette classe est utilisée uniquement afin d'afficher une configuration de QdS. Par conséquent elle hérite de *WindowsForm* et ne détient qu'un seul *textBox*. Sa seule méthode est le constructeur de la classe. Elle prend comme paramètre une chaîne de caractères correspondant à la politique de service ainsi qu'une chaîne de caractères correspondant à l'identificateur de la politique de service. La première chaîne de caractères est donc insérée dans le *textbox* tandis que la seconde est insérée comme titre de la fenêtre.

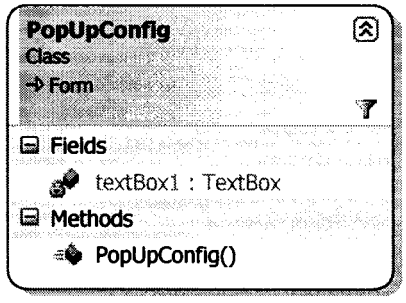


Figure 32 Détail de la classe PopUpConfig

4.3.4.5 Le contrôle utilisateur QoSStatsMonitor

Ce contrôle utilisateur a principalement les mêmes contrôles et fonctionnalités que ceux du contrôle utilisateur *QoSStatsUC* de l'applet. Vous pouvez donc référez à la section 4.3.2.5 pour mieux comprendre les fonctionnalités des *comboBox* et des boutons. La Figure 33 présente le détail de la classe du contrôle utilisateur *QoSStatsMonitor*.

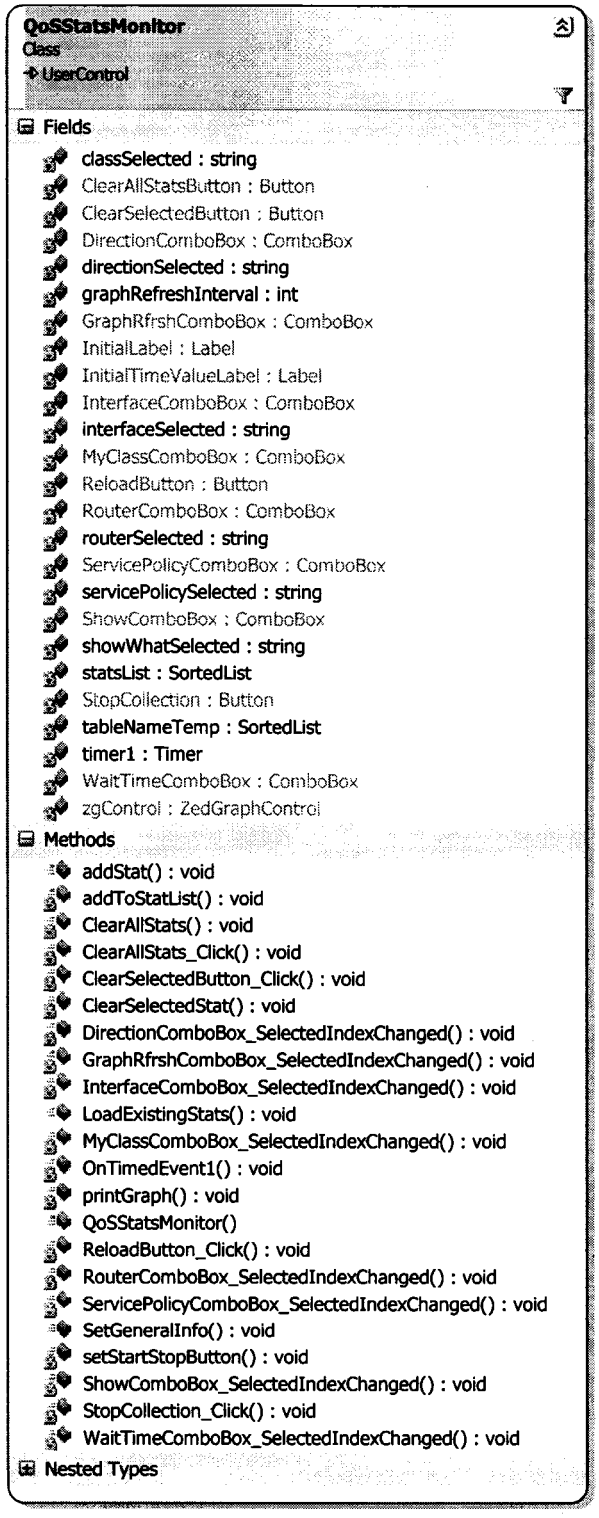


Figure 33 Détail du contrôle utilisateur QoSStatsMonitor

Ce contrôle utilisateur a cependant certaines fonctionnalités que le contrôle utilisateur *QoSStatsUC* de l'applet n'a pas. Dans un premier temps, l'usager a la possibilité de stopper et de redémarrer à sa guise la collecte d'information pour une politique de service donnée. Cette fonctionnalité est principalement utilisée pour le débogage et n'est absolument pas utile pour un utilisateur de l'applet. Dans un deuxième temps, l'usager a la possibilité de modifier le temps d'attente entre deux collectes de statistiques consécutives pour une politique de service donnée. Cette fonctionnalité ne doit pas être accessible à un utilisateur de l'applet étant donné qu'elle peut compromettre les performances du système. Il faut donc qu'elle soit utilisée avec précautions, par un utilisateur averti.

Dans un troisième temps, *QoSStatsMonitor* possède une autre fonctionnalité très importante et qui constitue en fait, sa principale raison d'être. En effet, *QoSStatsMonitor* est en charge de la gestion des instances de surveillance *QoScollecte*. Une telle instance est créée pour chaque politique de service surveillée. Il est important de mentionner que jusqu'à présent, aucun test de performance n'a été effectué afin de déterminer le nombre maximal de politique de service que QMA peut surveiller en concurrence. Mais il est évident que QMA ne serait pas en mesure de surveiller tout un réseau à lui seul. Référez-vous à la section RECOMMANDATIONS pour une architecture plus performante.

Le contrôle utilisateur *QoSStatsMonitor* possède une liste triée (*statsList*) qui contient la liste des instances de *QoScollecte* indexées par un identificateur qui permet de reconnaître une politique de façon unique dans tout le réseau. Cet identificateur est constitué du nom de l'équipement, du nom de l'interface, de la direction ainsi que du nom de la politique tous séparés par un « _ ». Pour ajouter un élément à cette liste, la méthode *addToStatList* doit être appelée. Cette méthode vérifie d'abord si la statistique à ajouter existe déjà dans la liste. Si elle n'y est pas, elle crée une nouvelle instance de *QoScollecte* et l'ajoute à la liste tout simplement. La section 4.3.4.6 présente plus en détail cette dernière classe.

4.3.4.6 La classe QoScollecte

Cette classe est utilisée afin de récolter les statistiques d'une politique de services de façon périodique. La Figure 34 en présente le détail.

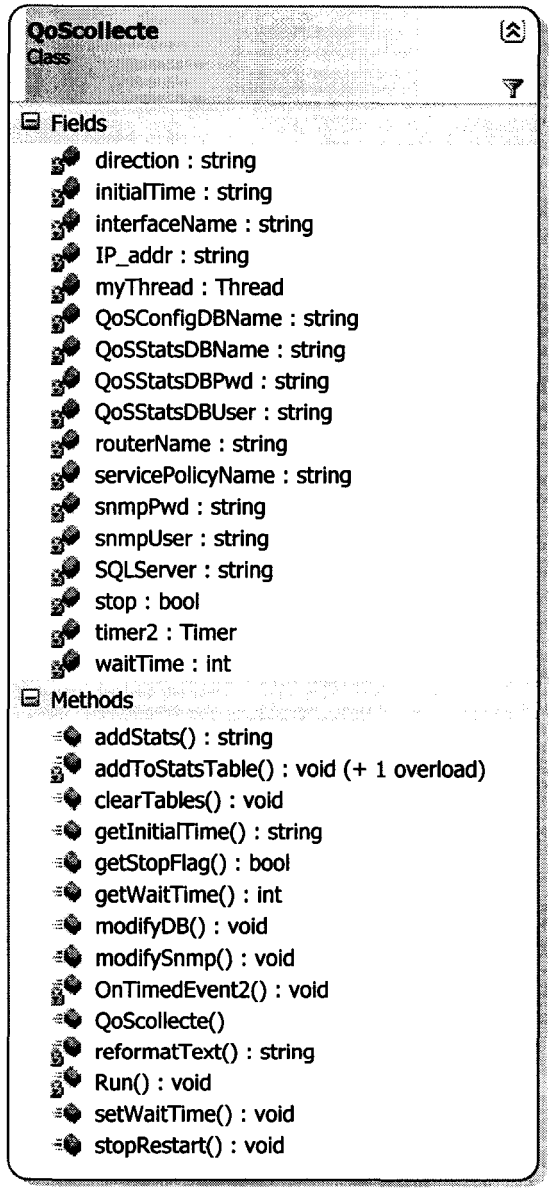


Figure 34 Détail de la classe QoScollecte

D'abord on peut voir dans cette figure que cette classe nécessite la connaissance des informations générales au sujet des bases de données et des configurations SNMP. Ces divers attributs sont introduits dans les paramètres du constructeur de la classe. Les méthodes *modifyDB* et *modifySnmp* ont été créées afin de pouvoir modifier ces attributs. De plus, les informations permettant d'identifier la politique de service de façon unique (Nom du routeur, nom de l'interface, direction et nom de la politique) sont également données dans les paramètres du constructeur. Lors de l'exécution de ce dernier, la méthode *addStats* est appelée afin de créer les tables requises, si nécessaire, dans la base de données (voir ANNEXE 2 pour plus de détail sur ces tables). Par la suite, l'adresse IP requise pour pouvoir communiquer avec l'équipement en question, est récupérée de la table *Router*. Puis, la date et l'heure de la création de l'objet sont conservées dans l'attribut *initialTime*. Finalement un *thread* est créé et lancé afin d'exécuter la méthode *Run* dont l'organigramme est illustré à la Figure 35.

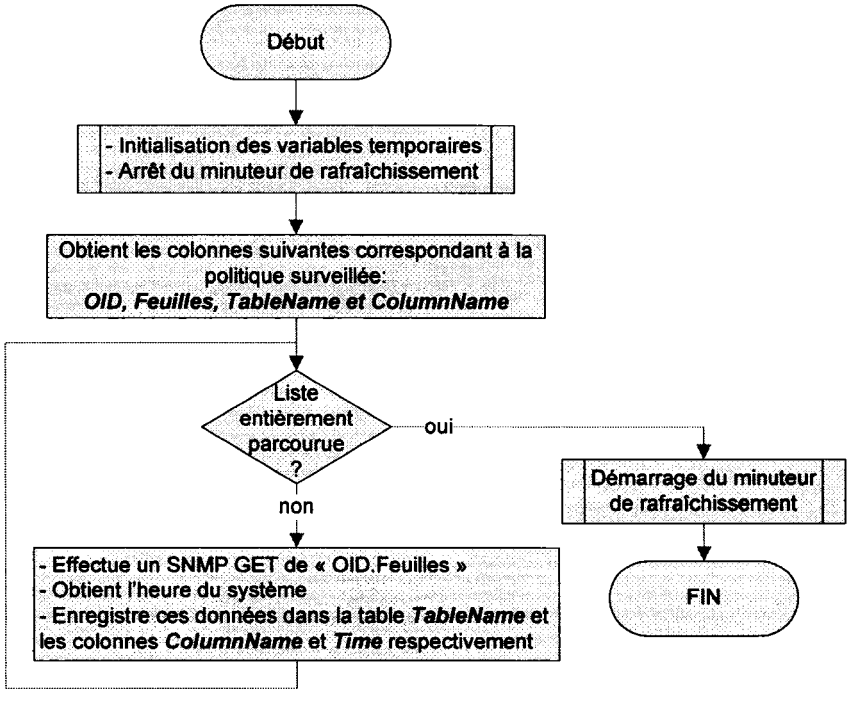


Figure 35 Organigramme de la méthode Run

Dans cette figure, on voit que le *thread* (*myThread*) débute en arrêtant le minuteur (*Timer2*) et en obtenant les colonnes *OID*, *Feuilles*, *TableName* et *ColumnName* depuis la table *StatsTablesInfo*. Par la suite chaque ligne du *DataSet* obtenu sera traitée en boucle. Pour chaque combinaison *OID.Feuilles* de la liste, une requête SNMP GET sera envoyée à l'équipement approprié afin d'obtenir la statistique désirée. Par la suite, la référence temporelle sera prise en se basant sur la date et l'heure du système à laquelle la réponse SNMP aura été obtenue. Finalement, la nouvelle statistique sera introduite dans la colonne *ColumnName* et la référence temporelle dans la colonne *Time* de la table *TableName*. Lorsque toute la liste aura été parcourue, le minuteur sera redémarré afin d'attendre un certain temps avant d'effectuer cette routine à nouveau. À chaque fois que le minuteur expire, la méthode *OnTimedEvent2* est lancée afin de démarrer à nouveau l'exécution de la méthode *Run* par *myThread*.

Le reste des méthodes est décrit dans le Tableau XVII.

Tableau XVII

Descriptions de quelques méthodes de la classe QoScollecte

Méthodes	**Descriptions**
clearTables	Supprime toutes les tables de la base de données qui correspondent à la politique de service surveillée par cet objet. Supprime également toutes les entrées correspondantes dans la table *StatsTablesInfo*.
getInitialtime	Retourne la date et l'heure de création de l'objet de surveillance.
getStopFlag	Retourne l'attribut *stop* qui informe si la collecte de statistiques est stoppée ou non.
getWaitTime	Retourne l'attribut *initialTime* qui correspond à l'intervalle d'attente entre deux collectes de statistiques.

Tableau XVII (suite)

Méthodes	Descriptions
reformatText	Reformate le texte passé en paramètre afin de remplacer les caractères '.', '/' et '-' par le caractère '_'. Cette méthode est particulièrement utile pour donner des noms significatifs aux tables et aux colonnes de la base de données étant donné qu'il s'agit de caractères interdits. Par exemple, les chaînes de caractères « FastEthernet0/0.50 » ou « class-default » ne seraient pas utilisables.
setWaitTime	Reconfigure l'attribut *waitTime* qui définit l'intervalle d'attente entre deux collectes de statistiques consécutives (utilisé par le minuteur *Timer2*).
stopRestart	Arrête ou redémarre la collecte de statistiques et configure l'attribut *stop* en conséquence.

4.4 Interaction entre les composantes de l'architecture générale

Cette section présente les interactions entre les diverses composantes de l'architecture. D'abord, la première sous-section présentera un diagramme de classes général afin de voir comment chacune des classes est utilisée. Quant à la seconde sous-section, elle présentera les diagrammes de séquences qui permettent de voir l'interaction entre les composantes de l'architecture et ce, pour les principales opérations pouvant être initiées par l'utilisateur.

4.4.1 Diagramme de classes

Cette section présente le diagramme de classe général, présenté à la Figure 36, dans lequel l'interaction entre les diverses classes du système est représentée. On voit que la classe *QMA* de l'applet possède une référence vers le service Web *QMA_WebService*.

Tel que mentionné dans la section 4.3.2, cette référence est par la suite distribuée aux contrôles utilisateur de l'applet qui la nécessite.

Par ailleurs, le service Web utilise une instance de la classe *Control_Auth* de *ModulesQMA* afin d'authentifier un usager. Cette classe possède également une instance de *Control_DB* et de *Database* de *ModulesQMA* afin d'interagir avec le réseau et/ou la base de données. Ce service Web peut créer, lorsque requis, une instance de la classe *Socket* afin de communiquer avec le contrôleur de serveur.

Quant aux différents modules de *ModulesQMA*, ceux-ci sont créés de façon sporadique par le service Web ou encore par le contrôleur de serveur. En fait, leurs instances sont créées à chaque fois que le service Web ou le contrôleur de serveur désire exécuter une opération d'authentification ou encore une opération sur la base de données ou sur le réseau.

Finalement, le contrôleur de serveur possède des instances des classes de *ModulesQMA* dont *Database*, *Control_DB* et *Snmp*. La première étant utilisée lorsque le contrôleur de serveur désire effectuer des opérations sur la base de données uniquement. La seconde étant utilisée lorsque le contrôleur de serveur désire remplir la base de données selon les configurations du réseau. La troisième étant utilisée par la classe *QoScollecte* afin que le contrôleur de serveur puisse interroger le réseau pour obtenir les statistiques qu'il désire. Finalement, cette composante de l'architecture possède une instance de la classe *ClientUdp*, qui hérite de la classe *Socket*, utilisée pour recevoir des commandes du service Web.

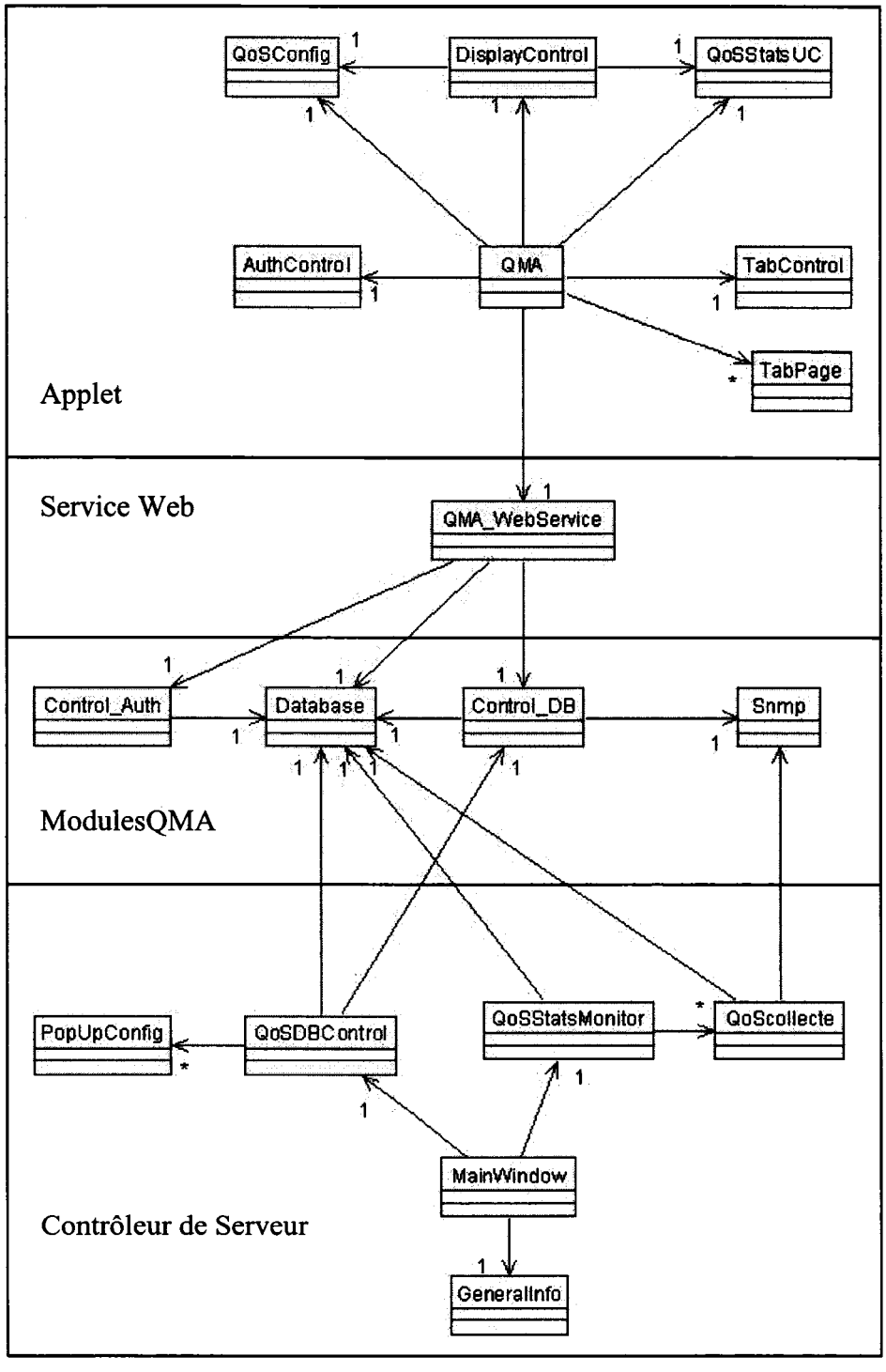


Figure 36 Diagramme de classe général montrant l'interaction entre les modules

4.4.2 Diagrammes de séquences

Cette section présente les diagrammes de séquences des principales actions pouvant être exécutée par un utilisateur. Le principal but de cette section est de représenter, pour ces actions, les interactions entre les diverses composantes de l'architecture. Encore une fois, seules les actions concernant la QdS seront représentées. Tel qu'il peut être vu dans le manuel de l'utilisateur (voir ANNEXE 3), ces actions consistent à demander, pour un équipement donné, un renouvellement des configurations de QdS. Par ailleurs, l'usager peut demander de visualiser une configuration, ou encore de démarrer une collecte de statistique.

4.4.2.1 Opération *ReNew QoS Configurations*

La Figure 37 représente le diagramme de séquence réalisé lorsque l'usager sélectionne l'option *ReNew QoS Configurations*. La classe *DisplayControl* de l'applet appelle alors la méthode *renewQoSConfig* du service Web. Par la suite, ce dernier appelle la méthode *fill_QoS_Config_Tables* qui est en charge d'interroger l'équipement en question et de remplir la base de données de configurations de QdS.

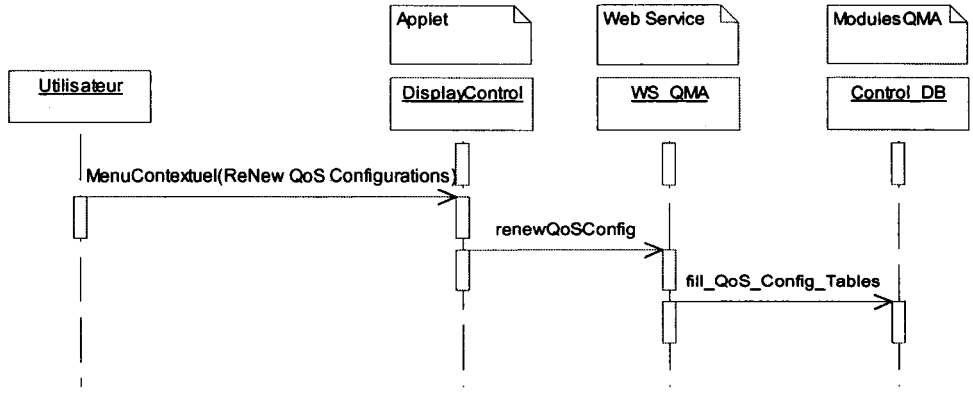


Figure 37 Diagramme de séquence de l'opération *ReNew QoS Configurations*

4.4.2.2 Opération *Show Service Policy*

La Figure 38 représente, quant à elle, la séquence d'exécution lorsque l'utilisateur a sélectionné l'option *Show Service Policy* du menu contextuel. Le contrôle utilisateur obtient la liste des interfaces, via le service Web et la classe *Database*, sur lesquelles une configuration de QdS a été appliquée. Cette liste est alors affichée dans le *listBox1*. Lorsque l'utilisateur sélectionne une interface, le contrôle utilisateur interroge la base de données, via le service Web, afin d'obtenir la liste des politiques de service appliquées sur l'interface sélectionnée. Cette liste est alors affichée dans le *listBox2*.

Lorsque l'utilisateur sélectionne la politique de service dont il désire voir la configuration, le contrôle utilisateur *DisplayControl* appelle la méthode *getQoSConfig* du service Web. Ce dernier communique avec le contrôleur de serveur pour lui envoyer la commande *GetQoSConfig*. Lorsque le contrôle utilisateur *MainWindow* du contrôleur de serveur intercepte la commande, il lance alors la méthode *getQosConfig* du contrôle utilisateur *QoSDBControl*. Ce dernier formate, dans une chaîne de caractères, les informations de configuration obtenues de la base de données et retourne cette chaîne de caractères au *MainWindow*. Ce dernier retourne la réponse au service Web qui la retourne également au contrôle utilisateur *DisplayControl* de l'applet. Finalement, la configuration est ajoutée à la liste de configuration du contrôle utilisateur *QoSConfig*.

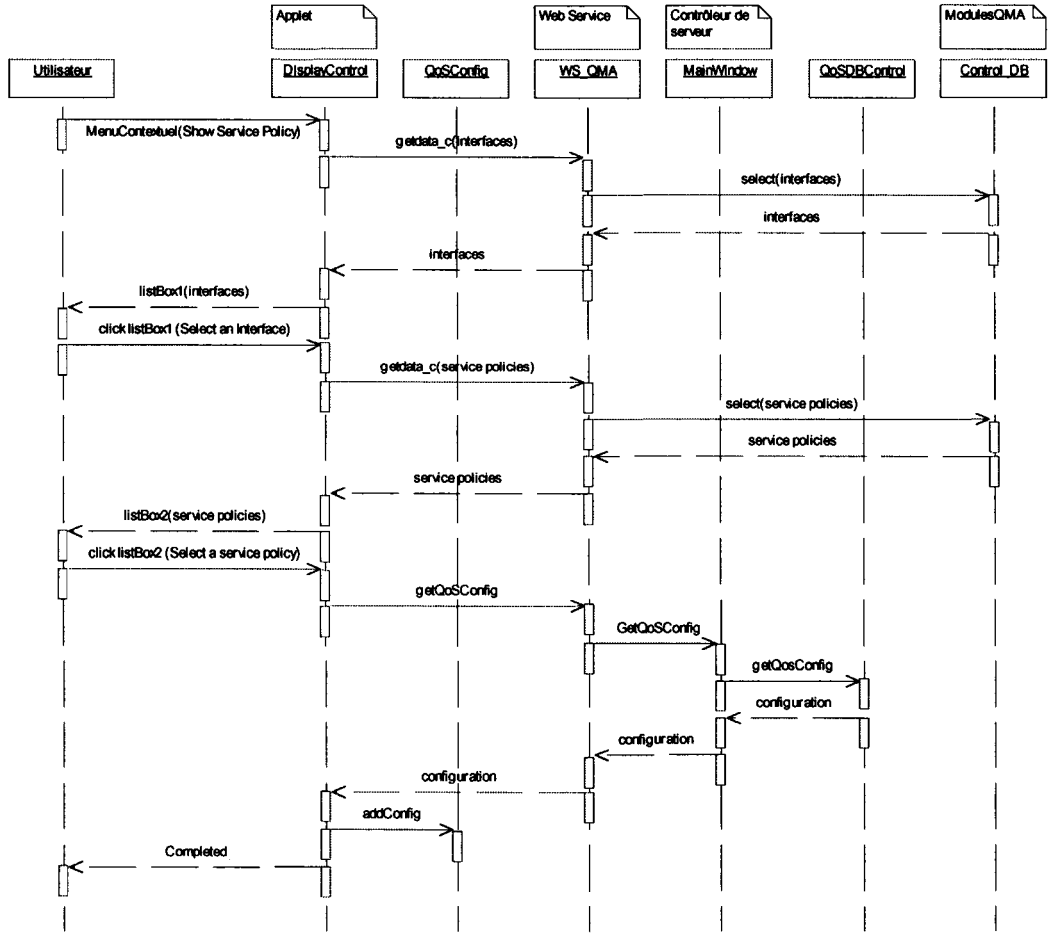


Figure 38 Diagramme de séquence de l'opération *Show Service Policy*

4.4.2.3 Opération *Begin Statistic Collection*

La Figure 39 représente la séquence d'exécution réalisée lorsque l'utilisateur sélectionne l'option *Begin Statistic Collection* du menu contextuel. Tel que spécifié lors de la séquence d'exécution de l'option *Show Service Policy*, le *listBox1* dresse la liste des interfaces sur lesquelles une politique de service est appliquée. Lorsque l'utilisateur

sélectionne une interface, le *listBox2* dresse alors la liste des politiques de service appliquées sur l'interface.

Lorsque l'utilisateur sélectionne une politique de service, le contrôle utilisateur *DisplayControl* de l'applet lance la méthode *addStats* du contrôle utilisateur *QoSStatsUC*. Ce dernier appelle la méthode *addQoSStats* du service Web lequel envoie un message au contrôleur de serveur dont le nom de commande est *addStats*. Ce message est alors intercepté par le contrôle utilisateur *MainWindow* du contrôleur de serveur lequel lance la méthode *addStat* de *QoSStatsMonitor*. Cette méthode vérifie dans sa liste si elle possède une instance de *QoScollecte* correspondant à celle demandée. Si non, un nouvel objet *QoScollecte* est créé. Trois messages peuvent être retournés par le *QoSStatsMonitor*. Le message *OK* sera retourné si l'opération a bien été exécutée, le message *Error* sera retourné si l'opération a échouée et le message *Already Exist* sera retourné si les tables existaient déjà dans la base de données.

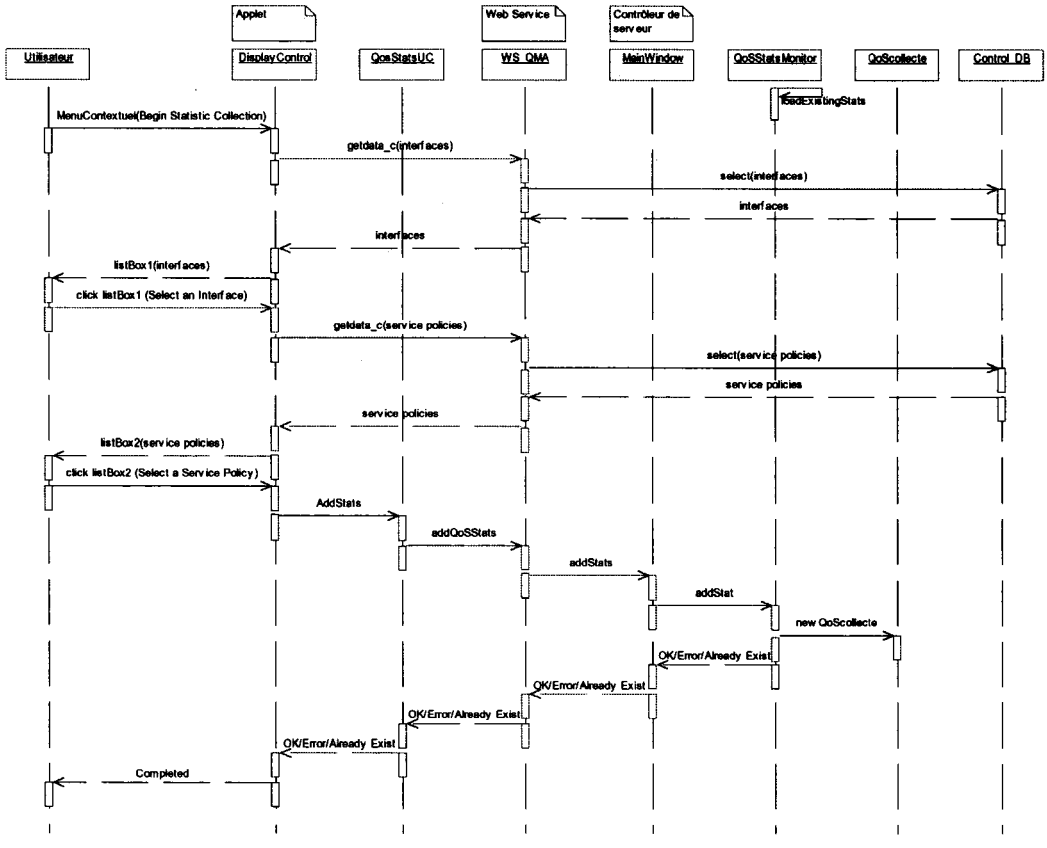


Figure 39 Diagramme de séquence de l'opération *Begin Statistic Collection*

4.5 Intégration d'autres équipementiers à QMA

Cette section présente une approche pouvant être utilisée, pour le développement futur de QMA, afin que le système puisse interragir avec des équipements d'un équipementier autre que Cisco Systems. La base de données actuellement créée et présentée à l'ANNEXE 2 utilise un format général pouvant être adapté, s'il y a lieu, pour qu'il puisse supporter de nouvelles MIB. Dans le cas où les modifications à cette base de données seraient trop importantes pour simplement y interfacer une nouvelle MIB, une nouvelle base de données pourrait être créée pour cette MIB spécifique. Dans une telle

situation, une table devra être créée pour conserver la liste des équipements et de la base de données qu'ils utilisent. Cette table pourrait, par exemple, inclure les colonnes suivantes :

a. ***HostName*** : Nom d'hôte de l'équipement.
b. ***IP*** : Adresse IP utilisée pour rejoindre l'équipement.
c. ***Constructor*** : Nom de l'équipementier.
d. ***Model*** : Nom ou numéro du modèle de l'équipement.
e. ***QoSConfigDatabase*** : Nom de la base de données utilisée pour emmagasiner les configurations de QdS de cet équipement.

Une fois ceci réalisé, certaines modifications devront être apportées à la classe *ControlDB* de *ModulesQMA* afin que QMA aille d'abord obtenir les informations sur l'équipementier et le modèle de l'équipement à interroger. Une fois que *ControlDB* connaît ces informations, il n'aurait donc qu'à utiliser les OID adéquats et à sauvegarder les réponses dans la base de données correspondante. Similairement, à chaque fois qu'une classe aurait besoin d'accéder à la base de données de configuration de QdS, cette classe devrait préalablement aller voir dans cette table quelle base de données interroger.

Pour conclure, après avoir décrit l'architecture générale du système QMA, ce chapitre a détaillé chaque composante de cette architecture. Pour chaque composante, les classes utilisées ainsi que leurs principaux attributs et méthodes ont été listés et décrits. Par la suite, ce chapitre a expliqué l'interaction entre les diverses composantes de l'architecture et ce, pour les principales commandes pouvant être lancées par l'utilisateur. Pour terminer, ce chapitre a présenté une approche pouvant être utilisée pour permettre au système QMA d'interragir avec plusieurs équipementiers. Le chapitre qui suit valide le fonctionnement de ces commandes et, du fait même, l'interaction entre les diverses composantes de l'architecture générale.

CHAPITRE 5

RÉSULTATS ET ANALYSE

Suite à la conception du système QMA, il s'avère impératif d'en valider les principales fonctionnalités. Ce chapitre présente donc les divers résultats de validation. En fait, ces résultats prouvent le bon fonctionnement des principales fonctionnalités relatives à la qualité de service. D'abord l'option de renouvellement des configurations sera présentée, puis l'affichage des configurations et enfin, l'affichage des statistiques des mécanismes de QdS. Mais avant de débuter la validation, il est essentiel de savoir dans quel contexte le système a été testé. La Figure 40 présente l'architecture du banc d'essai tel que présenté par le système QMA.

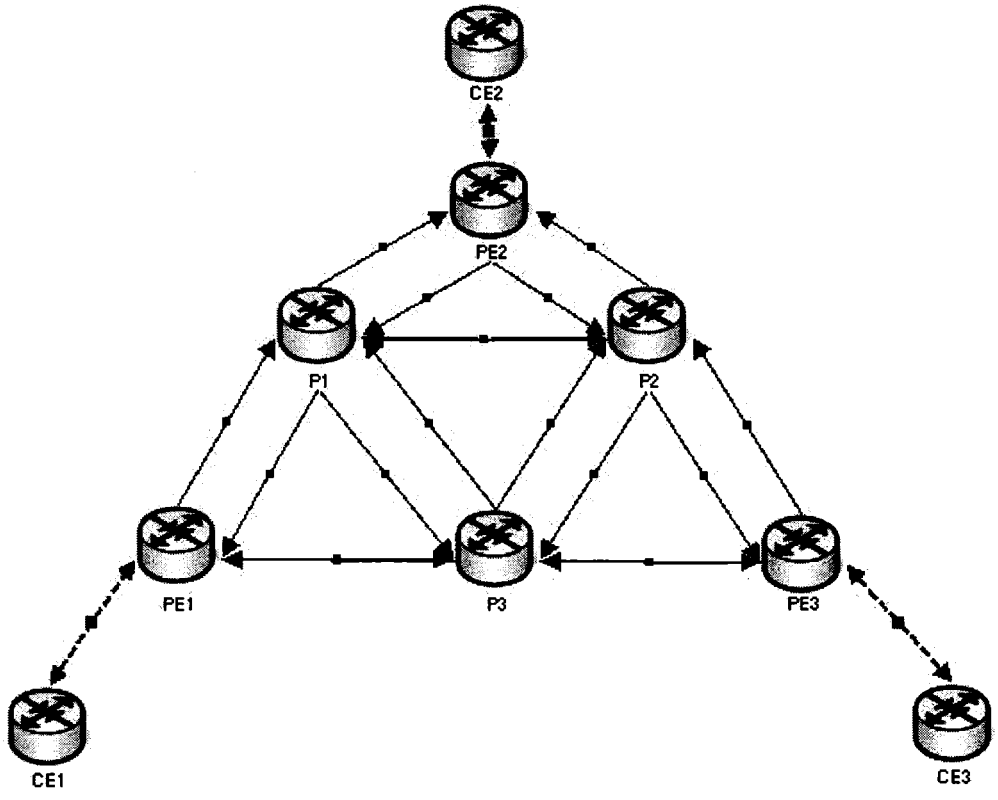


Figure 40 Schéma du réseau surveillé

Dans la figure précédente, on distingue trois types de routeurs dont les noms commencent respectivement par CE (*Customer Edge router*), PE (*Provider Edge router*) et P (*Provider router*). Un routeur dont le nom commence par CE, indique qu'il s'agit d'un routeur situé chez le client du fournisseur de service et qu'il est en bordure du réseau du client et du fournisseur. Pour le client, ce routeur sert d'accès au réseau du fournisseur. Cependant, pour des fins de tests, nous posons l'hypothèse que ce type de routeur est sous le contrôle du système QMA.

Un routeur dont le nom commence par PE, indique qu'il s'agit d'un routeur situé dans le réseau du fournisseur. Ce type de routeur effectue la transition entre le réseau non-MPLS (réseau du client) et le réseau MPLS (réseau du fournisseur de service).

Enfin, un routeur dont le nom débute par P signifie qu'il s'agit d'un routeur situé dans le cœur du réseau MPLS du fournisseur de service. Le principal rôle de ce type d'équipement est de commuter les paquets en se basant sur les *labels* de l'entête MPLS.

D'un point de vue des mécanismes de qualité de service, les principales configurations se situent sur les CE et les PE. C'est pourquoi les sous-sections qui suivent se concentreront principalement sur ce type d'équipement.

5.1 Renouvellement des configurations

Cette section présente les résultats obtenus lorsque l'option *ReNew QoS Configurations* est sélectionnée (voir Figure 41). Une fois l'exécution de cette commande terminée, un message est affiché dans une bulle située en bas et à droite de l'écran (voir Figure 42). Notez que cette bulle n'est affichée que pour une période de 20 secondes. Étant donné qu'un utilisateur pourrait, pour une raison quelconque, ne pas voir cette bulle lors de sa période d'affichage, une nouvelle entrée est insérée dans l'onglet *Events*. On peut voir,

dans la Figure 43, qu'il y eut un *ReNew QoS Configurations* d'exécuté pour chacun des routeurs du réseau.

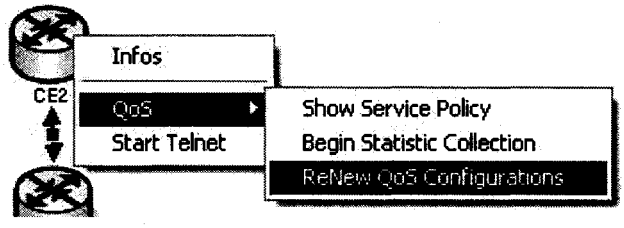


Figure 41 Sélection de l'option *ReNew QoS Configurations*

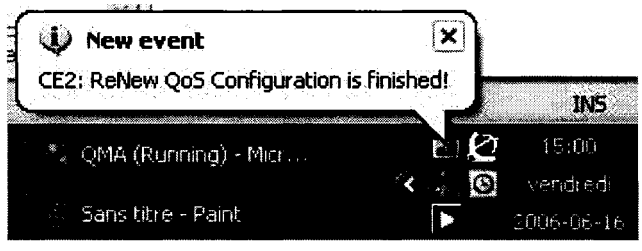


Figure 42 Bulle d'information apparaissant lorsque l'exécution de l'option *ReNew QoS Configurations* est terminée

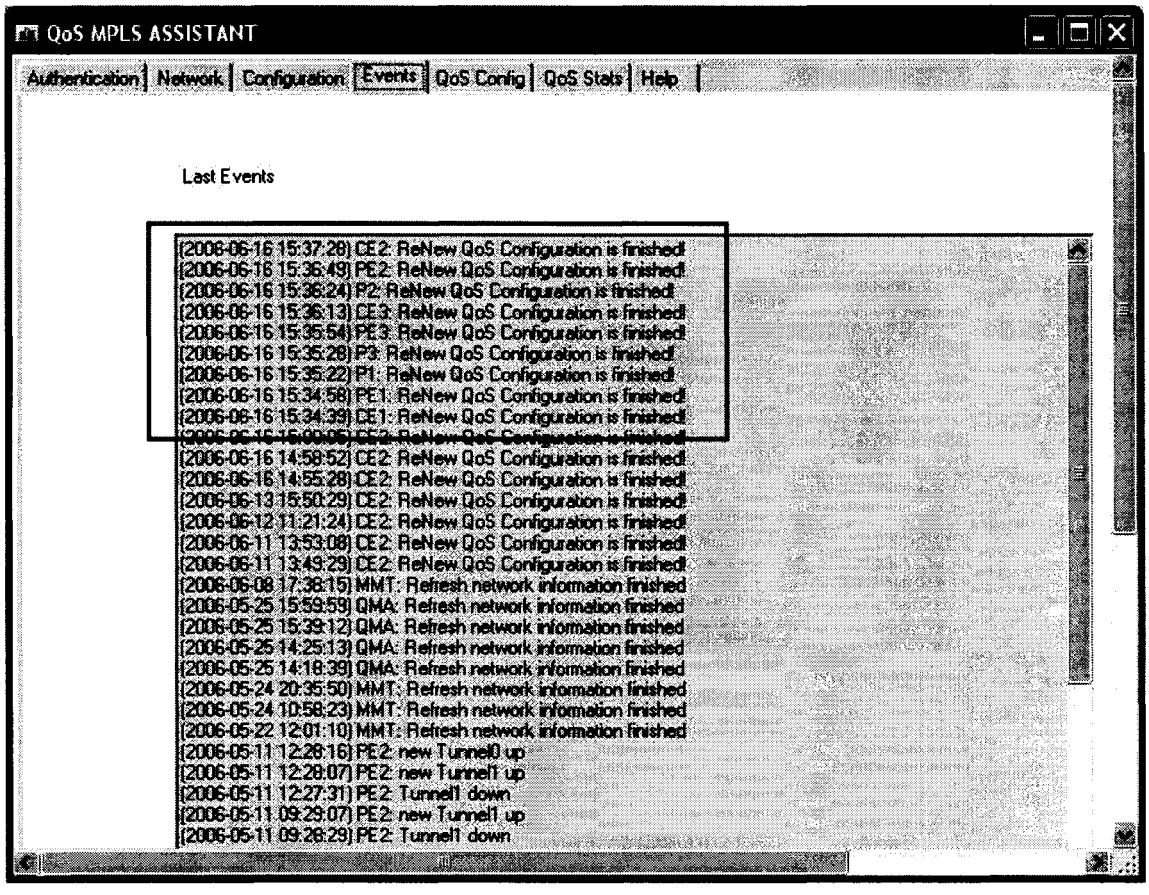


Figure 43 Événement affiché lorsque l'option *ReNew QoS Configurations* est terminée

5.2 Affichage des configurations

L'ANNEXE 3 présente les manipulations à effectuer afin de pouvoir visualiser une configuration de QdS sur un équipement spécifique. La présente section montre une configuration de QdS réalisée sur le routeur CE2. La Figure 44 présente un message qui a pour but d'informer l'utilisateur qu'il n'y a pas de configuration de QdS sur l'équipement qu'il a sélectionné.

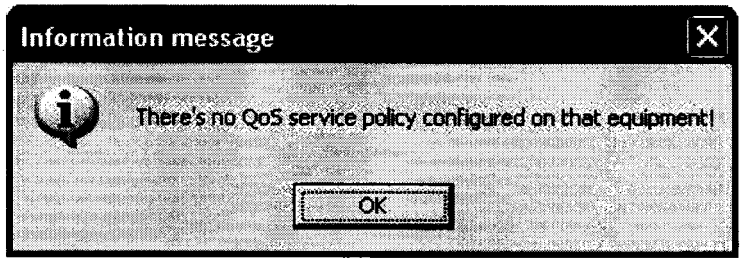


Figure 44 Message indiquant qu'il n'y a pas de configuration de QdS sur l'équipement sélectionné

Par ailleurs, la Figure 45 montre la configuration obtenue avec la commande *show run* effectuée dans le CLI. Dans cette figure, l'information a été triée afin de ne montrer que celle qui est pertinente à cette politique. Normalement cette configuration est mélangée avec d'autres configurations, c'est donc à l'opérateur d'en retirer les informations pertinentes.

La Figure 46 et la Figure 47 montrent la configuration obtenue avec QMA. La première a été obtenue avec l'interface utilisateur de l'applet tandis que la seconde a été obtenue depuis le contrôleur de serveur. Le format des configurations obtenues avec QMA est expliqué dans le manuel utilisateur de l'ANNEXE 3. En comparant ces trois figures, il est clair que le format offert par QMA synthétise mieux les informations de configuration d'une politique de service. On peut entre autre y voir aisément la hiérarchie entre les politiques parent et enfant. Le concept utilisé pour représenter cette hiérarchie consiste à insérer un retrait supplémentaire en début de ligne à tout ce qui a un niveau hiérarchique inférieur. Par exemple, toutes les classes qui sont sous-jacentes hiérarchiquement à une politique ont un retrait supplémentaire par rapport au retrait du nom de la politique de service. Similairement, tous les mécanismes de QdS configurés au sein d'une classe ont un retrait supplémentaire par rapport au retrait appliqué pour la classe.

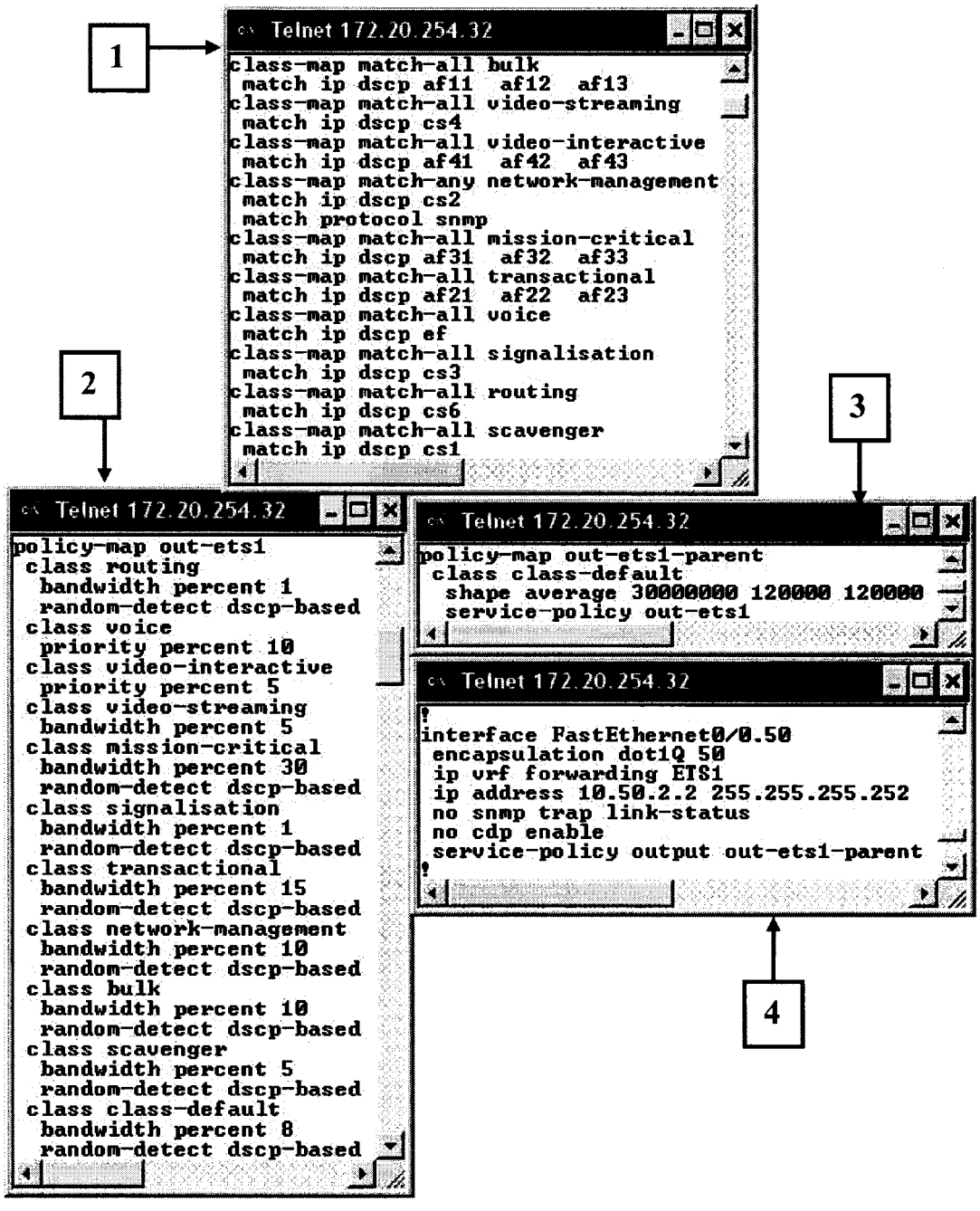


Figure 45 Configuration de la politique *out-ets1-parent* sur CE2 obtenue par la commande *show run*

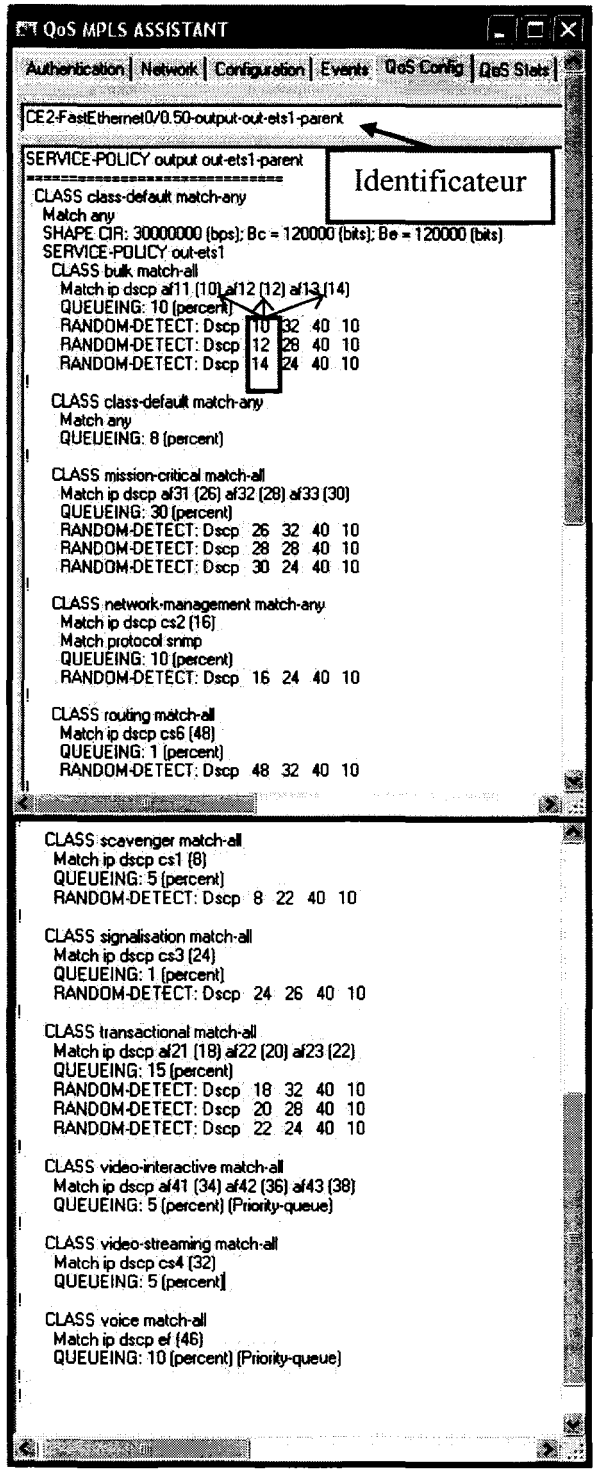


Figure 46 Configuration de la politique *out-ets1-parent* sur CE2 obtenue avec l'applet de QMA

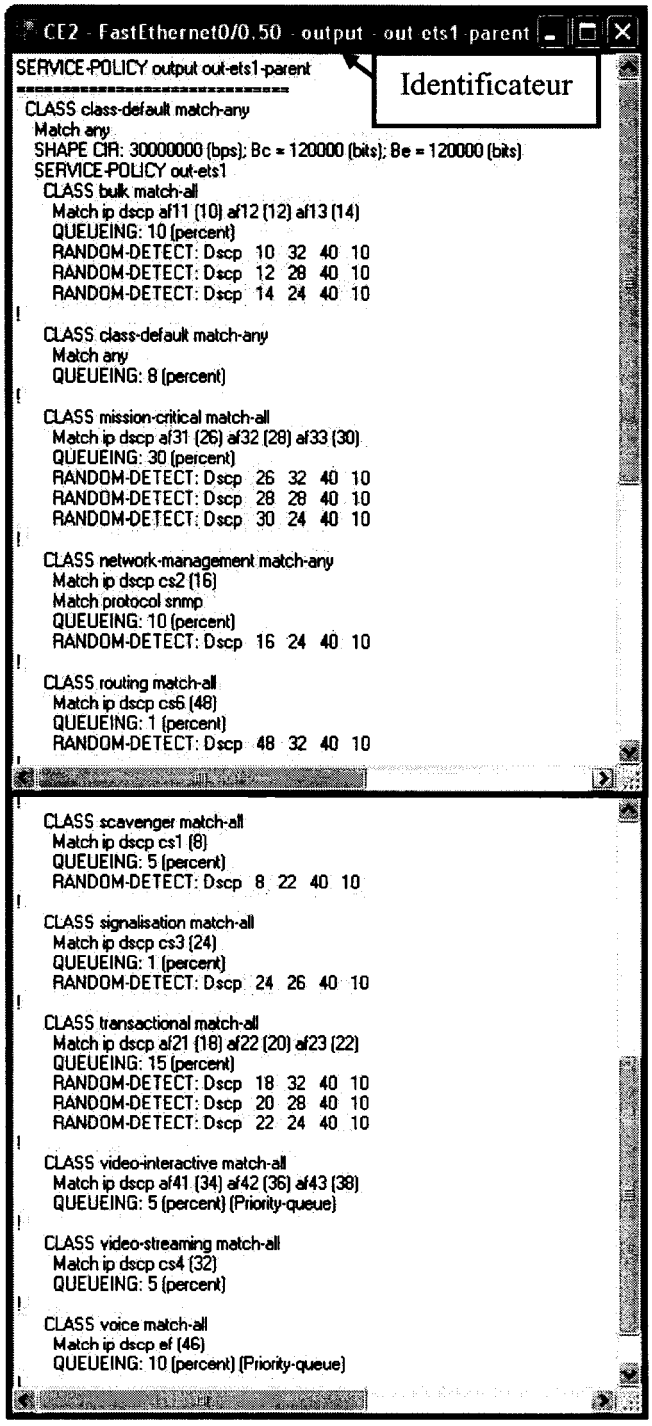


Figure 47 Configuration de la politique *out-ets1-parent* sur CE2 obtenue avec le contrôleur de serveur de QMA

5.3 Affichage des statistiques

Tel que stipulé précédemment, cette section a pour but de valider que la récolte et l'affichage de statistiques des mécanismes de QdS se fait adéquatement. Afin de pouvoir valider chaque mécanisme, un seul scénario a été réalisé. Dans ce scénario, en référant à la Figure 40, du trafic a été généré de l'interface LAN de CE2 vers l'interface LAN de CE3. Deux politiques de service ont été appliquées, soit la politique *out-ets1-parent*, vue dans la section précédente, ainsi que la politique *in-ets1-police* (voir Figure 48), toutes deux étant appliquées sur CE2. Par conséquent, ce routeur sera le seul surveillé mais pour les deux politiques de service simultanément.

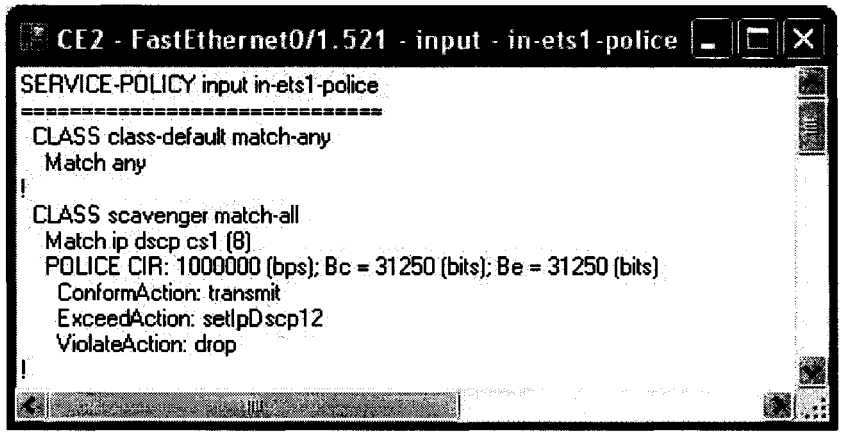


Figure 48 Configuration, vue du contrôleur de serveur, de la politique *in-ets1-police*

La première politique de service, *out-ets1-parent*, est appliquée sur le trafic sortant de l'interface FastEthernet0/0.50 de CE2, soit l'interface WAN. Sachant que cette politique effectue du lissage à un taux de 30Mbps, plus de 30Mbps ont été générés de façon à créer un goulot d'étranglement sur cette interface. Ainsi les mécanismes de QdS, configurés dans cette politique de service, ont été actifs pour la durée de l'essai. Par conséquent, la prise des statistiques suivantes a pu être validée en surveillant uniquement cette politique de service, sur le routeur CE2.

a. CPU
b. Class-Map
c. Queueing
d. Lissage (*Shaping*)
e. Évitement de congestion (RED)

La seconde politique de service, *in-ets1-police*, est appliquée sur le trafic entrant par l'interface *FastEthernet0/1.521* du router CE2, soit l'interface LAN. Cette politique n'inclut qu'une seule classe (*scavenger*), à l'exception de la classe *class-default*, qui effectue du *policing* à un taux de 1Mbps. Sa configuration consiste à transmettre intacte le trafic conforme, transmettre en remarquant la valeur DSCP à 12 (soit AF12) le trafic en excès puis rejeter le trafic violant les paramètres de configuration du *policing*. Cette politique est utilisée pour valider la prise de statistiques pour le mécanisme de *policing*.

Les résultats pouvant être visualisés pour chacune de ces statistiques sont résumés dans le Tableau XX du manuel utilisateur (ANNEXE 3).

Étant donné que la politique de service *out-ets1-parent* effectue du lissage à un taux de 30Mbps, approximativement 45Mbps on été injectés dans le réseau afin de s'assurer que les mécanismes de QdS étaient bien actifs. Le Tableau XVIII présente le débit injecté dans chacune des classes configurées dans la politique de service. On remarque que le débit total est de 45Mbps et qu'il est divisé entre les classes, en suivant le pourcentage de bande passante configuré à chacune d'elle dans la politique de service *out-ets1-parent*. Ainsi, chaque classe reçoit plus de trafic qu'elle ne peut en transmettre.

Tableau XVIII

Débit injecté par classe et marque appliquée à ce débit lors de sa génération

Nom de la classe	Marque DSCP appliquée au trafic généré	Pourcentage du trafic total injecté (%)	Débit injecté (Mbps)
routing	CS6	1	0.45
voice	EF	10	4.5
video-interactive	AF41	5	2.25
video-streaming	CS4	5	2.25
mission-critical	AF31	30	13.5
signalisation	CS3	1	0.45
transactional	AF21	15	6.75
network-management	CS2	10	4.5
bulk	AF11	10	4.5
scavenger	CS1	5	2.25
class-default	0	8	3.6
Total	**N/A**	**100**	**45**

Dans le tableau précédent, on voit que 2.25Mbps sont injectés dans la classe *scavenger*, ainsi le mécanisme *policing* de la politique de service *in-ets1-police* sera également actif et doit rejeter environ 1.25Mbps de trafic.

Les sous-sections qui suivent présentent donc divers résultats, récoltés lors de ce test, dont le but est de valider le bon fonctionnement de la prise de statistiques des mécanismes de QdS.

5.3.1 CPU

La performance des équipements est une des principales causes de la dégradation de la QdS. En effet, si un équipement a été placé dans un contexte où la quantité d'informations à commuter est trop grande par rapport à ce que l'équipement peut supporter, ce dernier rejettera des paquets de façon purement aléatoire. Un très bon indicateur, permettant de voir si l'équipement a atteint son niveau de saturation, est le pourcentage d'utilisation du CPU.

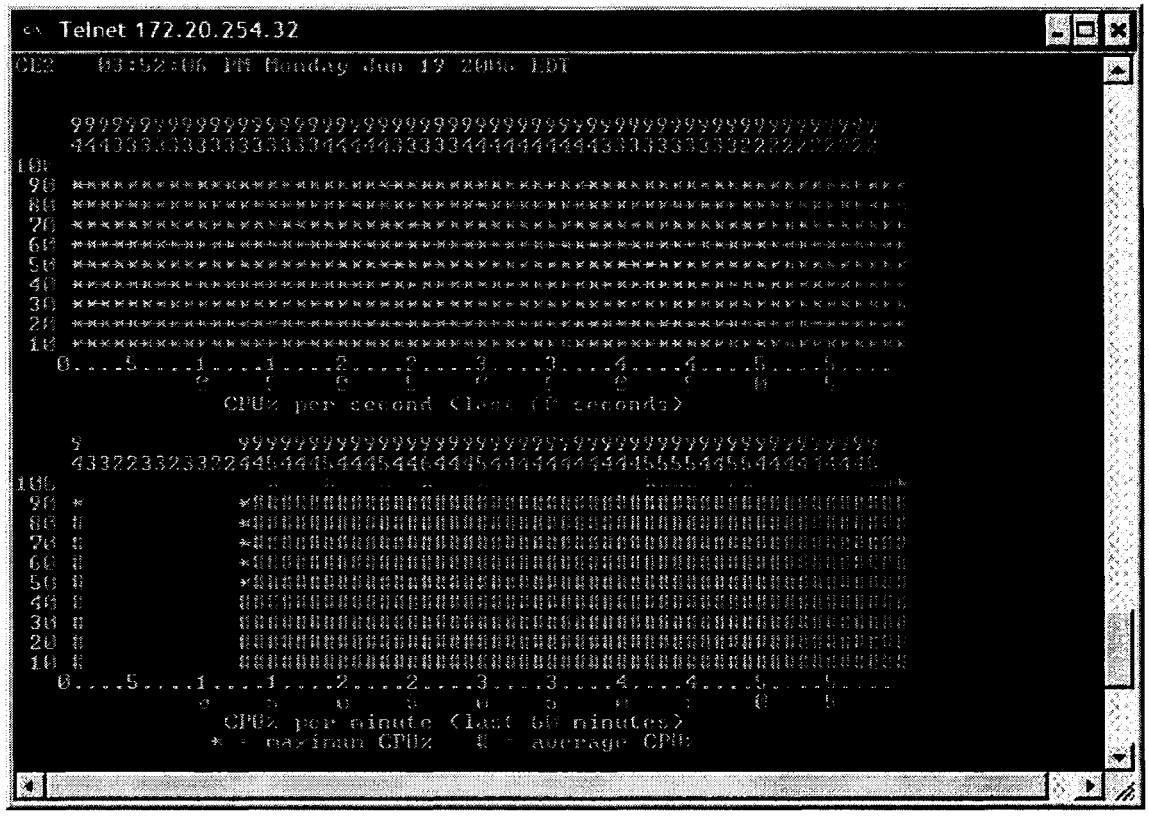


Figure 49 Pourcentage d'utilisation du CPU obtenu via une commande dans le CLI

La Figure 49 montre la statistique d'utilisation du CPU obtenue en exécutant la commande ***show processes cpu history*** dans le CLI. La Figure 50 montre, quant à elle, le pourcentage d'utilisation du CPU obtenu avec QMA. Notez que les valeurs obtenues

avec QMA correspondent approximativement aux valeurs obtenues avec la commande exécutée dans le CLI. En fait, cette dernière effectue une moyenne sur un intervalle régulier (seconde, minute ou heure) tandis que QMA obtient les valeurs exactes obtenues lors de l'interrogation de l'agent SNMP. Un avantage à utiliser QMA, pour la statistique de CPU, est que la statistique peut être récoltée constamment tandis que la statistique obtenue à l'aide de la commande dans le CLI, ne montre que l'historique des 72 dernières heures. De plus, pour chacune des 72 heures, seul une moyenne calculée pour chaque heure est affichée. Par conséquent, les résultats affichés dans cet historique ne sont pas précis.

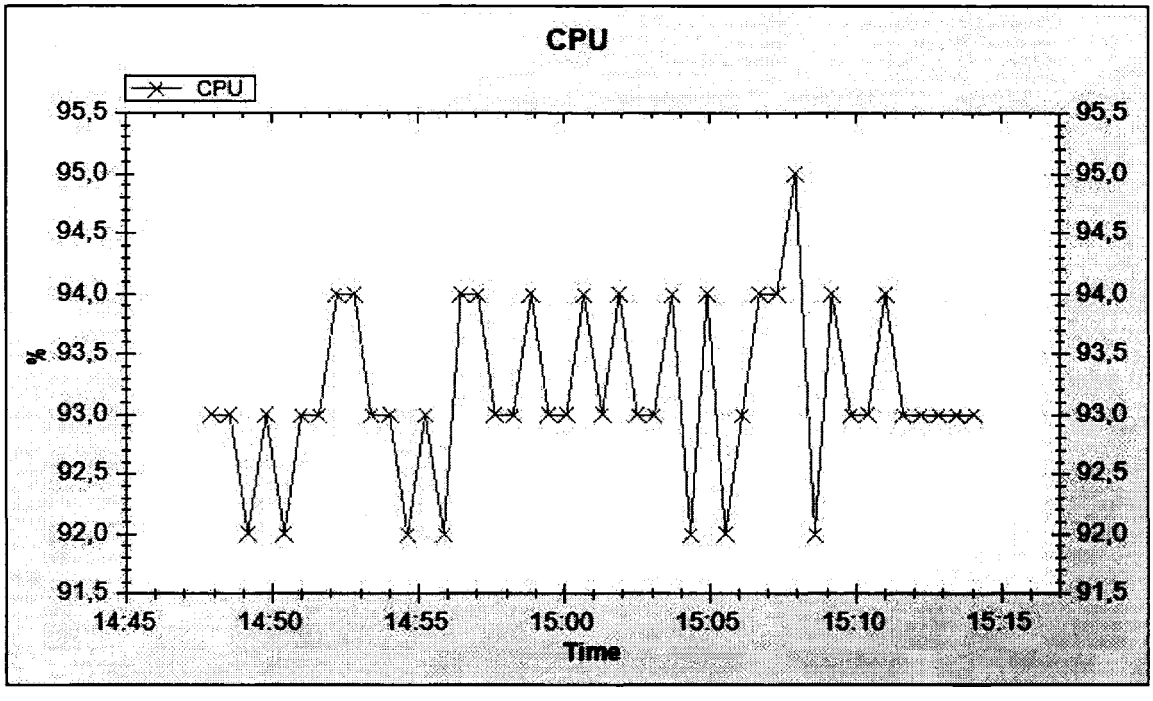


Figure 50 Pourcentage d'utilisation du CPU obtenu avec QMA

5.3.2 Class-Map

Une des statistiques les plus intéressantes à visualiser lorsqu'on surveille les mécanismes de QdS est sans nul doute les statistiques nommées *class-map*. Ces statistiques sont de

très bons indicateurs du niveau d'utilisation d'une classe. Elle peut, en effet, donner le débit de trafic transmis et le débit de trafic rejeté. La Figure 51 et la Figure 52, présentent les résultats de ces statistiques obtenus par la commande ***policy-map interface f0/0.50*** exécutée dans le CLI, puis avec le système QMA respectivement.

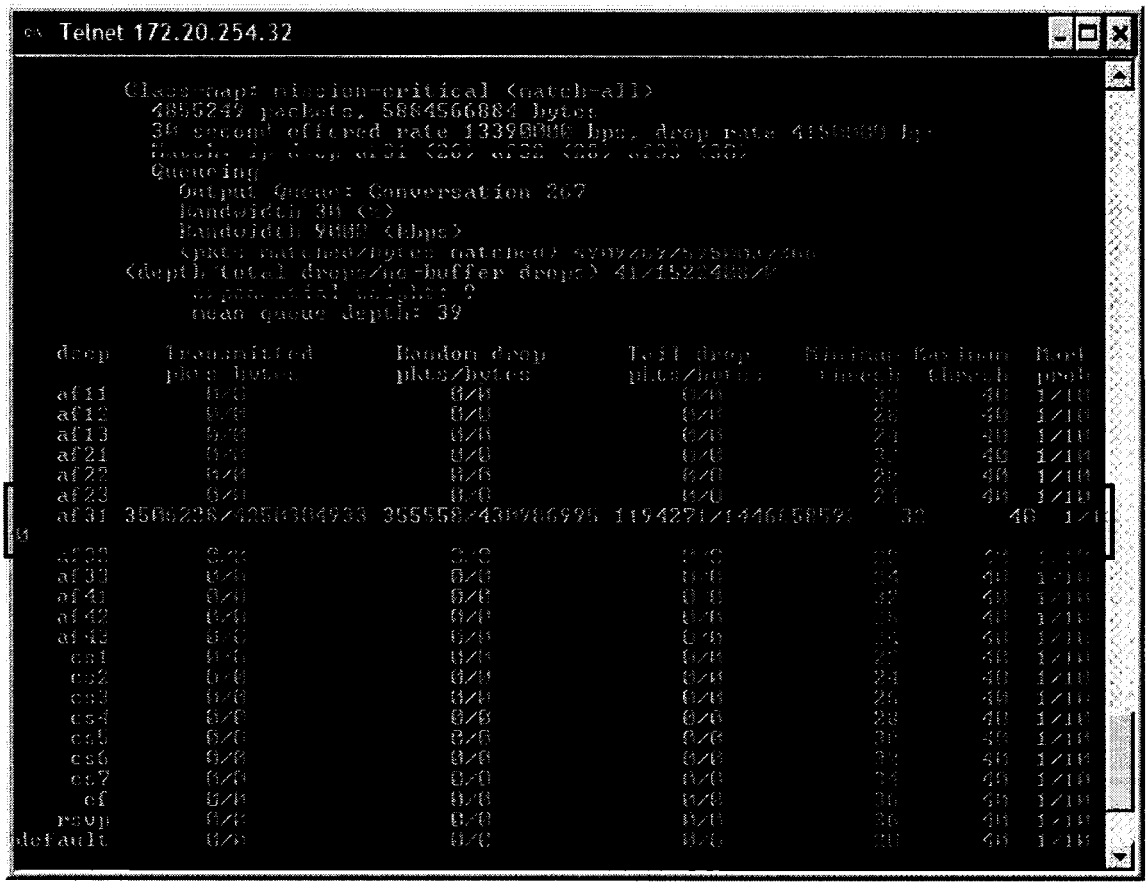


Figure 51 Statistiques de *class-map*, de *queueing* et de RED obtenues via une commande dans le CLI

En considérant la classe *mission-critical*, 13.5Mbps y ont été injectés (voir Tableau XVIII) alors que cette classe ne peut en transmettre que 9Mbps (30% de 30Mbps). Par conséquent une moyenne de 4.5Mbps (13.5Mpbs – 9Mbps) est rejetée. En regardant la Figure 51, on peut voir qu'en moyenne 13.39Mbps ont été reçus dans cette classe alors que 4.15Mbps ont été jetés. Ainsi, 9.24Mbps ont pu être transmis. Notez qu'une partie

des 240kbps supplémentaires transmis, provient du fait que du *policing* est appliqué sur la classe *scavenger* à un taux de 1Mbps. Étant donné que cette classe est autorisée à transmettre 1.5Mbps (5% de 30Mbps) par la politique *out-ets1-parent*, les 500kbps non utilisés de cette classe sont répartis aux autres classes selon le poids de chacune.

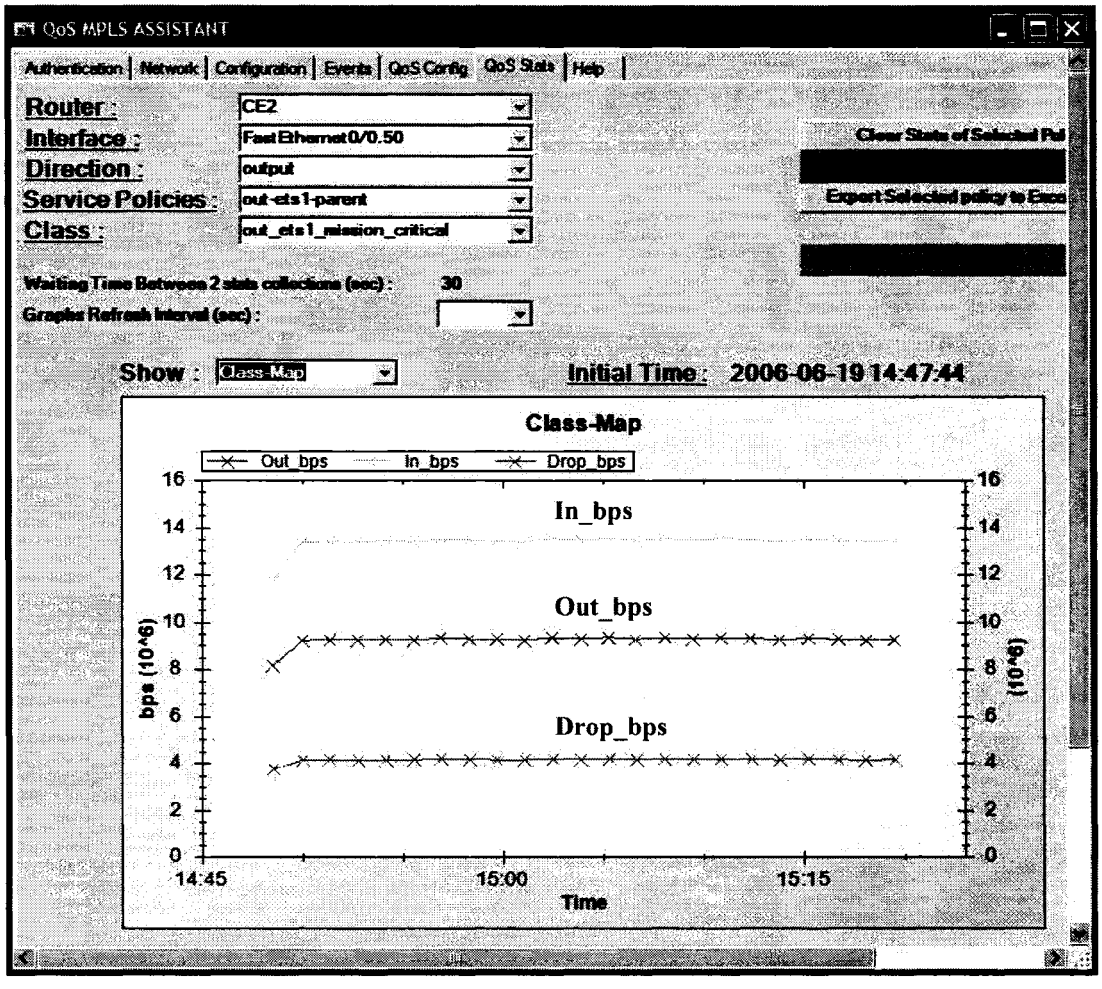


Figure 52 Statistiques de *class-map* obtenues avec QMA

La Figure 52 présente les résultats obtenus avec le système QMA et montre, du fait même, une très forte cohérence avec les résultats obtenus par la commande du CLI. Ainsi, on peut voir qu'environ 13Mbps entrait dans la classe alors qu'environ 9Mbps ont été transmis et qu'un peu plus de 4Mbps ont été rejetés.

5.3.3 Queueing

Afin de pouvoir visualiser le nombre de paquets dans une file d'attente ainsi que le nombre de paquets rejetés, les statistiques de *queueing* sont prélevées. Ces statistiques peuvent être obtenues en utilisant la commande du CLI spécifiée dans la section 5.3.2. On peut donc voir, dans la Figure 51, qu'au moment où la commande a été exécutée, 41 paquets se trouvaient dans la file d'attente et qu'un total de 1 522 408 paquets ont été rejetés.

La Figure 53 présente l'historique de ces statistiques. On peut voir sur ce graphique que deux échelles différentes y sont présentes. Celle de gauche est utilisée pour la statistique *DiscardPkt_pkt* tandis que celle de droite est utilisée pour les deux autres statistiques soit *CurrentQDepth_pkt* et *MaxQDepth_pkt*. On peut noter que la statistique *CurrentQDepth* varie entre 40 et 42 paquets ce qui est cohérent avec le résultat cité précédemment.

Concernant la statistique *DiscardPkt_pkt*, on voit qu'elle est en constante augmentation. Ceci s'explique par le fait que le générateur transmettait des paquets de façon continue. Par conséquent, il y avait constamment des paquets rejetés. Advenant le cas où il n'y aurait plus eu de rejet durant une certaine période, un plateau aurait alors été affiché sur le graphique durant cette période. On remarque également que la courbe stoppe à une valeur d'environ 1 000 000 paquets rejetés ce qui laisse un écart d'environ 400 000 paquets obtenus par la commande du CLI. Cet écart s'explique par le fait que la commande du CLI a été effectuée quelques minutes après l'arrêt de la récolte de statistiques. De plus, comme le générateur a continué de fonctionner durant un certain temps après l'arrêt de la récolte de statistiques, un écart s'est creusé entre les valeurs présentées à la Figure 51 et à la Figure 53.

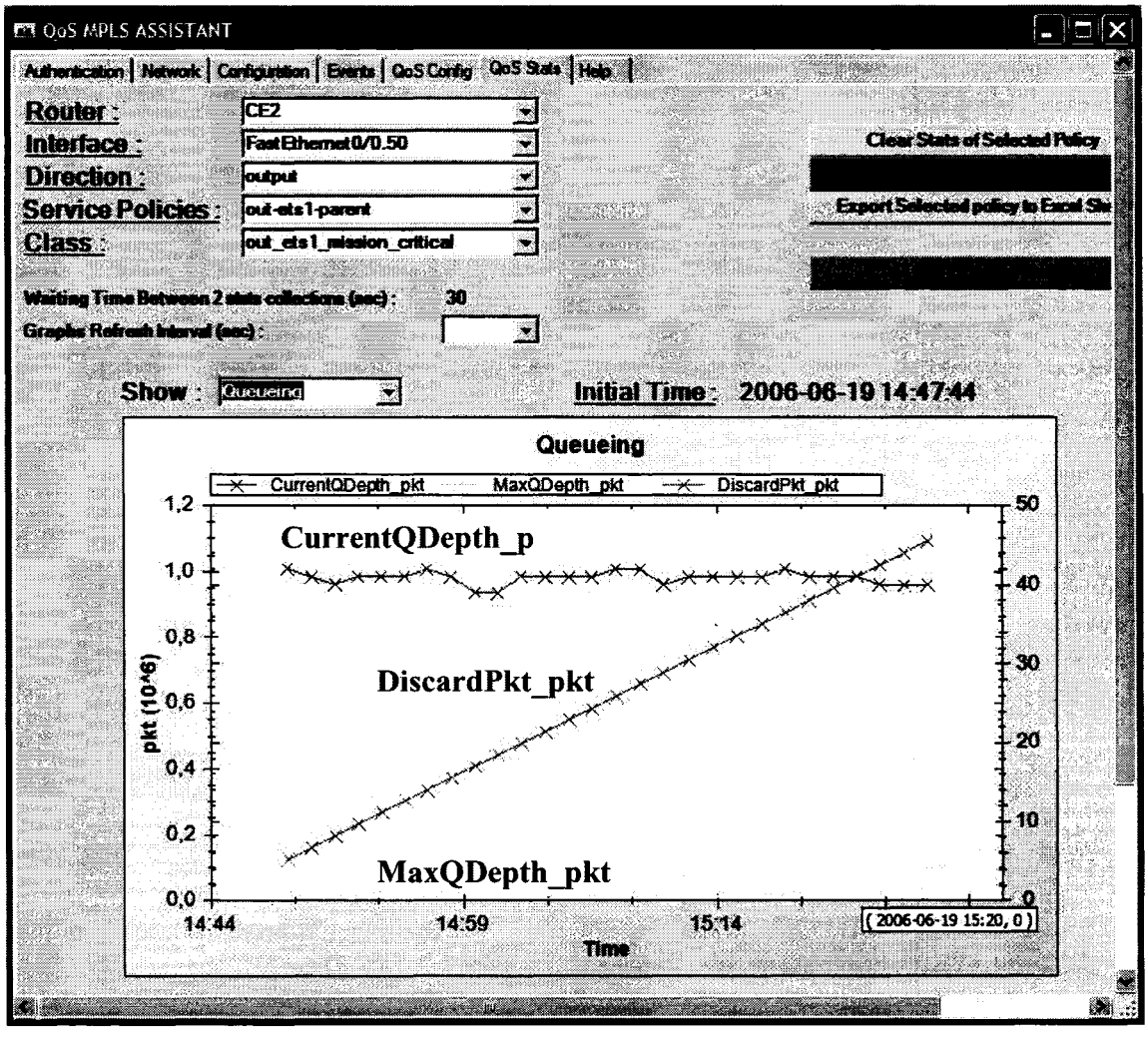


Figure 53 Statistiques de *Queueing* obtenues avec QMA

Finalement, la légende de la Figure 53 montre qu'il y a une autre statistique nommée *MaxQDepth_pkt* qui était supposée indiquer le nombre maximum de paquets qu'il y a eu, à un certain moment, dans la file d'attente. Par conséquent, cette statistique affiche la valeur maximale que la statistique *CurrentQSize_pkt* a atteint à un certain moment. Cependant on voit bien, dans la figure, que cette valeur est constamment nulle. Après investigation, on peut affirmer que la valeur retournée par l'agent SNMP est bien nulle et que le système QMA n'est pas en cause. De plus, la pertinence de cette statistique est

remise en question puisqu'il n'est pas très pratique de connaître la valeur maximale que *CurrentQSize_pkt* a atteint si on peut visualiser l'historique de cette dernière.

5.3.4 Lissage (*Shaping*)

Les statistiques de lissage sont pratiques pour un opérateur réseau qui désire savoir quand le lissage a été activé et à quelle fréquence il l'est. La Figure 54 présente les statistiques obtenues à l'aide de la commande ***show policy-map interface f0/0.50*** dans le CLI tandis que la Figure 55 présente les statistiques obtenues avec QMA.

Dans la première figure, on peut voir certaines informations de configuration mais également des informations statistiques, par exemple le nombre de paquets dans la file de lissage, le nombre de paquets qui ont subi un délai ou encore une indication spécifiant si le lissage est actif ou non.

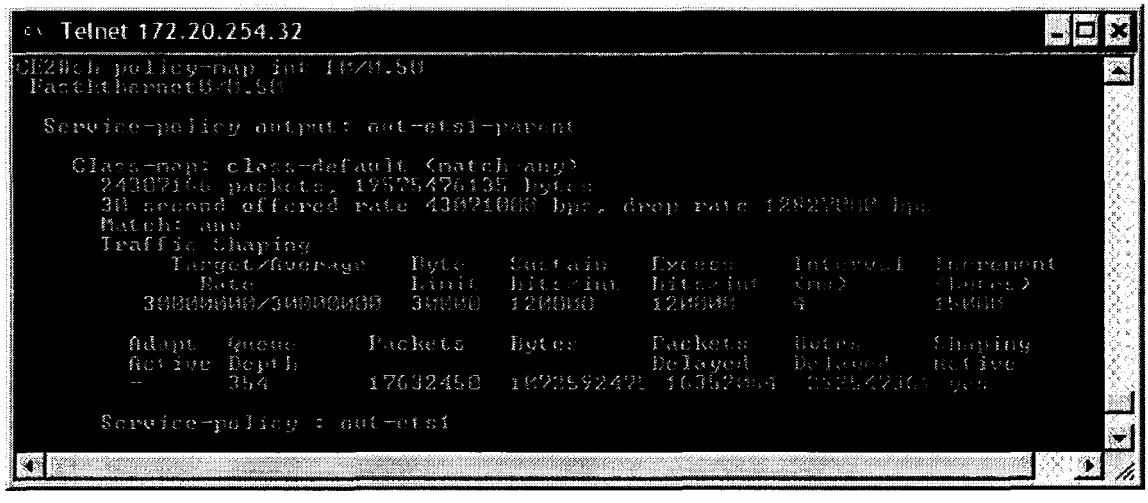


Figure 54 Statistiques de lissage obtenues à l'aide d'une commande dans le CLI

Tel que présenté dans la seconde figure, les informations, obtenues en utilisant la commande du CLI, peuvent également être visualisées avec le système QMA.

Cependant une information supplémentaire est disponible avec QMA. Ces informations sont les suivantes :

a. *IsActive* : Mis à 1 pour indiquer que le lissage est actif et à 2 pour indiquer le contraire.
b. *CurrentQSize_pkt* : Nombre de paquets présents dans la file d'attente de lissage.
c. *Delayed_pkt* : Nombre de paquets qui ont subi un délai dû à ce mécanisme.
d. *Drop_pkt* : Nombre de paquets rejetés.

La Figure 55 présente les résultats obtenus avec QMA. On peut affirmer que ces résultats sont cohérents avec ceux présentés à la Figure 54. Notez que pour les statistiques *CurrentQSize_pkt* et *IsActive*, l'échelle de droite est utilisée tandis que l'échelle de gauche est utilisée pour les deux statistiques restantes.

En regardant les résultats de la statistique *CurrentQSize_pkt*, on peut voir qu'il est, en moyenne, légèrement supérieur à 350 paquets ce qui correspond à la valeur obtenus par la commande du CLI.

Concernant la statistique *Delayed_pkt*, on remarque que la valeur obtenue avec la commande du CLI, soit 16 352 064 paquets, est supérieure à la dernière valeur obtenue lors de la collecte d'information avec QMA. Ceci est dû à la même raison évoquée dans la section 5.3.3, c'est-à-dire que la collecte de statistiques avec QMA avait été stoppée quelques minutes avant d'exécuter la commande du CLI et que le générateur de trafic était toujours en marche. Cependant cette valeur suit la tangente de la courbe obtenue avec QMA et n'affecte donc en rien la cohérence des résultats de QMA.

Quant aux résultats de la statistique *Drop_pkt*, ils ne peuvent être validés avec les résultats obtenus par la commande du CLI. Cependant on peut affirmer qu'ils sont cohérents puisqu'ils adoptent la tendance attendue, c'est-à-dire qu'ils sont en constante évolution.

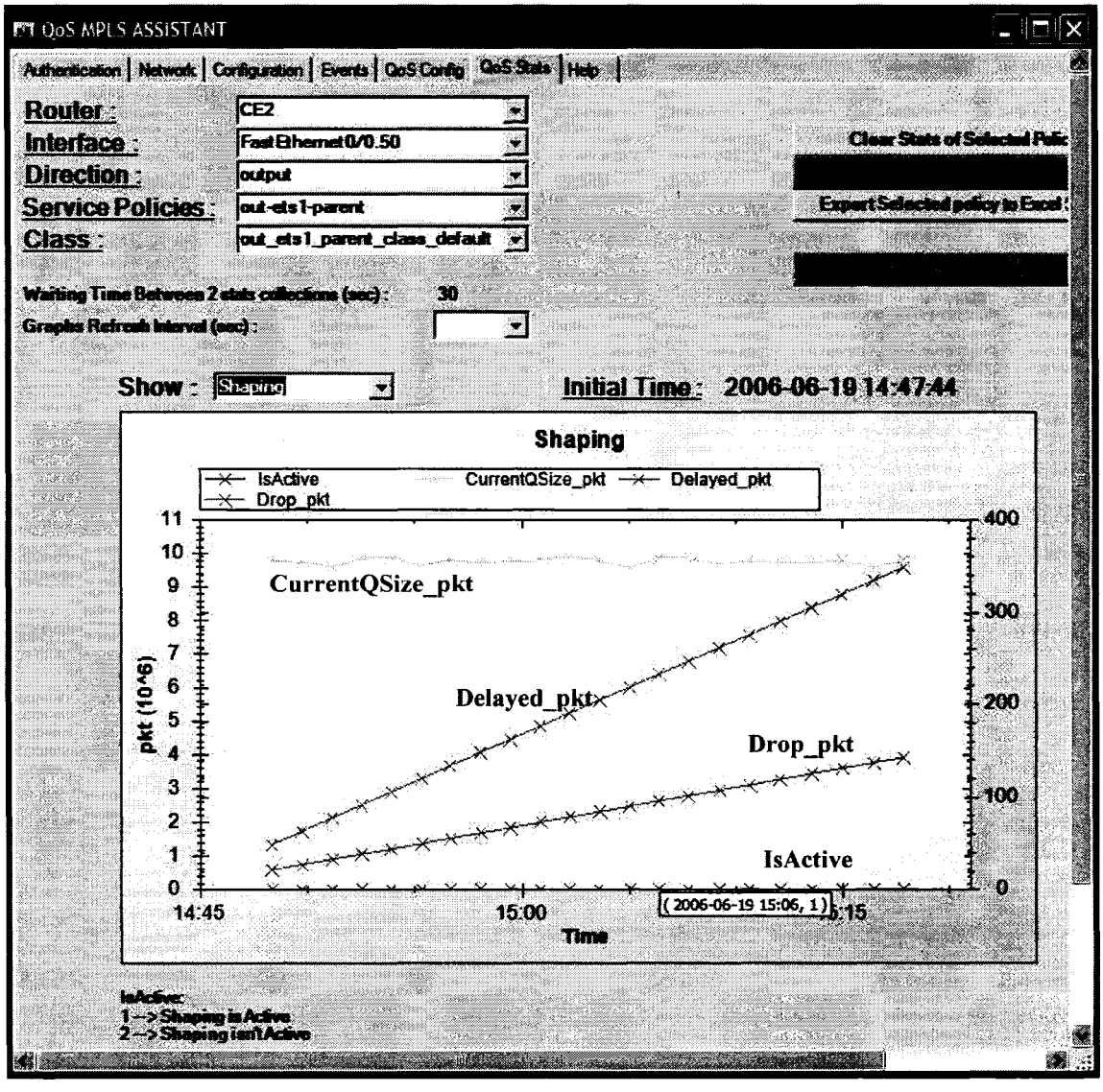


Figure 55 Statistiques de lissage obtenues avec QMA

Finalement, il est très difficile de voir la statistique *IsActive* puisqu'elle est affichée sur une échelle trop grande. En regardant la figure en bas à droite, on voit que la valeur d'un des points de cette courbe a été affichée (**2006-06-19 15:06, 1**). Cette valeur est tout à fait cohérente puisque le lissage était actif tout au long de la simulation.

5.3.5 Évitement de congestion (RED)

Les divers paramètres du mécanisme d'évitement de congestion sont très difficiles à définir adéquatement. Les statistiques de ce mécanisme peuvent grandement aider à cette tâche. En effet, un opérateur peut visualiser, pour une valeur DSCP ou *Precedence* spécifique, le nombre de paquet rejetés par ce mécanisme et ainsi ajuster les paramètres à sa guise. La Figure 51 montre les statistiques de ce mécanisme pour toutes les valeurs de PHB possibles. Étant donné que la classe *mission-critical* n'inclut que les paquets marqués AF31, toutes les autres valeurs de PHB sont nulles. On peut donc affirmer que l'usage de cette commande, via le CLI, pour obtenir des statistiques, montre des informations par défaut qui restent toujours nulles.

La Figure 56 montre, quant à elle, les statistiques du mécanisme d'évitement de congestion obtenues avec QMA. Pour chaque valeur DSCP ou *Precedence*, inclue dans les *match statement* d'une classe, le nombre de paquets rejetés par ce mécanisme (*RdmDrop_pkt*) et le nombre de paquets rejetés dû à un débordement de la file d'attente (*TailDrop_pkt*) sont affichés.

Prenons l'exemple présenté à cette figure. Sachant que la classe *mission-critical* inclut les paquets marqués AF31, AF32 et AF33, les statistiques sont affichées pour ces trois valeurs DSCP uniquement. Comme le générateur ne transmettait que du trafic marqué DSCP AF31, les statistiques pour les deux autres marques sont restées nulles. On voit dans cette figure que les deux courbes, associées à la marque AF31, sont en constante augmentation et terminent leur course à approximativement 750 000 paquets pour la statistique *TailDrop_pkt* tandis que la statistique *RdmDrop_pkt* termine sa course à environ 225 000 paquets. En regardant les résultats obtenus à la Figure 51, on remarque que les rejets sont de 1 194 271 et de 355 558 paquets pour les statistiques *RdmDrop* et *TailDrop* respectivement. L'écart entre les deux résultats s'explique par la même raison évoquée dans les sections 5.3.3 et 5.3.4. C'est-à-dire que la récolte de statistiques avec

QMA avait été arrêtée quelques minutes avant d'effectuer la commande dans le CLI et parce que le générateur était toujours en fonction durant cette période de temps. Cela dit, les résultats obtenus avec QMA restent tout à fait cohérents.

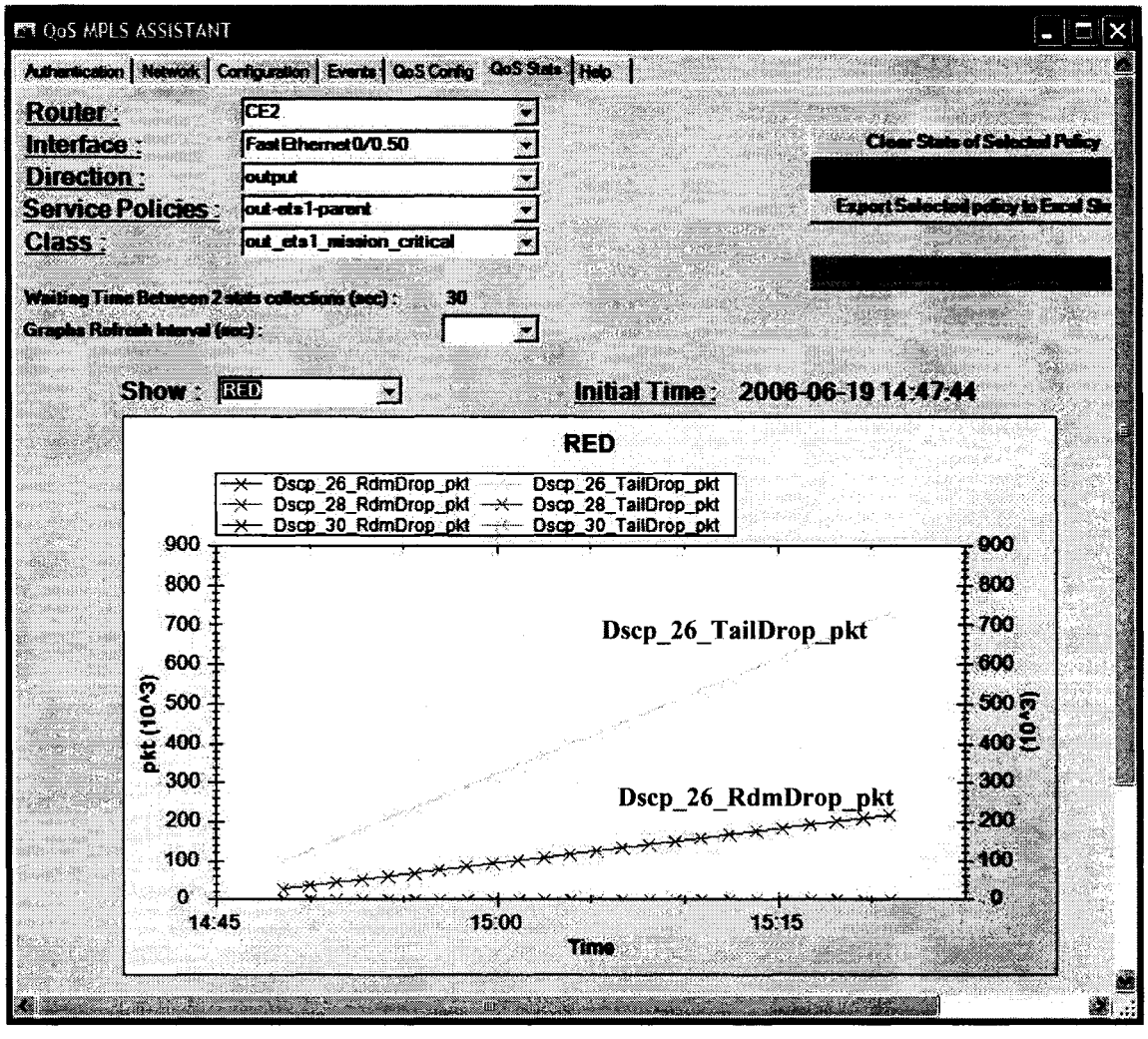


Figure 56 Statistiques du mécanisme d'évitement de congestion (RED) obtenues avec QMA

5.3.6 *Policing*

Pour un opérateur réseau, il s'avère très pratique de connaître les statistiques de *policing*, sous forme d'historique, afin de connaître la quantité de trafic qui a subie les actions *conform*, *exceed* et *violate*. Ainsi il sera plus en mesure de reconnaître si la spécification de ses paramètres de *policing* est adéquate.

La Figure 57 présente le résultat obtenu à l'aide de la commande ***show policy-map interface f0/1.521*** exécutée dans le CLI. Étant donné que 2.25Mbps ont été injectés dans la classe *scavenger* et que le *policing* était appliqué à cette classe pour un CIR de 1Mbps, ont peut conclure que 1Mbps devrait être transmis tandis que 1.25Mbps devrait être rejeté. Dans cette figure, on peut voir que 2.216Mbps ont été reçus par la classe alors que 1.003Mbps (*Conform_Action*) ont été transmis et que 1.215Mbps ont été rejetés (*Violate_Action*).

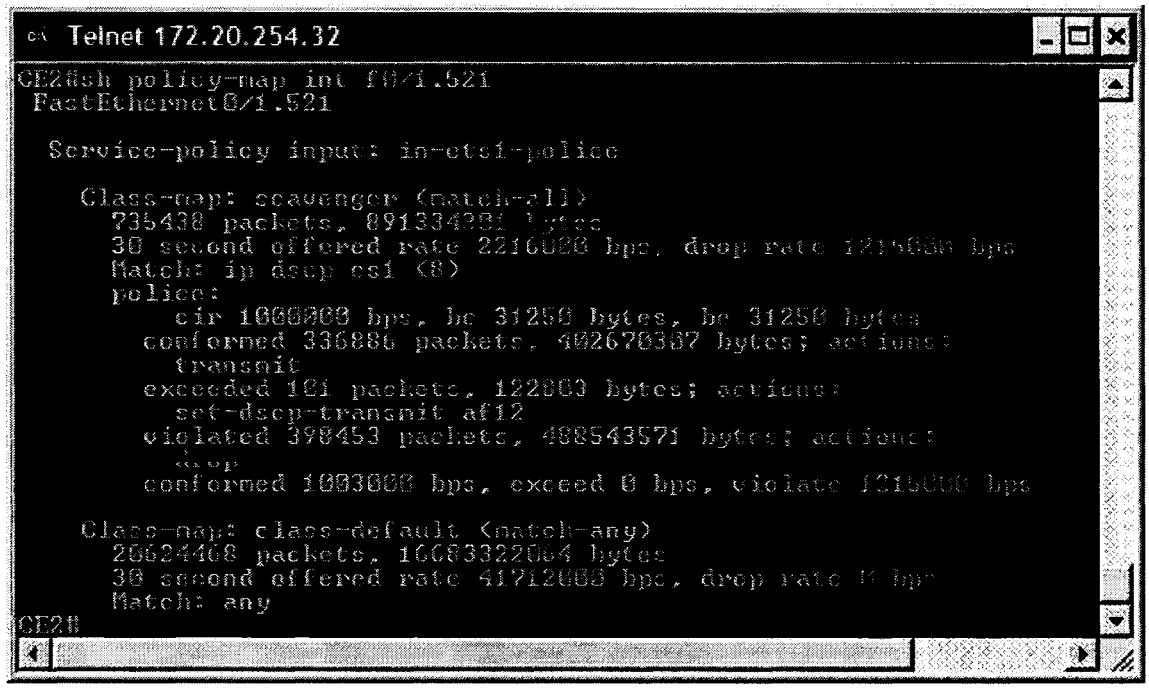


Figure 57 Statistiques du *policing* obtenues à l'aide d'une commande dans le CLI

De façon similaire, la Figure 58 montre les résultats obtenus avec QMA. On peut y voir qu'environ 1Mbps ont été associés aux actions du trafic conforme (*Conform_bps*) tandis qu'un peu moins de 1.225Mbps, ont été associés aux actions du trafic violant les spécifications (*Violate_bps*).

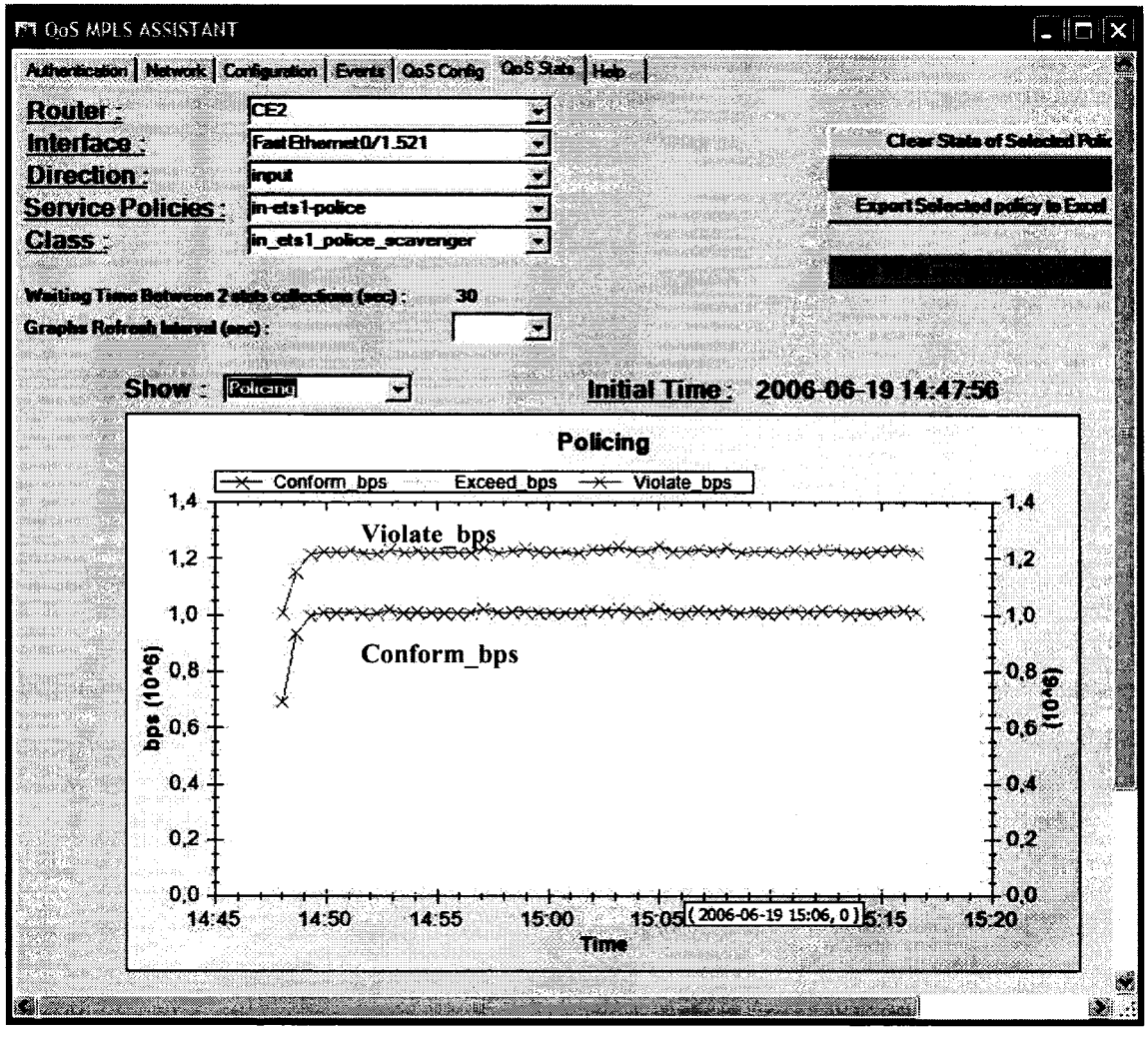


Figure 58 Statistiques du *policing* obtenues avec QMA

Dans les deux figures on remarque que 0Mbps est associé à l'action pour le trafic en excès. En référant à la section 2.2.6, on peut comprendre que cette action n'est applicable que lorsqu'il y a eu une certaine période d'inactivité. Dans notre scénario,

comme nous transmettons en continu, il n'y a pas de période d'inactivité et par conséquent, il n'y a aucun trafic associé à la statistique *Exceed_bps*. Les résultats obtenus à l'aide de QMA sont donc tout à fait cohérents.

5.3.7 Suppression de statistiques

Cette section a pour but de valider le fonctionnement des boutons utilisés pour supprimer les statistiques. La Figure 59 présente deux actions posées.

Dans la partie supérieure de la figure, on peut voir que deux politiques de service sont sous surveillance. Par la suite, l'utilisateur a sélectionné la politique *in-ets1-police* puis a cliqué sur le bouton *Clear Stats of Selected Policy* ce qui a supprimé les statistiques de cette politique tel qu'il peut être vu dans la partie au centre de la figure. Finalement l'utilisateur a cliqué sur le bouton *Clear All Statistics*, ce qui a supprimé les statistiques de toutes les politiques de service qui étaient toujours en train d'être surveillées (voir partie inférieure de la figure).

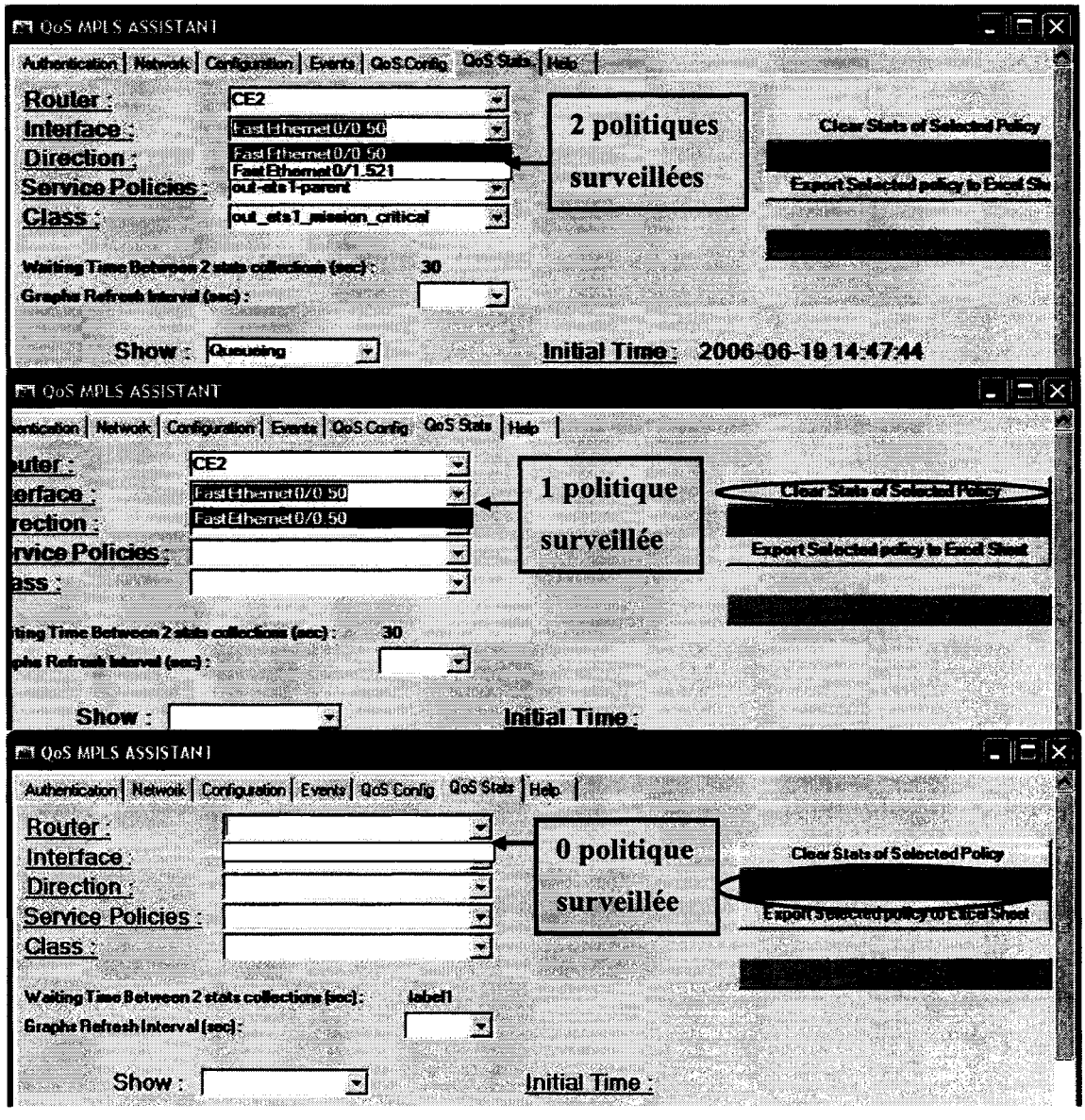


Figure 59 Validation du fonctionnement des boutons de suppression de statistiques

Pour terminer, ce chapitre a validé les principales fonctionnalités de QdS pouvant être effectuées par un utilisateur. On a d'abord validé le fonctionnement de l'option *ReNew QoS Configurations*, puis l'obtention d'une configuration d'une politique de service et finalement l'obtention de statistiques pour chacun des mécanismes de QdS. Pour cette dernière validation, un scénario de test a été effectué dans les laboratoires de Bell

Canada. Finalement, l'obtention des configurations et des statistiques ont été validés par une série de commandes effectuées dans le CLI de l'équipement concerné.

Tel que précisé dans le CHAPITRE 4, le but de ce projet consiste à développer une architecture de base permettant la visualisation des mécanismes de QdS. Par conséquent, seules certaines spécifications fonctionnelles ont été implémentées. Parmi elles, on retrouve la visualisation des configurations et des statistiques des mécanismes de QdS. Ainsi les spécifications fonctionnelles concernant la configuration semi-automatique et la génération d'alertes et de rapports n'ont pas été implémentées. Cependant, ces spécifications non-implémentées sont incluses dans la liste de recommandations à effectuer pour la poursuite du projet.

Pour terminer, certaines spécifications non-fonctionnelles ont été atteintes concernant entre autres l'utilisation du système ainsi que le support technologique. Cependant, il est très important de prendre en considération toutes les spécifications dans le développement future du système et ce, dans le but d'avoir un système plus performant et fiable. À ce sujet, une recommandation a été donnée (voir section RECOMMANDATIONS) quant à une nouvelle architecture dont le but est d'améliorer les performances globales du système.

CONCLUSION

Ce mémoire a été réalisé afin de décrire le développement du système de surveillance QMA réalisé dans le cadre d'un projet avec l'industrie. Dans un premier temps, une mise en contexte a été réalisée en définissant d'abord la problématique résolue par le système QMA. Dans un deuxième temps, tout le contexte entourant QMA a été développé en décrivant d'abord l'architecture des NGN, le rôle de QMA a pu être explicité dans cette architecture. Le développement réalisé dans le cadre de ce mémoire consiste en une architecture de base dont le but final vise le rôle du *Bandwidth Manager* de l'architecture de référence des NGN. Étant donné que le principal rôle de QMA, dans le contexte de ce mémoire, est de surveiller les mécanismes de QdS, ceux-ci ont été étudiés en détail dans le CHAPITRE 2. Par ailleurs, ce chapitre a également présenté une étude des principaux protocoles permettant la surveillance d'un réseau.

Dans un troisième temps, les spécifications fonctionnelles et non fonctionnelles de QMA ont été énumérées. Ainsi les principales fonctionnalités du système ont été définies ainsi que les principales exigences permettant un système convivial et performant. Dans un quatrième temps, l'architecture du système a été décrite de façon générale, puis en détail. En effet, chaque composante de l'architecture de QMA a été décomposée afin de pouvoir avoir une vue plus précise de chacune d'elle. De plus, l'interaction entre chacune d'elle a également été présentée.

Finalement, le bon fonctionnement du système a été validé par une série de tests réalisés dans les laboratoires de Bell Canada. Ces essais montrent que QMA répond aux principales exigences pour lesquelles le système a été développé. Par ailleurs, quelques recommandations suivent cette conclusion, dont les buts visent une amélioration des performances et des fonctionnalités. Advenant le cas où ces nouvelles fonctionnalités étaient introduites dans QMA, ce dernier répondrait mieux au rôle du *Bandwidth*

Manager tel que défini dans l'architecture de référence des NGN. De plus, il deviendrait un outil grandement utile pour la recherche.

Ce mémoire a abouti à un outil pouvant être utilisé dans l'industrie afin de pouvoir mieux visualiser les configurations des mécanismes de QdS ainsi que pour visualiser l'utilisation de ceux-ci. Bien que QMA ait été initialement développé sur une plateforme de test contenant uniquement des équipements Cisco, l'architecture proposée permet facilement l'ajout d'autres équipementiers. Cet aspect constitue le principal avantage de QMA face aux autres systèmes pouvant effectuer une surveillance des mécanismes de QdS.

RECOMMANDATIONS

Cette section énumère une série de recommandations qui concernent l'amélioration du système ainsi que le développement qu'il peut prendre dans un cadre de recherches futures. Les améliorations concernent l'amélioration des performances du système ainsi que l'amélioration de ses fonctionnalités.

Aspects non implémentés

Le travail fait jusqu'à présent ne constitue qu'une architecture de base du système à laquelle d'autres fonctionnalités peuvent être greffées. Par conséquent, toutes les spécifications énumérées dans la section 3.2 n'ont pas été réalisées. Afin d'obtenir un outil complet, ces spécifications doivent toutes être implémentées. Ainsi le système permettra d'offrir plus de fonctionnalité à l'utilisateur en lui permettant de configurer ou de modifier une politique de service par le biais de QMA, d'obtenir des alertes lorsque certains événements surviennent ainsi que d'obtenir des rapports de performance et d'alertes sur une base régulière. De plus, les performances générales du système seront améliorées et maintenues à un seuil acceptable. Finalement, l'utilisation du système sera facilitée par le biais d'aide à l'utilisateur.

De plus, les configurations et statistiques concernant le marquage n'ont pu être obtenues sur les équipements Cisco. En fait, ces derniers comportent une erreur dans l'IOS qui rend intraitable les OID correspondants au marquage. Par conséquent, lorsque cette erreur sera réglée, il sera alors indispensable de traiter ce mécanisme dans QMA.

Limitations actuelles du système

Un aspect très important à considérer, lors du développement d'un outil de surveillance centralisé comme QMA, est la limite du système en termes d'équipements pouvant être

surveillés en concurrence. Lorsque le système est surchargé, une dégradation des performances générales du système peut se faire ressentir. Lorsque les performances du système se dégradent, le temps de réponse augmente pour les utilisateurs. Dans le cadre de ce mémoire, aucun test de performance n'a été réalisé à cet effet. Il s'avère donc primordial de déterminer cette limitation afin de préserver un niveau de performance acceptable.

Une autre limitation contraignante du système actuel se rapporte à l'affichage de la topologie du réseau. Les équipements affichés sont présentement insérés directement dans le code et le système ne permet pas d'en ajouter un de façon dynamique. Il serait donc intéressant d'offrir une fonctionnalité à l'utilisateur afin qu'il puisse facilement ajouter un nouvel équipement dans l'interface graphique. De plus, la taille du schéma du réseau ne permet pas de pouvoir visualiser un réseau de grande dimension. Il est donc intéressant de modifier cette interface afin de pouvoir visualiser l'ensemble du réseau d'un fournisseur de service. L'approche adoptée par OPNET Technologies inc. (http://www.opnet.com), en utilisant des icônes de sous-réseau, est une solution à envisager.

Proposition d'une architecture pour améliorer les performances globales

La Figure 60 ci-dessous présente une architecture générale pouvant être développée dans le but d'améliorer les performances globales du système. Cette nouvelle architecture permettra l'amélioration des performances du système lorsque celui-ci est utilisé au sein d'un grand réseau comme celui d'un fournisseur de service. Cette architecture comporte plusieurs similarités avec l'architecture présentée dans la Figure 14. Ainsi le passage à la nouvelle architecture générale sera facilité.

D'abord les composantes *ModulesQMA*, *Contrôleur de serveur* ainsi que la base de données sont toutes regroupées dans une entité nommée *QMA Device* dans l'architecture

de la Figure 60. Cette entité est chargée de surveiller sa portion de réseau et de conserver les statistiques dans sa propre base de données. Elle est également chargée de conserver les informations SNMP de chaque équipement, sous sa supervision, de sorte qu'il puisse communiquer avec eux.

Par la suite, l'entité nommée *QMA Device Manager* est chargée de corréler toutes les informations obtenues des *QMA Device* afin qu'un utilisateur puisse avoir une vision globale de l'ensemble du réseau. Il est donc responsable de conserver, dans sa propre base de données, certaines informations des *QMA Device*. Comme par exemple, ces informations doivent être constituées d'une liste des *QMA Device*. Pour chacun d'eux, il doit conserver une liste des équipements qu'un *QMA Device* a le devoir de surveiller. De plus, il doit conserver les informations d'authentification des usagers.

Le service Web est utilisé uniquement comme interface entre l'utilisateur et le *QMA Device Manager*. Si l'utilisateur initie certaines commandes, le *QMA Device Manager* ou encore les *QMA Device* seront en charge de les exécuter.

Finalement, la communication entre le service Web et le *QMA Device Manager*, ainsi qu'entre ce dernier et les *QMA Device*, doit être réalisée par un protocole permettant un échange sécuritaire de l'information. Cette communication doit être réalisée de sorte que les informations échangées ne puissent être interceptées et/ou comprises par une entité non autorisée. Ces communications sont représentées par la notation *TBD* (*To Be Determined*) et peuvent être réalisées par un protocole comme SSH.

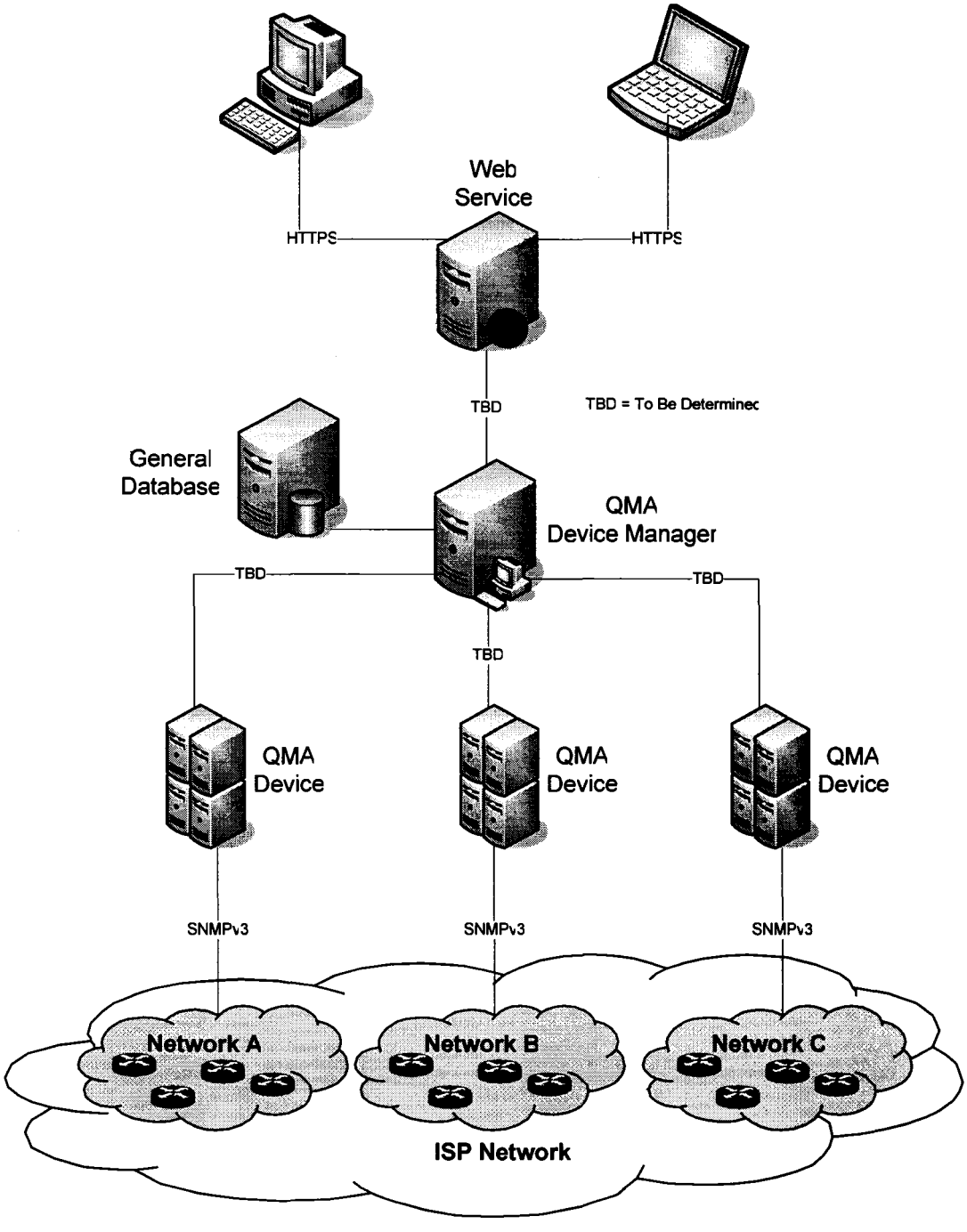


Figure 60 Architecture proposée pour améliorer les performances du système

Envoi de commandes aux équipements réseaux

Sachant qu'il est possible de se connecter à un équipement réseau par le biais d'une connexion sécurisée SSH, il serait intéressant de permettre au système d'envoyer des commandes à ceux-ci lorsque requis. Cette évolution permettra au fournisseur une plus grande flexibilité dans les services offerts à ses clients.

Comme par exemple, un client pourra se connecter sur le site Web du fournisseur de service et définir lui-même ses classes de services et la bande passante allouée à chacune d'elle. Le système implémentant cette solution pourra guider le client pas à pas dans la définition de ses services. Une fois la demande du client terminée, si le système l'accepte, les équipements réseaux pourront être configurés automatiquement de façon à répondre aux nouvelles spécifications du client. Il pourra par la suite recevoir un rapport hebdomadaire lui permettant de prendre une décision plus éclairée à l'égard des services réseaux pour lesquels il paie. Ceci n'est qu'un exemple utilisé pour montrer qu'un tel système permettra une plus grande flexibilité dans le développement de nouveaux services.

De plus, comme les configurations de QdS sont réalisées de façon statique par un administrateur réseau, un tel système permettra un plus grand dynamisme dans les configurations de QdS et rendra, du fait même, moins utile des algorithmes de QdS dynamiques tel que proposé dans [36]. De plus, en ayant une vue globale du réseau et en pouvant interagir sur celui-ci, ce système deviendra alors un outil très pratique pour la recherche future.

Ainsi certaines opérations pourront être automatisées par le système de façon à minimiser les interventions des opérateurs réseau. De plus, l'envoi de commandes à un équipement ouvre la porte à des recherches futures dont un des buts consistera à automatiser les configurations lorsque certains événements surviennent dans le réseau.

Par exemple, une fois cet ajout de fonctionnalité terminé, QMA sera beaucoup plus en mesure d'exécuter le rôle du *Bandwidth Manager* de l'architecture des NGN. Il ne suffira alors qu'à implémenter l'interface entre QMA et un gestionnaire d'appel (*Call Manager*) SIP.

ANNEXE 1

Principaux OID de la MIB de QdS de Cisco Systems inc.

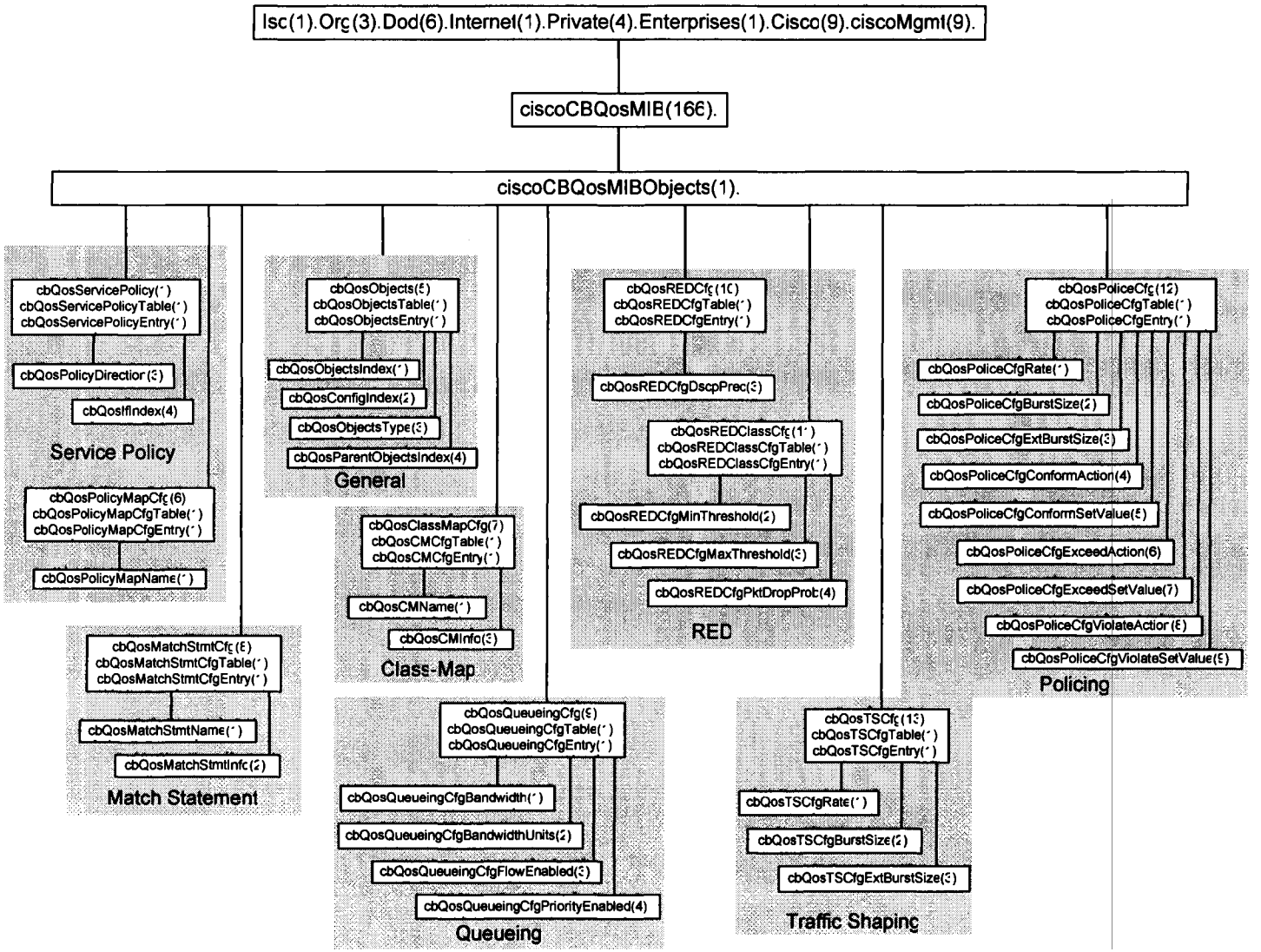


Figure 61 OID principaux utilisés pour l'obtention des configurations de QdS dans les routeurs Cisco

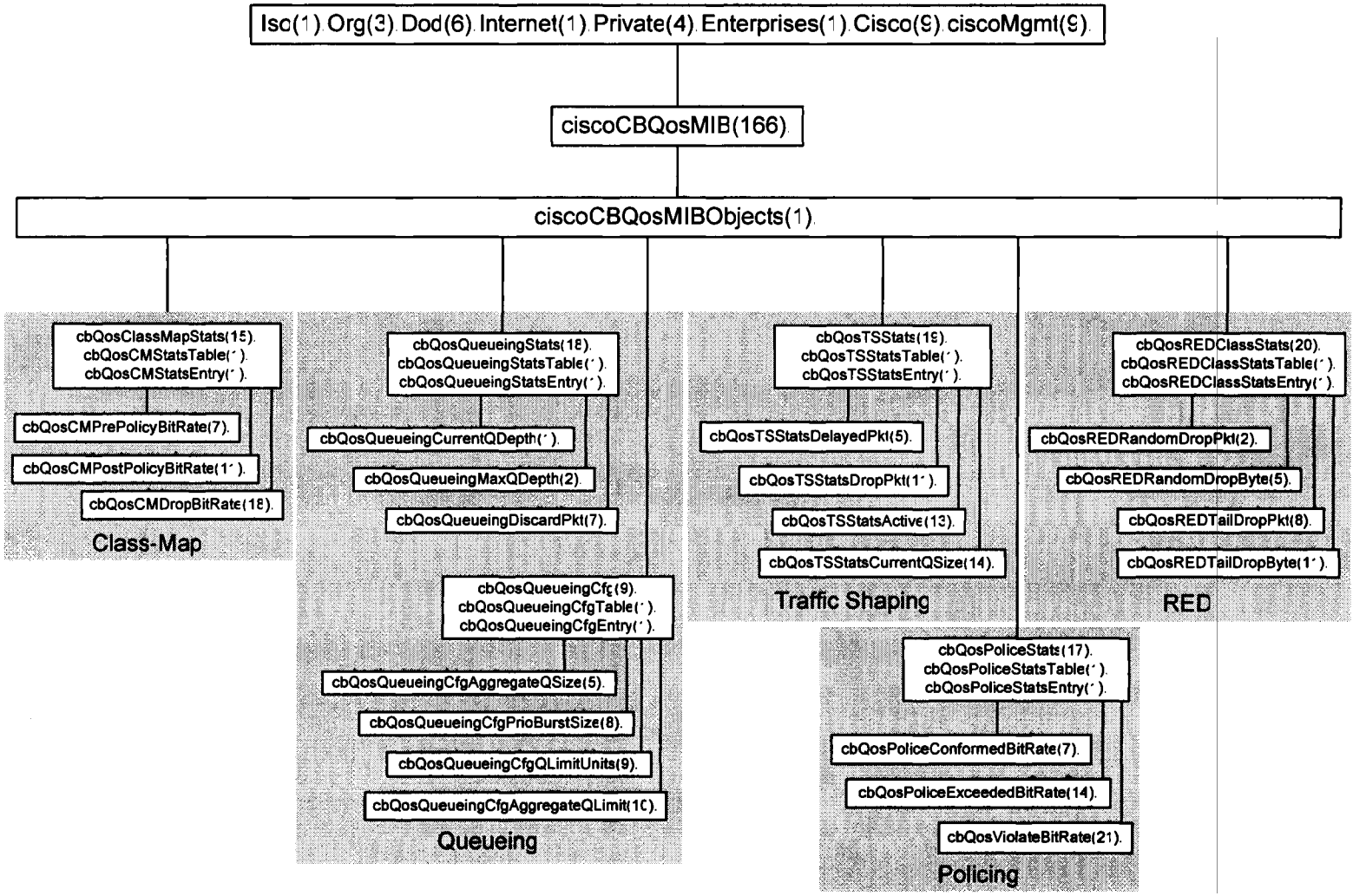


Figure 62 OID principaux utilisés pour l'obtention des statistiques des mécanismes de QdS dans les routeurs Cisco

ANNEXE 2

Description des bases de données

Cette annexe présentera les diverses bases de données et tables utilisées dans ce projet. Cependant, seules les tables nécessaires aux fonctionnalités de QdS seront présentées. Dans un premier temps, la base de données nécessaire à l'authentification sera présentée. Par la suite, la base de données de configuration sera détaillée. Finalement, la base de données de statistiques sera décrite.

Base de données d'authentification (QMA_Auth)

Tel que mentionné dans l'introduction de cette annexe, cette section présente les diverses tables de la base de données d'authentification. La Figure 63 présente les diverses tables et colonnes les composants.

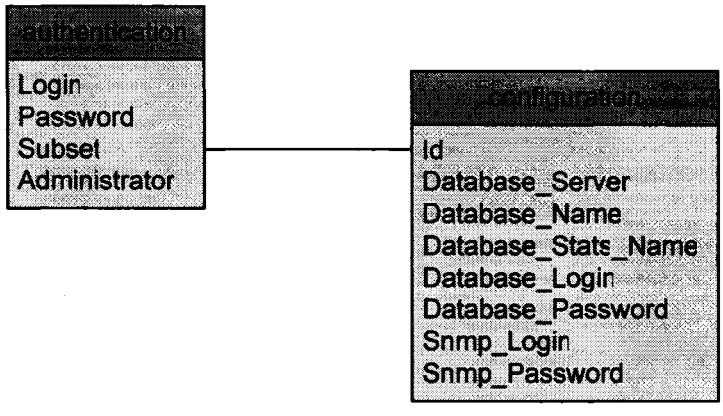


Figure 63 Base de données d'authentification

Dans cette figure, on peut voir que la base de données est composée de deux tables soit *authentification* et *configuration*. La table *authentification* est utilisée afin de conserver la liste des noms d'usager (colonne *Login*) ainsi que leur mot de passe haché (colonne *Password*). La colonne *Administrator* indique si l'usager correspondant est un administrateur. Lorsqu'un usager est un administrateur, il aura le droit d'ajouter ou de supprimer des usagers de cette liste. Finalement, la colonne *Subset* est utilisée afin de référer aux configurations correspondantes dans la table *configuration*. Ainsi cette colonne est en relation avec la colonne *Id* de la table *configuration*.

La table *configuration* comporte les informations générales utilisée pour communiquer avec les bases de données de configuration et de statistiques en plus des informations requises pour interroger les routeurs. Par conséquent, les descriptions des diverses colonnes sont résumées dans les quelques points suivants :

a. ***Database_Server*** : Nom du serveur SQL.
b. ***Database_Name*** : Nom de la base de données de configuration.
c. ***Database_Stats_Name*** : Nom de la base de données de statistiques.
d. ***Database_Login*** : Nom d'usager utilisé pour communiquer avec les bases de données.
e. ***Database_Password*** : Mot de passe utilisé pour communiquer avec les bases de données.
f. ***Snmp_Login*** : Nom d'usager utilisé pour communiquer avec les équipements réseau via le protocole SNMP.
g. ***Snmp_Password*** : Mot de passe utilisé pour communiquer avec les équipements réseau via le protocole SNMP.

Base de données de configuration

Tel que mentionné dans l'introduction de cette annexe, cette section présente les diverses tables de la base de données de configuration ainsi qu'un organigramme utilisé pour aller chercher une configuration de QdS. La Figure 64 présente la structure des tables utilisées pour conserver les configurations de QdS.

D'abord la table *Router* est la première table qui doit être remplie puisque les autres tables utilisent certaines informations de cette dernière. Elle est constituée de cinq colonnes dont la liste et les descriptions sont citées ci-dessous. En résumé, cette table dresse une liste des équipements du réseau ainsi que de toutes leurs interfaces tant physiques que virtuelles.

a. ***IP :*** Conserver l'adresse IP de l'interface du routeur spécifié par cette entrée.
b. ***Node :*** Conserve le nom d'hôte du routeur.

c. ***Name :*** Conserve le nom de l'interface

d. ***Interface_Index :*** Conserve l'index de l'interface en question.

e. ***Lsr_id :*** Spécifie le numéro d'identification du *Label Switched Router* (LSR). Cette information est utilisée pour savoir quels routeurs font partie de la portion MPLS du réseau.

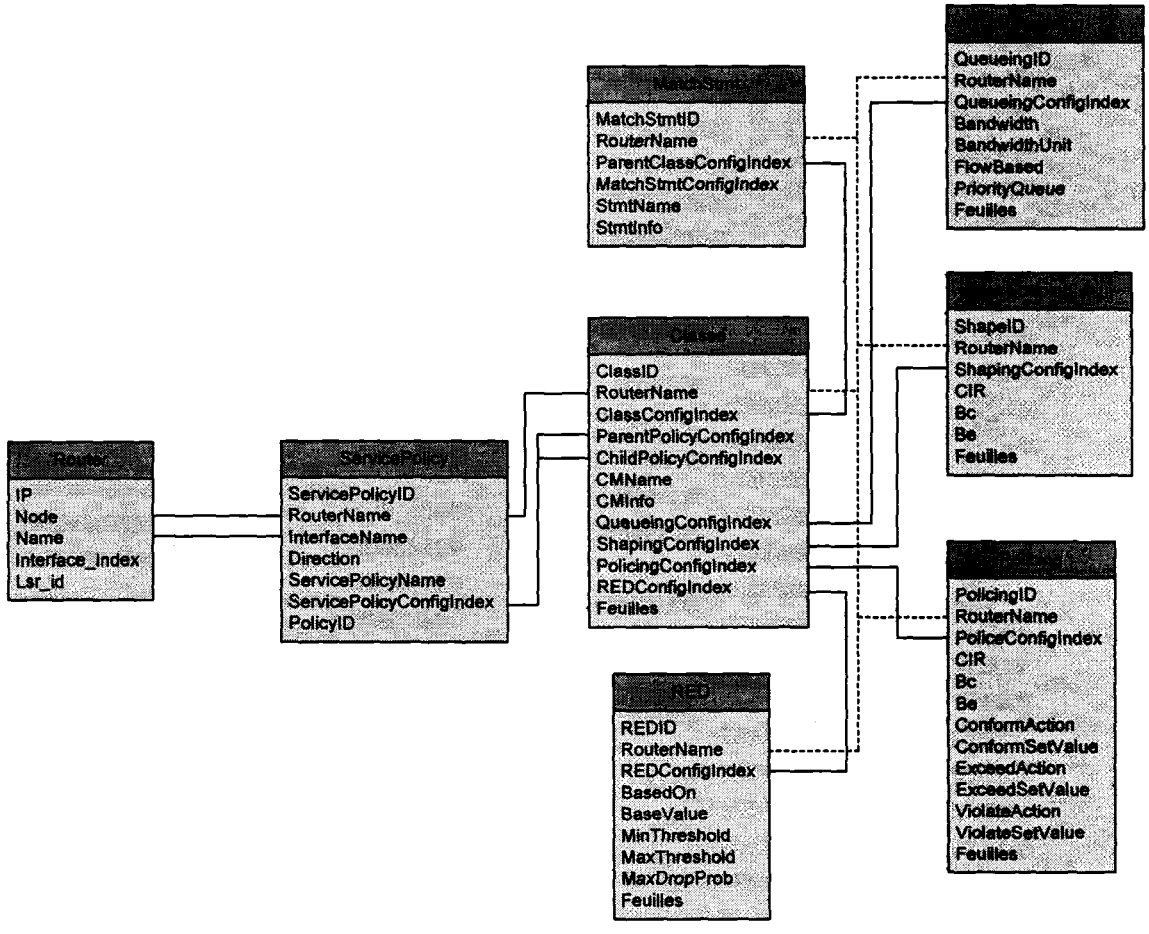


Figure 64 Base de données de configuration de le QdS

La table *ServicePolicy* est la table qui contient les informations générales d'une politique de service. Elle nécessite certaines entrées de la table *Router* dont le nom de l'équipement ainsi que le nom de l'interface. La liste et descriptions des colonnes sont citées ci-dessous. On y retrouve des informations générales à la politique de service

comme par exemple son nom ou encore la direction du trafic sur laquelle elle est appliquée. Cette table contient également des informations permettant de connaître sa configuration. C'est d'ailleurs le cas de la colonne *ServicePolicyConfigIndex* qui permet de connaître les relations parent/enfant entre la politique en question et les diverses classes.

a. ***ServicePolicyID :*** Identifie la politique de façon unique dans la table.
b. ***RouterName :*** Conserve le nom d'hôte de l'équipement sur lequel la politique de service est appliquée.
c. ***InterfaceName :*** Conserve le nom de l'interface de l'équipement en question sur laquelle la politique de service est appliquée.
d. ***Direction :*** Conserve la direction à laquelle la politique de service est appliquée.
e. ***ServicePolicyName :*** Conserve le nom de la politique de service.
f. ***ServicePolicyConfigIndex :*** Conserve l'index de configuration de la politique de service.
g. ***PolicyID :*** Conserve l'identificateur de la politique de service utilisé pour identifier cette dernière de façon unique dans l'équipement en question.

La table *Classe* est utilisée pour lister toutes les classes des politiques de service présentes dans la table *ServicePolicy*. Cette table renferme des informations générales de configuration de la classe en question (comme son nom ou son *match-mode*) mais renferme également des informations permettant de référer aux configurations des divers mécanismes de QdS configurés dans cette classe. La liste des colonnes et leurs descriptions sont listée ci-dessous.

a. ***classID :*** Identifie la classe de façon unique dans la table.
b. ***RouterName :*** Conserve le nom d'hôte de l'équipement sur lequel cette classe est configurée.
c. ***ClassConfigIndex :*** Conserve index de configuration de la classe.

d. ***ParentPolicyConfigIndex :*** Conserve l'index de configuration de la politique de service à laquelle cette classe appartient.
e. ***ChildPolicyConfigIndex :*** Dans le cas où la classe appelle une politique de service « enfant » (*hierarchical policy*), cette colonne conserve l'index de configuration de cette politique.
f. ***CMName :*** Conserve le nom de la classe.
g. ***CMInfo :*** Conserve le *match mode* de la classe.
h. ***QueueingConfigIndex :*** S'il y a lieu, conserve l'index de configuration du mécanisme d'ordonnancement.
i. ***ShapingConfigIndex :*** S'il y a lieu, conserve l'index de configuration du mécanisme de lissage.
j. ***PolicingConfigIndex :*** S'il y a lieu, conserve l'index de configuration du mécanisme de *policing*.
k. ***REDConfigIndex :*** S'il y a lieu, conserve l'index de configuration du mécanisme RED.
l. ***Feuilles :*** Conserve les feuilles de l'OID utilisé pour identifier cette classe, de façon unique, dans l'équipement en question. Ces feuilles seront utilisées pour la récolte de statistiques.

La table *MatchStmt* est utilisée pour lister tous les *match statememt* des classes présentent dans la table *Classe*. Notez que les *match statement* correspondent aux critères de classification d'une classe et débutent toujours par le mot clé « *match* ». On retrouve donc, dans cette table, les informations de configuration de ce *match statement* dont le *match statement* lui-même ainsi que le mode d'acceptation (*matchInfo*). De plus on y retrouve également les informations permettant d'identifier un *match statement* de façon unique dans tout le réseau (*RouterName* et *ParentClassConfigIndex*). La liste qui suit présente la liste et descriptions des colonnes de cette table.

a. ***MatchStmtID :*** Identifie un *match statement* de façon unique dans cette table.

b. ***RouterName :*** Conserve le nom d'hôte de l'équipement sur lequel ce *match statement* est appliqué.
c. ***ParentClassConfigIndex :*** Conserve l'index de configuration de la classe pour laquelle ce *match statement* a été configuré.
d. ***MatchStmtConfigIndex :*** Conserve l'index de configuration de ce *match statement.*
e. ***StmtName :*** Conserve le *match statement.*
f. ***StmtInfo :*** Identifie s'il s'agit d'un *match statement* d'acceptation (*none*) ou de rejet (*matchNot*) du trafic entrant cette classe.

La table *Queueing* est utilisée pour lister toutes les configurations d'ordonnancement présentes dans le réseau. Elle inclut des colonnes permettant d'identifier le type de mécanisme d'ordonnancement utilisé, la bande passante configurée ainsi que l'unité correspondant. De plus, elle comporte des entrées permettant d'identifier une configuration d'ordonnancement de façon unique dans le réseau. La liste des colonnes et leurs descriptions sont dressée ci-dessous.

a. ***QueueingID :*** Identifie une configuration d'ordonnancement de façon unique dans la table.
b. ***RouterName :*** Conserve le nom d'hôte de l'équipement sur lequel cette configuration d'ordonnancement a été appliquée.
c. ***QueueingConfigIndex :*** Conserve l'index de configuration d'une configuration d'ordonnancement.
d. ***Bandwidth :*** Conserve la quantité de bande passante configurée pour la classe à laquelle l'entrée de la table *Queueing* a été réalisée.
e. ***BandwidthUnit :*** Conserve l'unité de bande passante. Peut être soit un *percent* ou *kbps*.
f. ***FlowBased :*** Identifie si le type de mécanisme d'ordonnancement pour cette entrée est le WFQ ou non.

g. ***PriorityQueue :*** Identifie si le type de mécanisme d'ordonnancement pour cette entrée est le LLQ ou non.
h. ***Feuilles :*** Conserve les feuilles de l'OID utilisé pour identifier cette configuration d'ordonnancement, de façon unique, dans l'équipement en question. Ces feuilles seront utilisées pour la récolte de statistiques.

La table *Shaping* est utilisée pour lister toutes les configurations de lissage présentes dans le réseau. Elle comprend des colonnes permettant d'identifier une configuration de façon unique ainsi que des colonnes permettant de définir les divers paramètres de configuration. La liste des colonnes ainsi que leurs descriptions sont listés ci-dessous.

a. ***ShapeID :*** Identifie une configuration de façon unique dans la table.
b. ***RouterName :*** Conserve le nom d'hôte de l'équipement sur lequel cette configuration est appliquée.
c. ***ShapingConfigIndex :*** Conserve l'index de configuration des configurations de lissage.
d. ***CIR :*** Conserve la valeur du paramètre CIR d'une configuration de lissage.
e. ***Bc :*** Conserve la valeur du paramètre Bc d'une configuration de lissage.
f. ***Be :*** Conserve la valeur du paramètre Be d'une configuration de lissage.
g. ***Feuilles :*** Conserve les feuilles de l'OID utilisé pour identifier cette configuration de lissage, de façon unique, dans l'équipement en question. Ces feuilles seront utilisées pour la récolte de statistiques.

La table *Policing* est utilisée pour lister les configurations de *policing* présentes dans le réseau. Elle comporte des colonnes permettant d'identifier une configuration de façon unique dans tout le réseau (*RouterName* et *PoliceConfigIndex*). De plus elle comporte des colonnes permettant de connaître la configuration en question. La liste des colonnes et leurs descriptions sont dressés ci-dessous.

a. ***PolicingID :*** Identifie une configuration de façon unique dans la table.

b. ***RouterName :*** Conserve le nom d'hôte de l'équipement sur lequel la configuration est appliquée.
c. ***PoliceConfigIndex :*** Conserve l'index de configuration.
d. ***CIR :*** Conserve la valeur du paramètre CIR d'une configuration de *policing*.
e. ***Bc :*** Conserve la valeur du paramètre Bc d'une configuration de *policing*.
f. ***Be :*** Conserve la valeur du paramètre Be d'une configuration de *policing*.
g. ***ConformAction :*** Conserve la configuration de l'action prise pour le trafic conforme aux spécifications.
h. ***ConformSetValue :*** S'il y a lieu, conserve la valeur de remarquage pour le trafic conforme aux spécifications.
i. ***ExceedAction :*** Conserve la configuration de l'action prise pour le trafic excédant les spécifications.
j. ***ExceedSetValue :*** S'il y a lieu, conserve la valeur de remarquage pour le trafic excédant les spécifications.
k. ***ViolateAction :*** Conserve la configuration de l'action prise pour le trafic violant les spécifications.
l. ***ViolateSetValue :*** S'il y a lieu, conserve la valeur de remarquage pour le trafic violant les spécifications.
m. ***Feuilles :*** Conserve les feuilles de l'OID utilisé pour identifier cette configuration de *policing*, de façon unique, dans l'équipement en question. Ces feuilles seront utilisées pour la récolte de statistiques.

Finalement, la table *RED* est utilisée pour lister les configurations d'évitement de congestion (RED) présentes dans le réseau. Elle comporte des colonnes permettant d'identifier une configuration de façon unique dans tout le réseau (*RouterName* et *REDConfigIndex*). De plus elle comporte des colonnes permettant de connaître la configuration en question. Une particularité avec ce type de configuration est que, pour une classe donnée, il y a une entrée dans la table pour chaque valeur définie dans les

match statement de la classe en question. La liste des colonnes et leurs descriptions sont dressées ci-dessous.

a. ***REDID :*** Identifie la configuration de façon unique dans la table.

b. ***RouterName :*** Conserve le nom d'hôte de l'équipement sur lequel la configuration est appliquée.

c. ***REDConfigIndex :*** Conserve l'index de configuration.

d. ***BasedOn :*** Spécifie si le mécanisme est activé en se basant sur le champ DSCP ou *Precedence* de l'entête IP.

e. ***BaseValue :*** Valeur (relative à *BasedOn*) pour laquelle les paramètres ont été définis.

f. ***MinThreshold :*** Valeur du seuil minimum de la valeur définie dans *BaseValue.*

g. ***MaxThreshold :*** Valeur du seuil maximum de la valeur définie dans *BaseValue.*

h. ***MaxDropProb :*** Valeur du paramètre MPD de la valeur définie dans *BaseValue.*

i. ***Feuilles :*** Conserve les feuilles de l'OID utilisé pour identifier cette configuration du mécanisme d'évitement de congestion, de façon unique, dans l'équipement en question. Ces feuilles seront utilisées pour la récolte de statistiques.

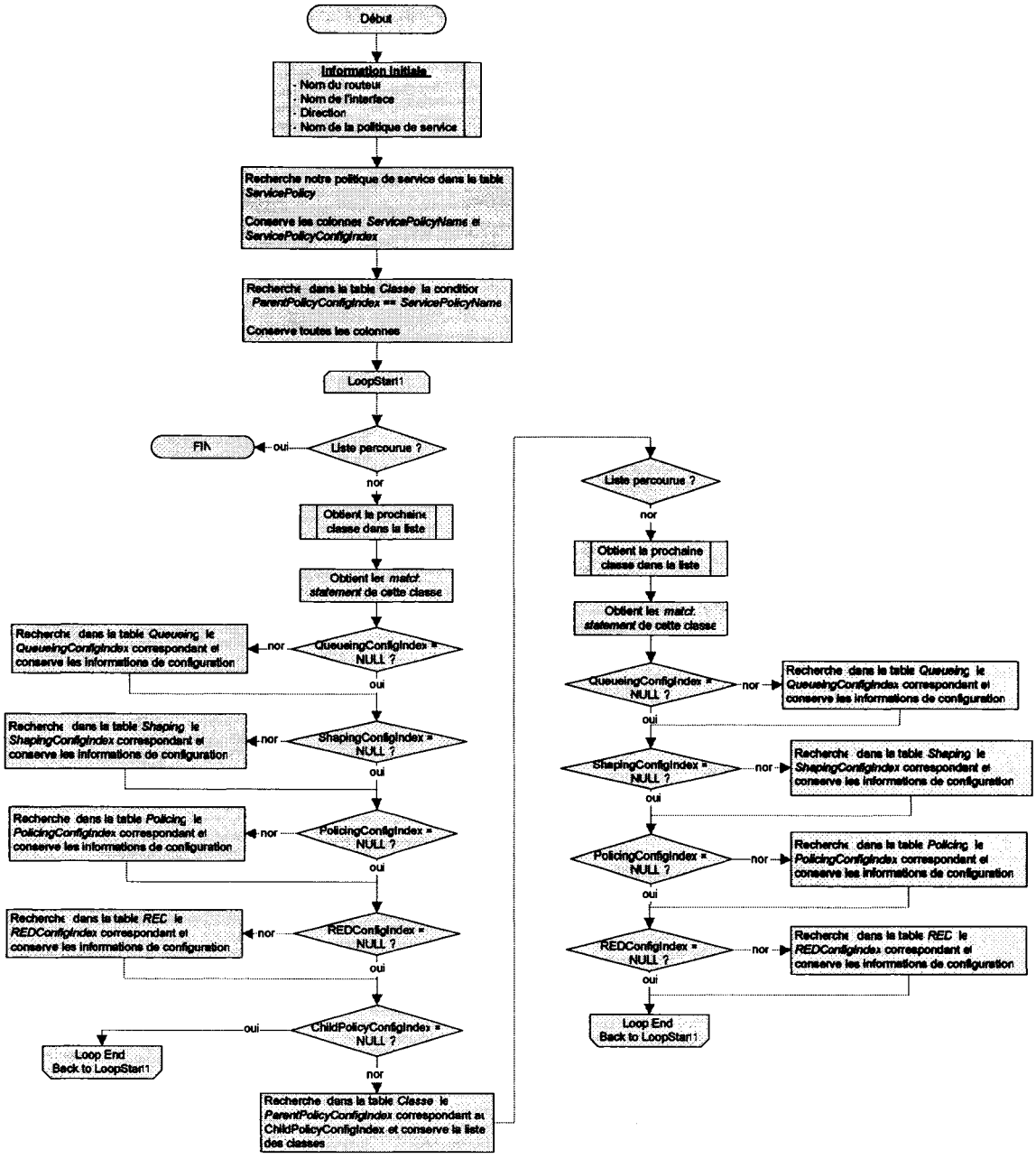


Figure 65 Organigramme utilisé pour obtenir une configuration de QdS

Finalement, la Figure 65 présente un organigramme utilisé pour aller chercher une configuration de QdS d'une politique de service donnée. Cet organigramme part du principe que l'utilisateur a spécifié le nom du routeur et de l'interface ainsi que la

direction et le nom de la politique de service à visualiser. D'abord il est nécessaire de connaître le *ServicePolicyConfigIndex* correspondant à la politique demandée. Celui-ci peut-être trouvé dans la table *ServicePolicy*. De plus, des informations générales à cette politique peuvent être trouvées dans cette table. Par la suite, il faut aller chercher la liste des classes, dans la table *Classe*, dont le *ParentConfigIndex* correspond à *ServicePolicyConfigIndex* précédemment obtenu. Pour chacune des classes de la liste, les *match statement* sont obtenus depuis la table *MatchStmt*. Par la suite, si le *QueueingConfigIndex*, le *ShapingConfigIndex*, le *PolicingConfigIndex* ou le *REDConfigIndex* sont non nulle, la table appropriée (*Queueing*, *Shaping*, *Policing* ou *RED*) est explorée afin de trouver les informations dont l'index de configuration correspond à celui défini dans la table *Classe*. Finalement si le *ChildPolicyConfigIndex* est non nul, le même processus est répété avec cette politique enfant afin d'obtenir la liste des classes et les configurations correspondantes.

Base de données de statistiques

Tel que mentionné dans l'introduction de cette annexe, cette section présente les diverses tables de la base de données de statistiques. La Figure 66 présente les diverses tables et colonnes les composants. On peut donc voir dans cette figure qu'une seule table compose cette base de données. En fait, cette table est la seule table statique de cette base de données les autres tables étant crées dynamiquement.

La table *StatsTablesInfo* est utilisée pour obtenir de l'information au sujet des autres tables qui sont créées dynamiquement. En effet, la table *StatsTablesInfo* contient une liste des autres tables présentent dans cette base de données. De plus, cette table contient des colonnes (*HostName*, *InterfaceName*, *Direction*, *ServicePolicyName* et *Class*) pour identifier chaque classe, d'une politique de service quelconque, de façon unique dans tout le réseau. Par ailleurs, cette table comprend la liste de tous les OIDs surveillés ainsi que le nom des colonnes dans lesquelles la valeur obtenue, suite à l'interrogation d'un

agent SNMP, sera introduite. La liste des diverses colonnes et descriptions associées est dressée ci-dessous.

a. ***HostName :*** Identifie un équipement réseau où la politique de service surveillée est appliquée.

b. ***InterfaceName :*** Identifie l'interface de l'équipement réseau sur laquelle la politique de service surveillée est appliquée.

c. ***Direction :*** Identifie la direction du trafic à laquelle la politique de service surveillée est appliquée.

d. ***ServicePolicyName :*** Identifie le nom de la politique de service surveillée.

e. ***Class :*** Identifie la classe surveillée de la politique de service sélectionnée.

f. ***InitialTime :*** Donne la date et l'heure où les tables correspondantes à la politique surveillée ont été créées.

g. ***StatsType :*** Identifie le type de statistiques récoltées pour la classe identifiée dans la colonne *Class*. Le type varie selon les mécanismes de QdS configurés dans la classe.

h. ***TableName :*** Identifie le nom de la table dans laquelle les statistiques seront enregistrées. Une table est créée pour chaque classe afin d'éviter qu'il y ait des colonnes avec le même nom. De plus, une table est créée pour récolter l'usage du CPU d'un équipement donné. Le nom d'une table pour l'usage du CPU et pour une classe donnée, est structuré comme suit :

HostName_InterfaceName_Direction_ServicePolicyName_CPU

HostName_InterfaceName_Direction_ServicePolicyName_Class

Dans ces noms de table, les caractères '.', '/' et '-' sont remplacés par le caractère '_' puisque les premiers ne peuvent être introduits dans un nom de table ou de colonne.

i. ***OID :*** Identifie l'OID de la statistique demandée.

j. ***Feuilles :*** Identifie la ou les feuilles requises pour compléter l'OID afin d'obtenir la statistique pour un mécanisme d'une classe précise.

k. ***ColumnName :*** Identifie la colonne de la table *TableName* dans laquelle seront introduites les statistiques identifiées par les colonnes *OID* et *Feuilles*.

l. ***WaitingTime :*** Identifie la durée d'attente entre deux collectes de statistiques.

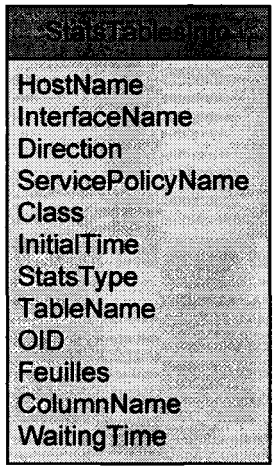


Figure 66 Base de données de statistiques de QdS

Pour chaque table de statistique créée dynamiquement, la première colonne est toujours la colonne *Time* qui identifie le temps (Date et heure) auquel la valeur de statistique a été reçue. Lorsqu'il s'agit de récolter les statistiques d'utilisation du CPU, seule la colonne CPU est ajoutée à cette table. Concernant les mécanismes de QdS, le Tableau XX de l'ANNEXE 3 résume le nom des colonnes ajoutées pour chacun des types de statistiques pouvant être récoltées.

ANNEXE 3

Manuel de l'utilisateur de QMA

Cette annexe présente le manuel d'utilisateur du système QMA. Il se concentre particulièrement sur l'usage des différentes options relatives à la qualité de service. Il existe deux interfaces utilisateurs. L'une est utilisée par un utilisateur standard qui se connecte depuis n'importe où dans le réseau (voir Figure 67) et l'autre par l'administrateur du système qui l'utilise depuis le serveur même (voir Figure 73) afin de contrôler ce dernier. Les sous-sections qui suivent présenteront l'usage de ces deux interfaces respectives.

Interface utilisateur de l'applet

Cette section présente l'utilisation de l'interface utilisateur de l'applet. La Figure 67 présente la fenêtre de l'applet tel que vue par un utilisateur standard. Avant de pouvoir faire quelque action que ce soit, l'utilisateur doit impérativement entrer un nom d'usager et un mot de passe valide. La Figure 67 montre, encerclés d'un cercle rouge, les divers onglets utilisés pour la visualisation de la QdS. Bien que tous les onglets de QdS soient accessibles après que l'utilisateur se soit authentifié, l'utilisateur ne pourra rien voir avant d'avoir effectué une action dans l'onglet *Network*.

À la Figure 68, l'onglet *Network* est représenté. On peut y voir les différents équipements représentés sous forme d'icônes. L'utilisateur peut cliquer sur un de ces icônes, en tenant le bouton enfoncé, et le déplacer à sa guise. On voit également, dans cette figure, que deux *ListBox* on été identifiées. Ces *ListBox* sont particulièrement utiles pour la sélection d'une politique de service d'un équipement donné. En effet, lorsque l'utilisateur désire sélectionner une politique pour un équipement donné, tant pour visualiser les configurations que les statistiques, le *Listbox1* est utilisé pour afficher toutes les interfaces de l'équipement sélectionné sur lesquelles une politique de service est appliquée. Le *ListBox2* est, quant à lui, utilisé lorsque l'utilisateur a sélectionné une interface dans le *ListBox1*. En effet, toutes les politiques de services appliquées sur l'équipement et l'interface sélectionnée sont alors listées. Le format d'affichage de ce

ListBox va comme suit : *Direction: Nom_de_la_politique_de_service*, où la *Direction* peut être *input* ou *output*.

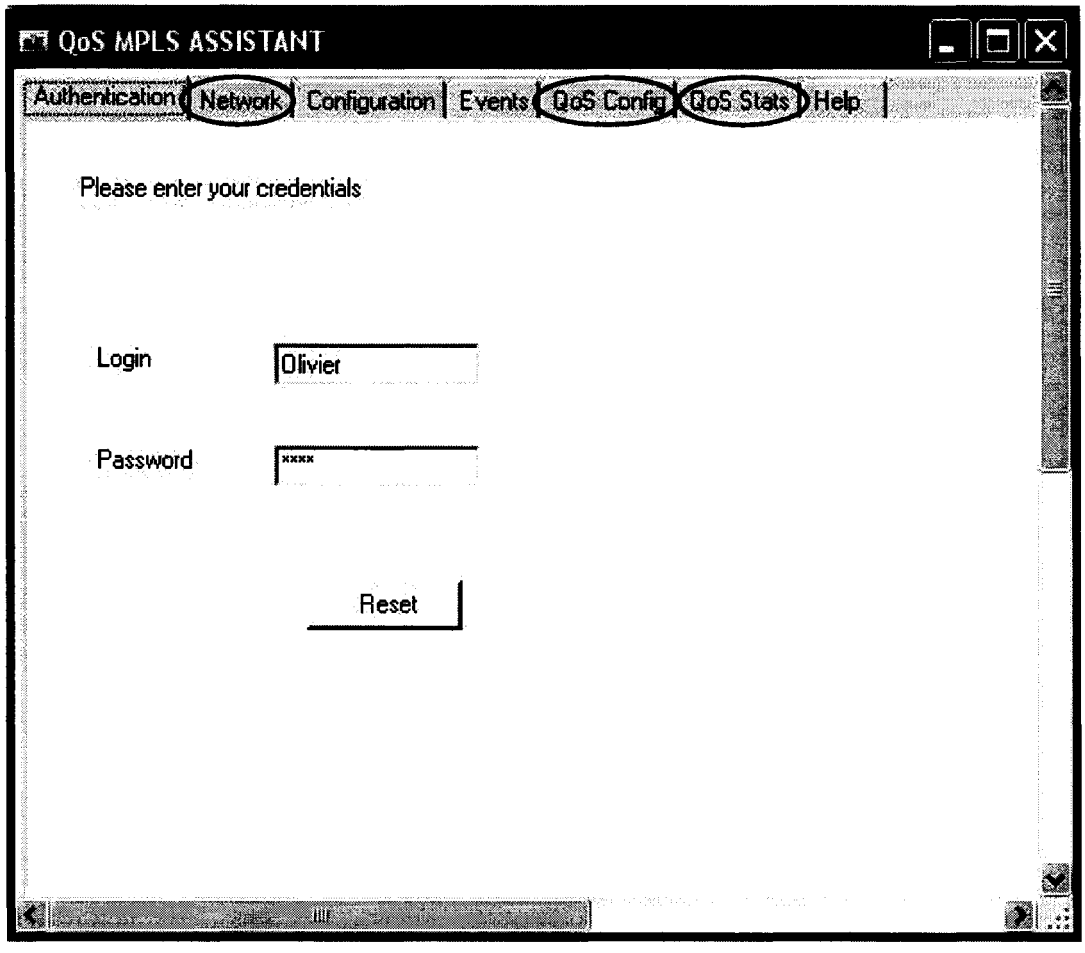


Figure 67 Fenêtre principale de l'applet vue par un utilisateur standard

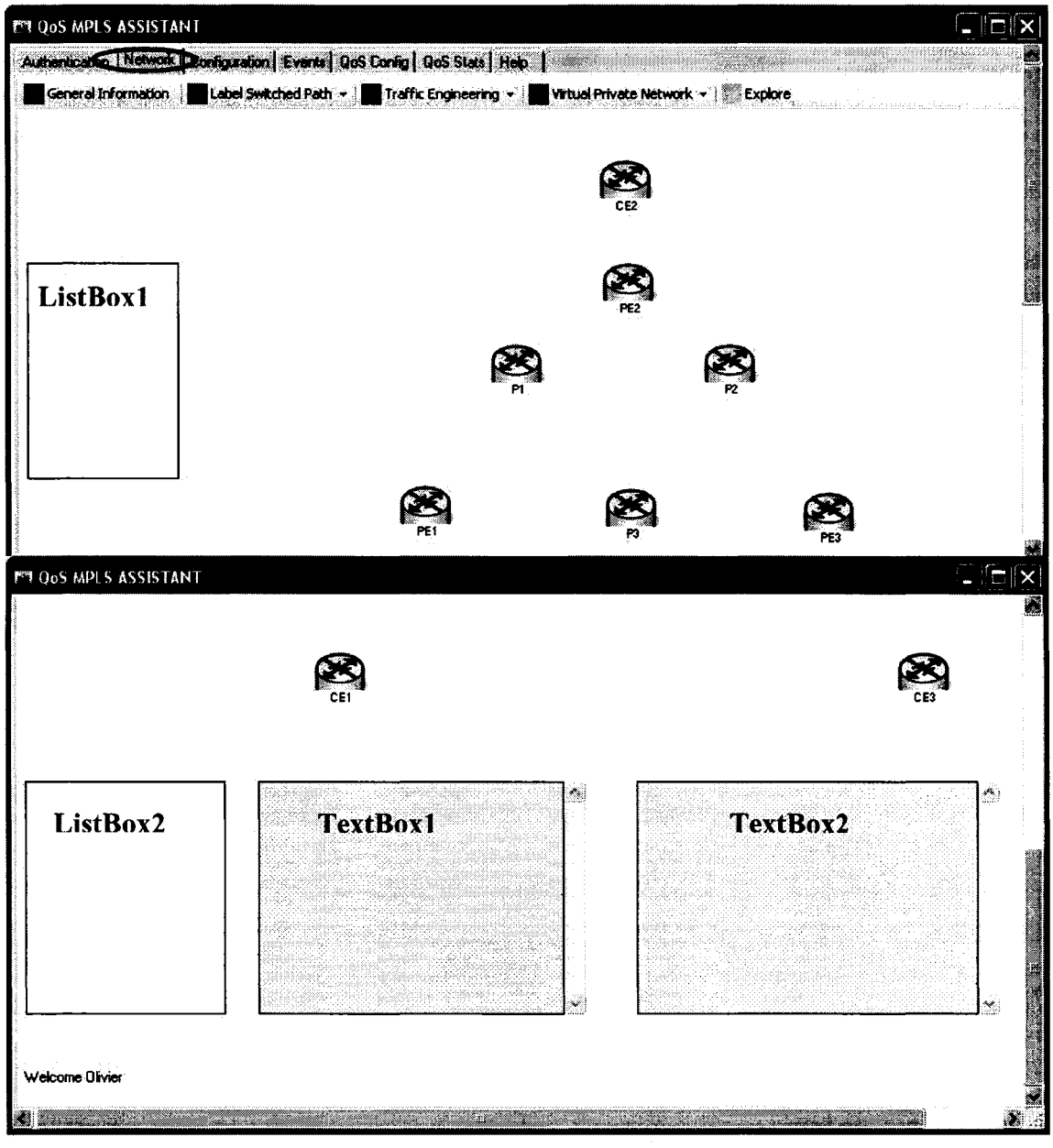


Figure 68 Onglet *Network* tel que vu par l'utilisateur de l'applet

La Figure 69 présente le menu contextuel obtenu lorsque l'utilisateur clique avec le bouton droit de la souris sur un équipement du schéma du réseau. L'option *Info* présente des informations générales, pour ce routeur, dans les *TextBox* 1 et 2. L'option *Start Telnet* est utilisée pour démarrer une session Telnet sur l'équipement sélectionné tandis

que l'option *Autres* a été laissée pour le développement futur. Le sous-menu *QoS* présente trois options à l'utilisateur. Évidemment ces options concernent les politiques de QdS configurées sur l'équipement sélectionné. L'option *ReNew QoS Configuration* est utilisée pour rafraîchir la base de données de configuration pour l'équipement sélectionné. L'option *Show Service Policy* est utilisée pour afficher la configuration d'une politique de service précise. Une fois que l'usager a cliqué sur cette option, les *ListBox* 1 et 2 se remplissent tel que mentionné précédemment. Lorsque l'utilisateur clique dans le *ListBox2*, pour sélectionner la politique de service dont il désire voir la configuration, l'onglet change automatiquement à l'onglet *QoS Config* et l'utilisateur peut visualiser la configuration tel que présenté dans la Figure 70. Quant à l'option *Begin Statistic Collection*, elle agit de façon similaire sur les *ListBox* de sorte que l'utilisateur puisse sélectionner la politique à surveiller. Une fois sélectionnée, la collecte de statistiques débute et l'onglet change automatiquement à l'onglet *QoS Stats* afin que l'utilisateur puisse visualiser ce qu'il désire.

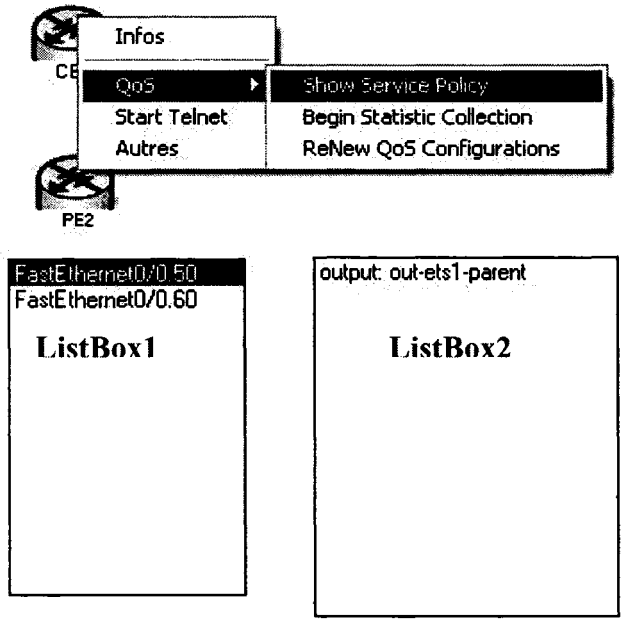


Figure 69 Présentation du menu contextuel et des *ListBox* de l'onglet *Network*

Onglet *QoS Config*

Tel que vu dans la Figure 70, un *ComboBox* contient la liste des identificateurs de politique que l'usager peut sélectionner pour afficher la politique. Ces identificateurs sont composés du nom de l'équipement, du nom de l'interface, de la direction ainsi que du nom de la politique de service tous séparés par un tiret (-).

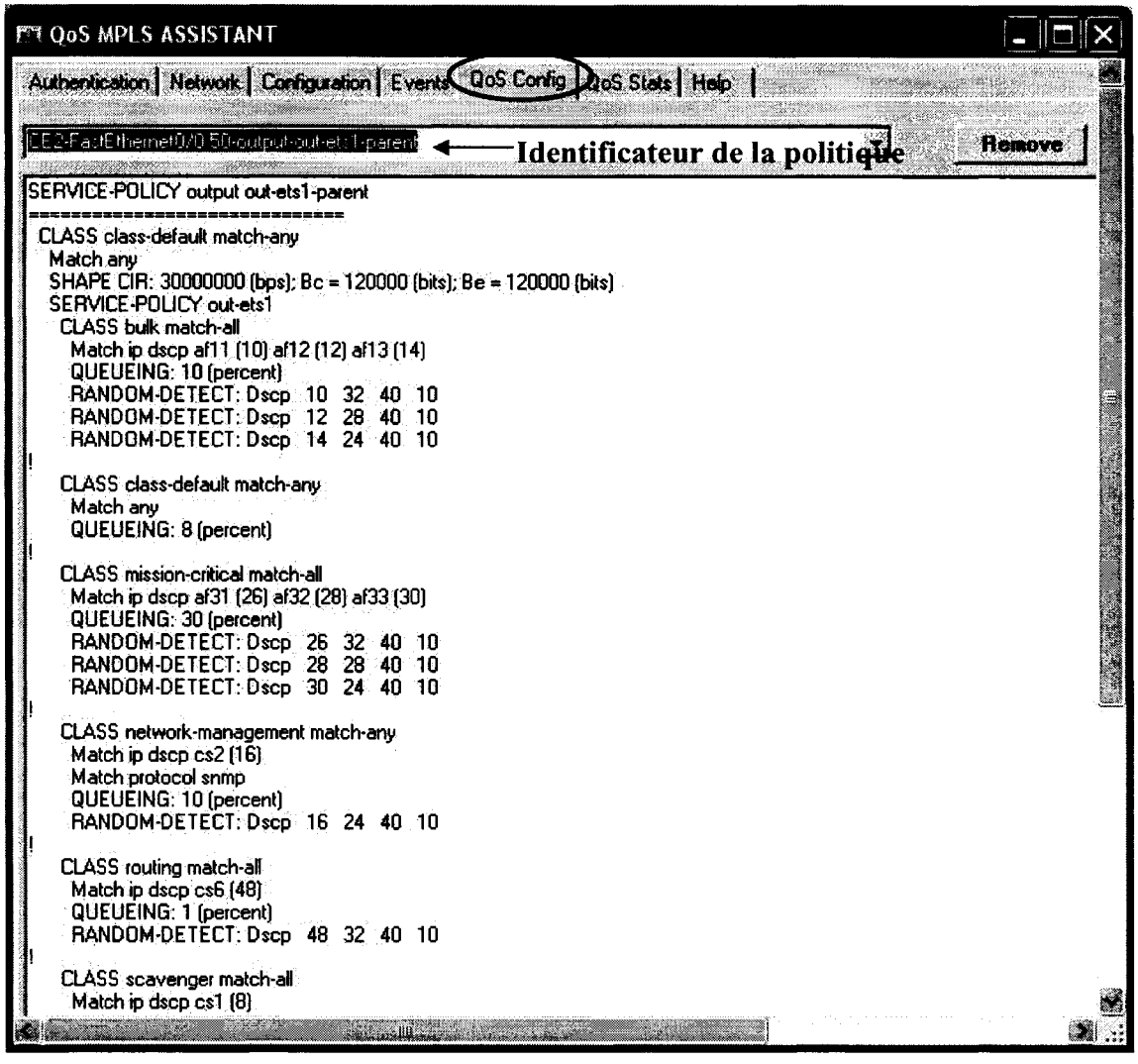


Figure 70 Exemple de configuration affichée dans l'onglet *QoS Config*

Le *TextBox* comporte la configuration en format texte de la politique sélectionnée dans le *ComboBox* (voir Figure 71). Le bouton *Remove* est quant à lui utilisé pour enlever une configuration de la liste de configuration du *ComboBox*.

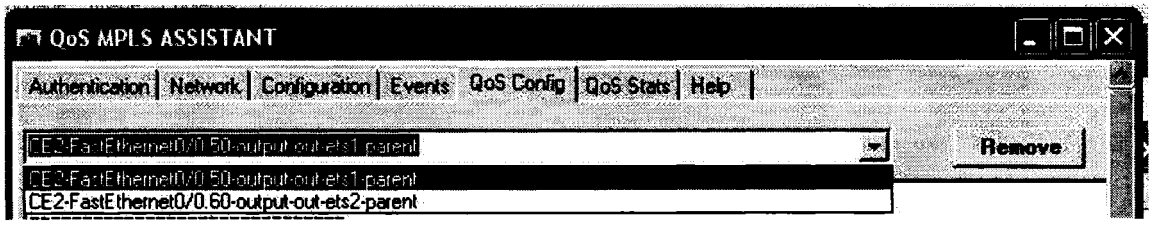


Figure 71 Exemple de sélection de configuration à afficher

Pour bien comprendre les configurations, il est essentiel de savoir d'abord que l'affichage utilise une certaine structure hiérarchique. D'abord la première ligne identifie principalement la direction ainsi que le nom de la politique de service sous le format : « SERVICE-POLICY *Direction Nom_de_la_politique* ». En dessous de cette ligne nous retrouvons l'identification de chacune des classes mais décalé légèrement par rapport à la première ligne. La liste des classes est présentée par ordre alphabétique. L'identification de la classe utilise la structure suivante : « CLASS *Nom_de_la_classe Match_Mode* » où le *match mode* identifie si l'opération OU (*match-any*) ou l'opération ET (*match-all*) est effectuée entre tous les *match statements*. Tout ce qui est implémenté à l'intérieur de la classe est décalé légèrement par rapport à l'identification de la classe. De plus tout ce qui peut être implémenté dans une classe est décrit dans le Tableau XIX.

Tableau XIX

Description de la présentation des configurations d'une classe

Identificateurs	Descriptions
Match	Identifie un *Match Statement*, c'est-à-dire les critères de sélection de la classe. Ils sont obtenus en format texte depuis la MIB et affichés tel quel.
SERVICE-POLICY	Identifie le nom de la politique de service appelée à l'intérieur de la classe. Utilisé principalement pour le *hierarchical traffic policy*.
QUEUEING	Indique la quantité de bande passante allouée à la classe ainsi que l'unité utilisés (*percent* ou *kbps*). De plus, lorsque la file est de type LLQ, « (Priority-queue) » est affiché à la fin de la ligne. De façon similaire, lorsque la file est de type *Fair-queue*, « (Flow-Based Fair-queueing) » est ajouté à la fin de la ligne. Dans le cas contraire, rien n'est ajouté à la ligne, ce qui signifie que le CBWFQ est employé.
SHAPE	Identifie les paramètres de lissage tel que CIR, Bc et Be et identifie également l'unité de chacun de ces paramètres.
POLICE	Identifie les paramètres de *Policing* tel que CIR, Bc et Be ainsi que leurs unités. Dans les lignes inférieures et avec un retrait plus grand, les actions pour le trafic conforme, en excès et violant les paramètres de *policing* sont identifiées. De plus, s'il y a remarquage pour l'une ou l'autre de ces actions, les valeurs de remarquage sont identifiées.

Tableau XIX (suite)

Identificateurs	Descriptions
RANDOM-DETECT	Identifie dans un premier temps, si le WRED est activé sur le champ DSCP ou le champ *Precedence* de l'entête IP. Si la classe rencontre des valeurs spécifique DSCP ou *Precedence*, les paramètres du WRED sont identifiés pour chacune de ces valeurs dans le format : RANDOM-DETECT: Dscp 26 32 40 10 À la suite de l'identificateur *Dscp*, la première valeur identifie la valeur DSCP, la seconde le seuil minimal, la troisième le seuil maximal puis la quatrième le paramètre MPD.

Onglet *QoS Stats*

La Figure 72 montre l'onglet *QoS Stats* tel que vu par l'utilisateur. Cependant, il n'y a pas de graphique affiché tant que l'utilisateur ne sélectionne pas une classe et une statistique à afficher dans les *ComboBox Class* et *Show* respectivement. Notez que le *Label* nommé *Initial Time* indique la date et l'heure du début de la collecte de statistiques.

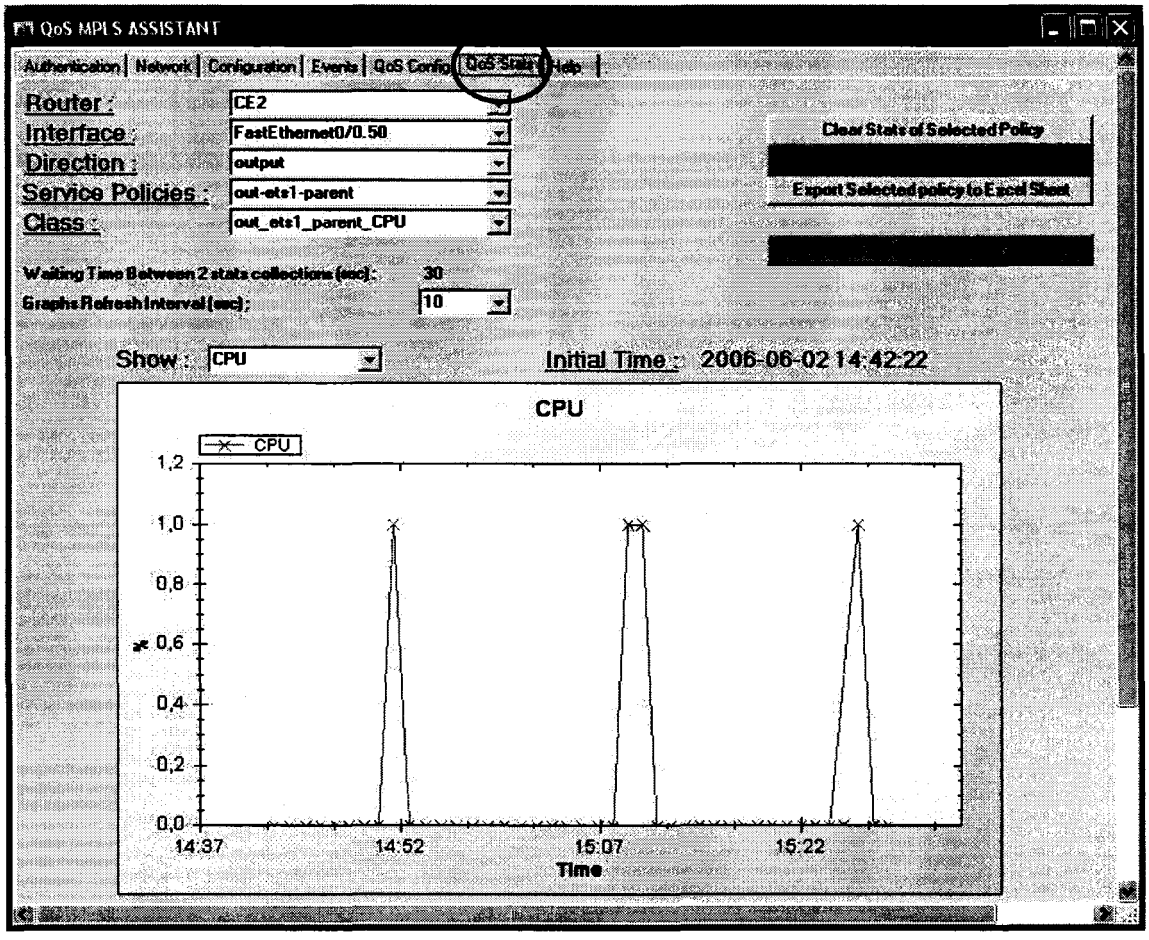


Figure 72 Exemple de statistiques affichées dans l'onglet *QoS Stats*

On peut voir dans cette figure qu'il y a sept *ComboBox* qui servent tous à sélectionner les statistiques à afficher. En voici leur liste et description :

a. **Router**
   - Utilisé pour sélectionner le routeur pour lequel l'usager désire avoir les statistiques.

b. **Interface**
   - Utilisé pour sélectionner l'interface du routeur dont l'usager désire avoir des statistiques.

c. **Direction**
   - Utilisé pour sélectionner la direction de la politique dont l'usager désire avoir des statistiques.

d. **Service Policies**
   - Utilisé pour sélectionner la politique dont l'usager désire avoir des statistiques.

e. **Class**
   - Utilisé pour sélectionner la classe dont l'usager désire avoir des statistiques. Il est important de noter que les noms des classes utilisent d'abord le nom de leur politique de service parent puis le nom de la classe séparés d'un « _ ». Cette façon de faire est pratique pour différencier les classes qui ont le même nom mais qui ont une politique de service parent différente. Par exemple, lorsque le *hierarchical policy* est réalisé, il y a toujours deux classes nommées *class-default*.

f. **Show**
   - Utilisé pour sélectionner les statistiques que l'usager désire voir. Pour une description des statistiques affichées lors d'une sélection dans ce *comboBox*, référez au Tableau XX.

g. **Graph Refresh Interval (sec)**
   - Utilisé pour modifier l'intervalle de rafraîchissement du graphique (par défaut à 30 secondes).

Tableau XX

Descriptions des statistiques affichées lors d'une sélection dans le *comboBox Show*

Types	Statistiques	Descriptions
CPU	CPU	Utilisation du CPU en %.
Class-Map	Out_bps	Débit du trafic sortant de la classe en bits par secondes.

Tableau XX (suite)

Types	Statistiques	Descriptions
Class-Map (suite)	In_bps	Débit du trafic entrant dans la classe en bits par secondes.
	Drop_bps	Débit du trafic rejeté de cette classe en bits par secondes.
Queueing	CurrentQDepth_pkt	Nombre de paquets présentement dans la file d'attente.
	MaxQDepth_pkt	Nombre maximal de paquets qu'il y a eu dans la file d'attente.
	DiscardPkt_pkt	Nombre de paquets rejetés.
Shaping	IsActive	Indique si le lissage est actif (valeur 1) ou inactif (valeur 2).
	CurrentQSize_pkt	Nombre de paquets présentement dans la file d'attente de lissage.
	Delayed_pkt	Nombre de paquets ayant subi un délai.
	Drop_pkt	Nombre de paquets rejetés.
Policing	Conform_bps	Débit du trafic conforme.
	Exceed_bps	Débit du trafic en excès.
	Violate_bps	Débit du trafic violant les paramètres.
RED	*X_Y*_RdmDrop_pkt	Nombre de paquets rejetés dû au mécanisme RED où *X* correspond au type de valeur sur laquelle le RED est appliqué (DSCP ou Precedence) et Y correspond à la valeur elle-même.
	*X_Y*_TailDrop_pkt	Nombre de paquets rejetés dû à un débordement de la file d'attente. Voir définition de *X* et *Y* dans la description de *X_Y*_RdmDrop_pkt.

Par ailleurs on peut voir qu'il y a quatre boutons pouvant être utilisés par l'usager dont voici la liste et la description de chacun :

a. **Clear Stats of Selected Policy**
   - Vide la base de données de toutes les statistiques correspondant à la politique sélectionnée.

b. **Clear All Statistics**
   - Vide la base de données de toutes les statistiques de toutes les politiques surveillées. (À utiliser avec précaution.)

c. **Export Selected policy to Excel Sheet**
   - Exporte la base de données de la politique sélectionnée vers un fichier Excel.

d. **Load Existing Stats**
   - Charge toutes statistiques présentement surveillées afin de les rendre disponibles à l'utilisateur.

Interface utilisateur du contrôleur du serveur

La Figure 73 présente l'interface utilisateur du contrôleur du serveur (*Server Controller*) présentée à l'usager lors du démarrage de l'application. Les onglets spécifiques à la QdS sont encerclés en rouge (*QoS Stats* et *Control QoS Config*) tandis que l'onglet général est encerclé par une ligne pointillée en bleue.

Dans l'onglet *QoS Stats*, on retrouve exactement les mêmes *ComboBox* que dans l'onglet *QoS Stats* de l'applet. Ces *ComboBox* ont exactement les mêmes fonctionnalités. Cependant on retrouve un *ComboBox* supplémentaire, soit *Waiting Time (sec)*. Celui-ci est utilisé afin de modifier le temps d'attente entre deux collectes d'informations. Cette option n'est pas accessible par un utilisateur normal puisqu'il pourrait venir compromettre les performances du système. Il doit donc être utilisé avec précaution.

En plus des *comboBox*, on retrouve à peu près les mêmes boutons que dans l'applet à l'exception du bouton « *Export Selected policy to Excel Sheet* » qui n'est pas présent dans cette interface puisqu'il est plutôt pratique pour un utilisateur qui désire conserver ses statistiques. De plus il y a le bouton « Stop Collection » qui est utilisé pour arrêter ou redémarrer la collecte d'information d'une politique de service particulière. Ce bouton correspond en fait à une option utilisée pour déboguer le système. C'est pourquoi elle n'est pas accessible à tous.

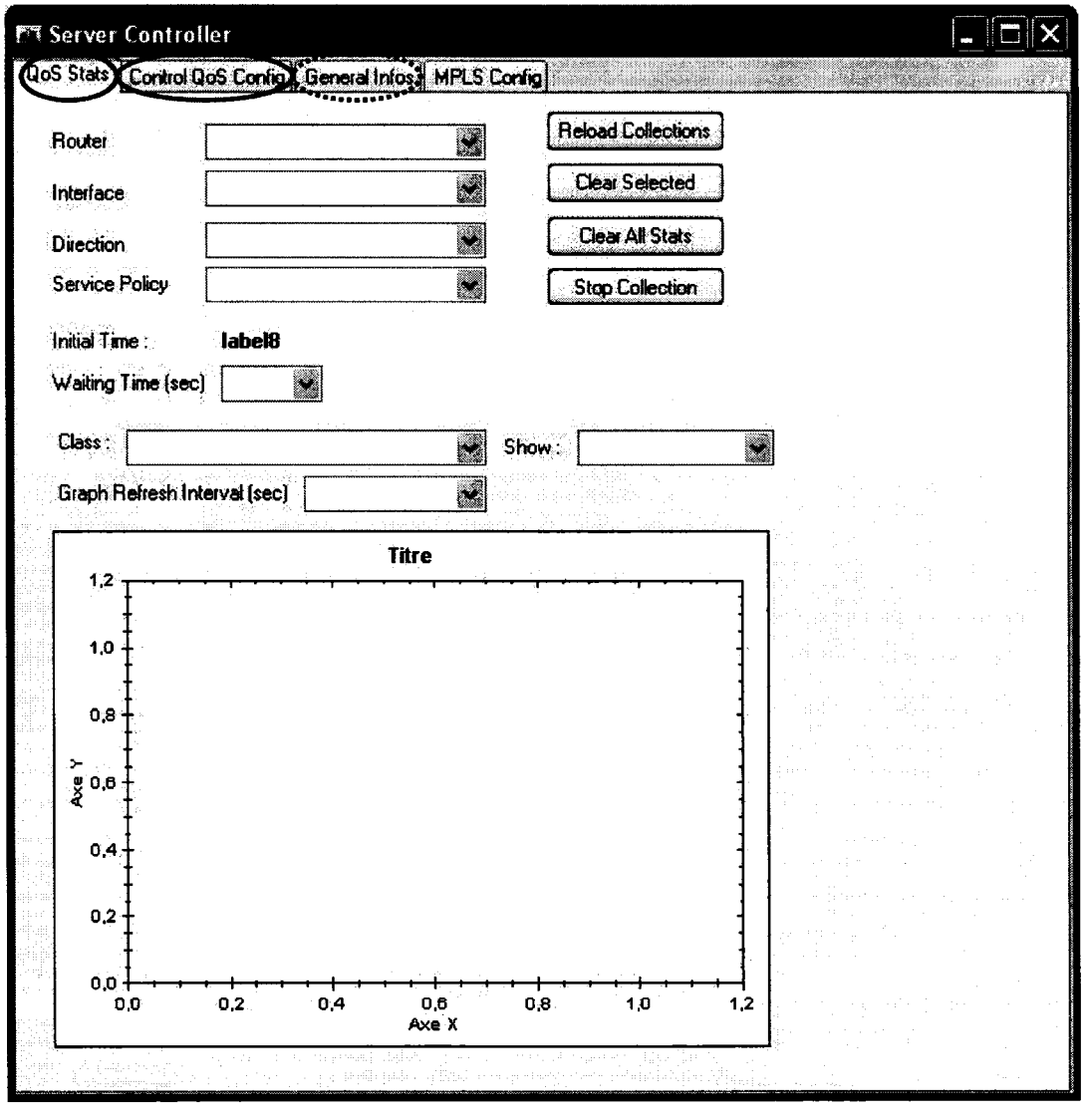


Figure 73 Fenêtre principale du contrôleur du serveur

La Figure 74 illustre l'interface de l'onglet *Control QoS Config* qui est utilisé principalement pour contrôler la base de données des configurations de QdS. Cet onglet est utilisé par un administrateur du système dans le but de faciliter le déboguage du système lorsque nécessaire.

Les boutons sont utilisés afin de remplir ou de vider les tables de configurations. D'abord les boutons *Fill_Router Table* et *Empty_Router Table* sont utilisés pour contrôler la table « Router » qui contient la liste des routeurs avec leur noms d'hôte, leurs noms d'interface ainsi que leurs adresses IP. Par la suite, les boutons encadrés en rouge sont utilisés pour contrôler les tables de configurations de la QdS soit en totalité (boutons *Fill_ALL* et *Empty_ALL*) ou en partie (les autres boutons excepté « *Clear ALL Tables for ALL* »). Tous, à l'exception du bouton « *Clear ALL Tables for ALL* », nécessitent la sélection d'un routeur dans la liste présentée à la gauche des boutons. Ainsi les tables sont contrôlées par routeur. Le bouton « *Clear ALL Tables for ALL* » est quand à lui utilisé pour vider toutes les tables de tous les routeurs.

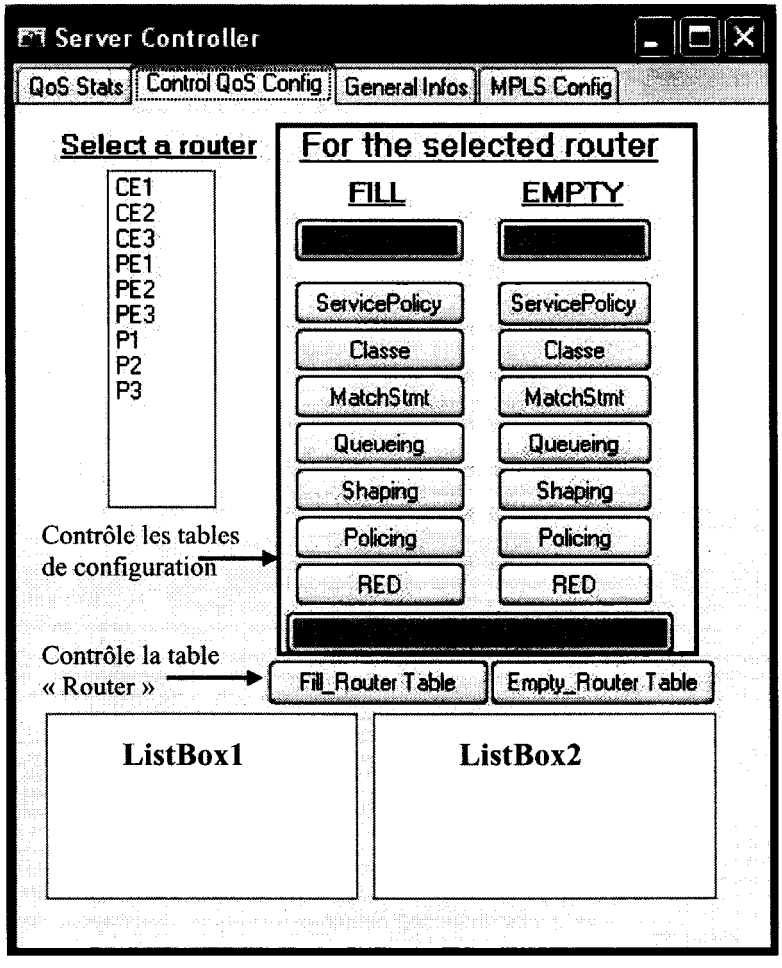


Figure 74 Onglet *Control QoS Config* du contrôleur de serveur

Une autre fonctionnalité de cet onglet consiste à afficher la configuration d'une politique de service donnée. En effet, si l'usager sélectionne un routeur, le *ListBox1* donne la liste des interfaces sur lesquelles une politique de service est appliquée pour ce routeur. Lorsque l'usager sélectionne ensuite une interface, le *listbox2* donne la liste des politiques configurée sur l'interface en question. Finalement, lorsque l'usager sélectionne une politique de service, sa configuration s'affiche dans une fenêtre de type Pop-up.

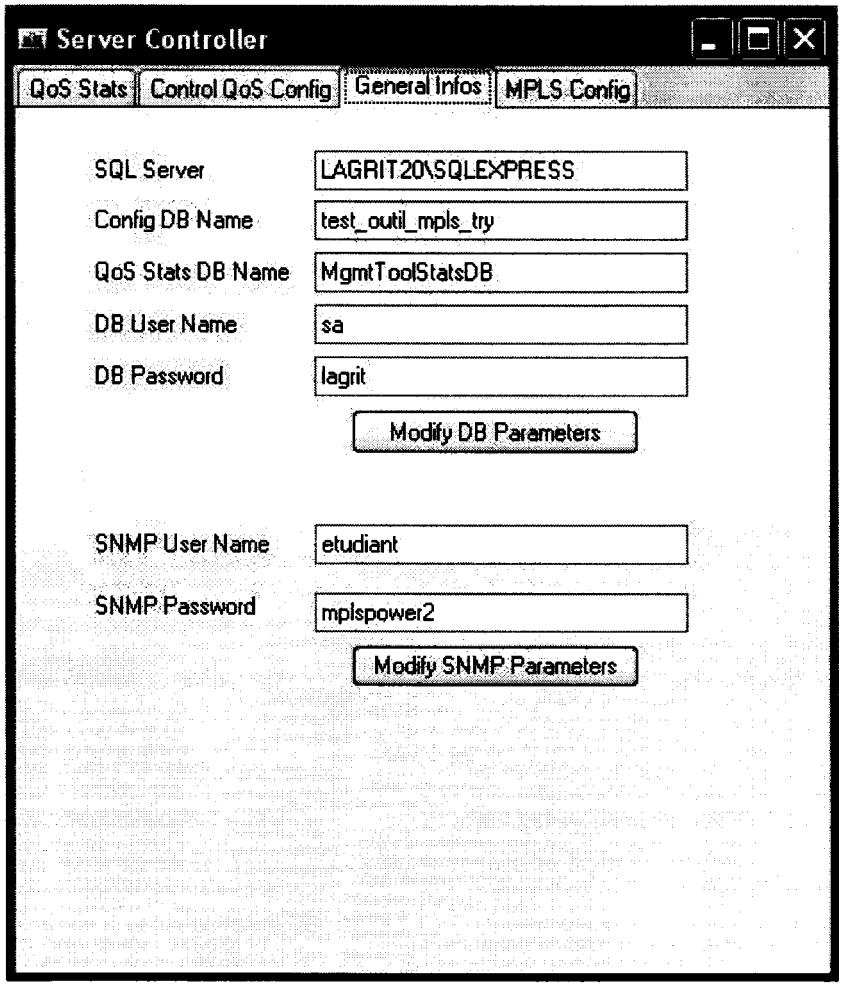


Figure 75 Onglet *General Infos* du contrôleur du serveur

La Figure 75 illustre l'onglet *General Infos* qui présente les paramètres de configuration globaux. On y retrouve entre autre le nom du serveur SQL, le nom des bases de données de configuration et de statistiques ainsi que leur nom d'usager et mot de passe. De plus, l'usager peut modifier ces paramètres directement dans les *TextBox* appropriés et appuyer sur le bouton *Modify DB Parameters* afin que les modifications soient appliquées. De façon similaire, on retrouve les paramètres généraux concernant SNMP, soit le nom d'usager et le mot de passe, puis un bouton pour appliquer les modifications à ces paramètres s'il y a lieu.

ANNEXE 4

Article soumis à la conférence ACM CoNEXT 2006

QoS for MPLS-based Virtual Private Networks

Mohamed EL HACHIMI	Marc-André BRETON	Maria BENNANI
Department of Software and IT Engineering	Department of Software and IT Engineering	Department of Software and IT Engineering
École de technologie supérieure, Montréal Canada	École de technologie supérieure, Montréal Canada	École de technologie supérieure, Montréal Canada
elhachimi_med@hotmail.com	**marc.andre.breton@lagrit.etsmtl.ca**	**maria.bennani@etsmtl.ca**

Abstract

Rapid growth of Internet traffic and the corresponding migration of real time and multimedia services towards IP, requires the introduction of technologies able to provide users with Quality of Service (QoS) and network operators with more dynamic and flexible resource utilization. In order to deal with the performance optimization in operational networks, Multi Protocol Label Switching (MPLS) brings the speed of Layer 2 switching to Layer 3 and enables advanced Traffic Engineering. In addition, MPLS offers two key advantages: it supports Quality of Service (QoS) and Virtual Private Networks (VPNs).

We consider the problem of providing per-customer service guarantees in MPLS based VPN routers; typically situated at the edge between a set of customers and a service provider network. Since all packets within a class of service are treated the same way, the granularity of services that can be offered to each customer is unclear. This paper focuses on the impact of scheduling and policing actions in the context of VPN services built on top of MPLS.

Work supported by Bell Canada and NSERC CRD grants

1 Introduction

MPLS [4] enables packets to be forwarded without undergoing the layer-3 stage. The initial goal of label-based switching used in MPLS was to increase the throughput of IP packet forwarding. Label-based switching methods allow routers to make forwarding decisions based on the contents of a simple label, rather than by performing a complex route lookup according to the destination IP address. This initial justification for MPLS is no longer perceived as the main benefit, since nowadays routers are able to perform route lookups at sufficiently high speeds to support most interface types. However, MPLS brings many other benefits to IP-based networks, including (1) traffic engineering, that is, the optimization of traffic handling in networks; (2) the support of quality of service; and (3) virtual private networks (VPNs), networks that offer private communication over the public Internet using secure links.

1.1 MPLS and Quality of Service

There is a growing need for Class-Of-Service (CoS) and traffic prioritization in order to support time-sensitive applications as voice and video over IP.

The connection oriented nature of MPLS provides the framework necessary to give quality guarantees to IP traffic. While QoS and Class of Service are not fundamental features of MPLS, they can be applied in MPLS networks where traffic engineering is used. This enables providers to establish Service Level Agreements (SLAs) with customers to guarantee service aspects such as network bandwidth, delay, and jitter. Value added services can be delivered in addition to basic data transport.

The differentiated services (DiffServ) [1] approach is a scalable way to provide QoS in IP Networks. However, DiffServ cannot offer end-to-end QoS by itself. In order to achieve quantitative end-to-end QoS guarantees and optimization of transmission resources, recent work in the IETF has focused on combining DiffServ and MPLS traffic engineering elements [2]. The DiffServ information in IP packet headers is mapped into the label information of the MPLS packets. MPLS routers act upon the prioritization information in the packets to forward the data appropriately. Some of the mechanisms used include traffic shaping, queuing, and packet classification. QoS is typically implemented at the edge of the MPLS cloud, where non-labelled traffic from the customer network enters the carrier network. At this entry point for example, delay-sensitive real-time traffic can be prioritized for delivery over bulk data transmissions.

1.2 MPLS based Virtual Private Networks

A Virtual Private Network (VPN) is a private network service delivered over a public (shared) network. VPNs benefit end customers by allowing remote locations to be securely connected over a public network, without the expense of buying or leasing dedicated network lines. MPLS enables VPNs [3] by providing a circuit-like, connection oriented framework, allowing carriers to deploy VPNs over the traditionally connectionless IP network infrastructure. The different VPN sites have to be interconnected through the provider's network (ISP network). The access points between a customer's site and the provider are called customer edge (CE) and provider edge (PE), respectively. The internal routers are called provider (P) routers (see Fig.1).

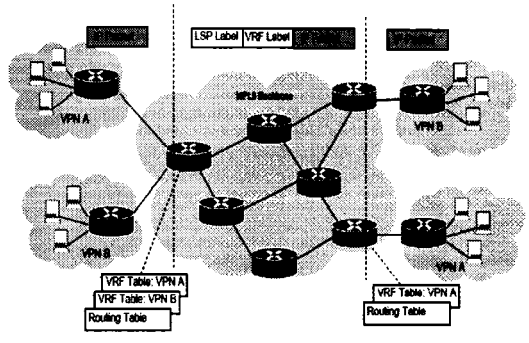


Figure 1. Layer 3 MPLS VPN network

MPLS VPNs fall into two broad classes those that operate at Layer 3 and those that operate at Layer 2. In this paper we focus on the Layer 3 VPNs. RFC 2547bis-based L3 VPNs use a two-level MPLS label stack (see Figure 1). The inner label carries VPN specific information from PE to PE. The outer label carries the hop-by-hop MPLS forwarding information. The P routers in the MPLS network only read and swap the outer label as the packet passes through the network. They do not read or act upon the inner VPN label; that

information is tunnelled across the network. Layer 3 VPNs use extensions to BGP, specifically Multi-Protocol BGP4, to distribute VPN routing information across the provider backbone. In an L3 VPN, the CE and PE routers are IP routing peers. The CE router provides the PE router with the routing information for the customers private network behind it.

One of the security mechanisms that are inherent in MPLS-based VPNs is traffic separation. In order to separate traffic, each MPLS-enabled VPN is assigned to a unique virtual routing and forwarding (VRF) instance. Traffic destined for each VRF carries its own label value, so each VPN is kept logically separate from every other VPN. In addition to the VRF tables, the PE router also stores the normal routing information it needs to send traffic over the public Internet.

A main requirement for MPLS based VPN services is to provide Quality of Service (QoS) guarantees. The motivation for this work is to gain a better understanding of the impact of aggregation on conformance checks that may be performed at MPLS based VPN network boundaries when providing classes of service. In this paper, we consider conformance deterioration caused by interactions among flows belonging to different VPNs but aggregated in the same traffic class.

2 Problem definition and system model

To achieve some level of Quality of Service (QoS) assurance, a network usually has Service Level Agreements (SLAs) with its users and neighbouring domains, which describe the QoS level that the service provider is committed to provide, and the specification of traffic that users or neighbouring domains are allowed to send. Violations occur when the application transmits faster than its contracted rate for an extended period of time, or generates a burst of packets that exceeds its burst tolerance.

It is known that the mechanism which manages the order of exit of packets on an interface is the scheduling. The scheduler determines the order in which the packets should be served by placing them into priority queues. Weights and priorities of the scheduler determine, dependent on the scheduler mode, the amount of bandwidth the connected component can achieve. Consequently, in situation of congestion, the various levels of services obtain a minimum guarantee service. Since the various VPN can have the same classes of services, a problem occurs when determining the order of exit of packets on the interface. Indeed, as the mechanism of scheduling differentiates only per class of service, it is impossible for him to carry out a differentiation by VPNs.

Avoiding contract violations is difficult if not impossible for many real-time applications. In the context of MPLS based VPN, we concentrate on the potential penalty imposed by aggregating traffic into a small number of packet treatments. Since flows may interact with each other and compete for resources at each node of a network, an interesting and important question arises as whether conforming traffics of a VPN will be protected from non conforming flows of other VPNs. To the best of our knowledge, it is the first attempt to focus on the problem of conformance in an

aggregate scheduling in the context of MPLS based VPN network.

In our investigation, the cross-traffic consists only of high priority traffic as our focus on the impact of aggregation. We assume that the impact of other traffic classes is minimal, e.g., because of the use of priority queues (high priority traffic is assigned to the higher priority) and a scheduling mechanism such as weighted fair queuing that isolates traffic classes. In order to investigate the issues described in this section, a test environment was built using real platform. Figure 2 shows the generic topology and setup used in most of the experiments we report on in this paper. In this topology, *VPN1* and *VPN2* are configured to carry traffics of costumers.

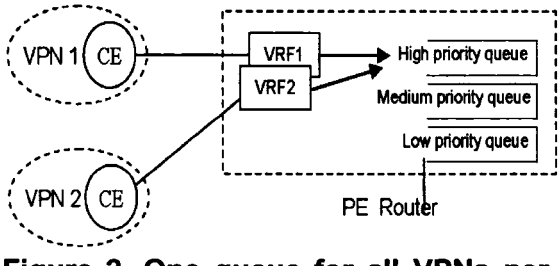


Figure 2. One queue for all VPNs per class of service

Two scenarios have been carried out in laboratory *SC1* and *SC2*. In these scenarios, only two VPNs were considered in order to reduce the configurations of the equipment. In both scenarios, only one queue was configured to be used to service high priority traffics belonging to *VPN1* and *VPN2*. Service rate is exponential with parameter $\mu_3 = 3$Mbps.

We consider the scenario *SC1* in which the hoped service rate for the *VPN1* is $\lambda_1 = 1$Mbps and for the *VPN2* is $\lambda_2 = 2$Mbps. In this scenario, the average arrival rates of packets is roughly equal to the hoped service rates so that $\lambda=\mu$. In this way, since the average arrival rates respect specificities of the contracts of service, it is clear that those will be respected.

Scénario	VPN	Configured μ (Mbps)	λ (Mbps)	Obtained μ (Mbps)
SC_1	VPN_1	3	1	1.076
	VPN_2		2	1.972
SC_2	VPN_1	3	1.15	0.849
	VPN_2		4.7	2.419

Table 1. Results in the case of one queue for all VPNs per class

The second scenario *SC2* was carried out to emphasize the disadvantage of such a configuration when the average arrival rates do not respect the specifications of the contracts. In this scenario, the hoped service rate is λ_1=1.15Mbps for the *VPN1* and λ_2=4.7Mbps for the *VPN2* . Indeed, as the average arrival rates are disproportionate compared to SLAs, the service rates obtained are also disproportionate and no guarantee of service can be considered in a such situation. The Table 1 summarizes the results obtained in both scenarios described previously.

3 Proposed solution and performance study

In order to provide separate guarantees to each flow, a provider edge switch router is required to differentiate between traffic from different customers. The proposed solution to resolve the problem described previously in section 2 concern the use of separate queue in the same class for each

VPN (or VRF)(Figure 3). Thus, the traffic generated by each VPN will be isolated and will not be influenced by the average arrival rate of another one.

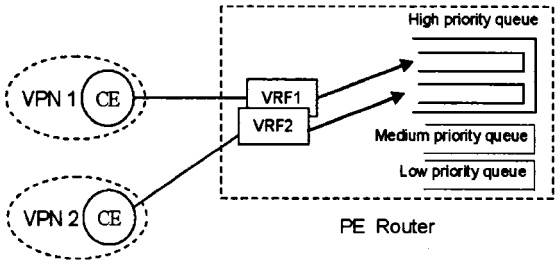


Figure 3. Separate queues per VPN and per class of service

In the following, we look at the per-flow (per-VPN) performance in different scenarios under different traffic characteristics in the case of a separate queue for each VPN. We consider two scenarios (*SC*3 and *SC*4). In both scenarios the service rate μ_1=1Mbps was configured for the *VPN*1 while the service rate μ_1=2Mbps was configured for the *VPN*2.

In the scenario *SC*3, the hoped service rate for the *VPN*1 is λ_1=3Mbps and for the *VPN*2 is λ_2=4Mbps. As expected, results from Table 2 show that when the average arrival rates of packets and proportions of bandwidth allocated to each one are not respected, the service rates of VPNs are respected.

Scénario	VPN	Configured μ (Mbps)	λ (Mbps)	Obtained μ (Mbps)
SC_3	VPN_1	1	3	0.987
	VPN_2	2	4	1.937
SC_4	VPN_1	1	0	0
	VPN_2	2	5	2.038

Table 2. Results in the case of separate queues per VPN and per class

In the scenario *SC*4, different hopped services are configured for the VPNs. The hoped service rate for the *VPN*1 is λ1=0Mbps and for the *VPN*2 is λ2=5Mbps.
This scenario has been realized in order to show that a VPN cannot use the unused bandwidth of another VPN. Such results are perfectly adequate in a context multi-VPN (Table 2).

Since one of the main services which can be provided in the context of MPLS based VPN network is the voice over IP, let us now see the impact of using different queue per VPN and per class of service on the delay.

4 Analysis of queuing delay

In order to study the effect of using a separate queue for each VPN and class of service on the delay, we propose in this section two scenarios. The first one uses one queue per class for all VPNs and the second one uses one queue per VPN and per class. In both scenarios, a LLQ (Low Latency Queuing) system is used to serve high priority traffics. First let us fix some parameters for this queue like the service rate μ and the average arrival rate λ and see the important impact of theses parameters on the delay of an LLQ in general. Knowing that a file LLQ can be modeled by a M/M/1/k queuing system with Poisson arrival process, exponentially distributed service rates and one server. K represents the total capacity of the system, or only the number of waiting positions. Certain parameters of performance can be theoretically given by the following formulas:

The load factor ρ is given by:

$$\rho = \lambda/\mu \quad (1)$$

λ = packets arrival rate (pkt/sec) (Poissonian)
μ = packets service rate (pkt/sec) (exponential)

The Packet Loss Ratio PLR is given by:

$$PLR = P_k = \frac{\rho^k \times (1-\rho)}{1-\rho^{k+1}} \quad (2)$$

N is the average number of packets in the system:

$$N = \frac{\rho \times (1 + k \times \rho^{k+1} - (k+1)\rho^k)}{(1-\rho)(1-\rho^{k+1})} \quad (3)$$

T is the average time spent by a packet in the system:

$$T = \frac{N}{\lambda \times (1 - PLR)} \quad (4)$$

Figure 4 shows well the influence which can have λ on the average delay and the blocking probability when μ is fixed to 3Mbps and k = 200 msec×μ. It is seen clearly that the times start to be more significant when λ > 90%×μ (or ρ > 0.9). However, with this value, the probability of blocking starts to be not null.

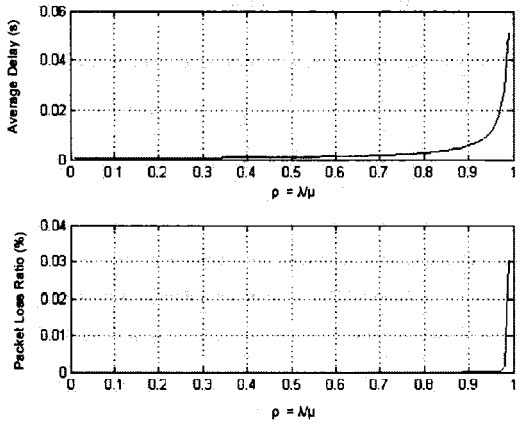


Figure 4. Average delay and blocking probability using fixed μ and variable λ

As a communication of VoIP is sensitive to packet losses, we can affirm that having a value of λ approximately equal to 80%×μ (or ρ = 0.8) is a good compromise so that the queue does not generate too many delays and losses. It is thus very important to oversize the queue from approximately 20% in order to avoid extra delay for the VoIP communications.

Let us now continue studying the scenarios we have present early in this section. In both cases the system was dimensioned in order to leave a margin of 20% between λ and μ. In the first scenario, only one M/M/1/k system was used to serve the two sources of traffic $S1$ and $S2$ while in the second scenario, two independent M/M/1/k systems were used to serve each source individually.

	λ (Mbps)	μ (Mbps)	N (pkt)	PLR (%)	T (msec)
S_1	0.8	N/A	N/A	N/A	N/A
S_2	1.6	N/A	N/A	N/A	N/A
Total	2.4	3	4	0	2.283

Table 3. Theoretical results in the case one queue for all VPNs per class

We can initially notice that in each case, the number of packets in the queues remains constant equal to 4 packets.

	λ (Mbps)	μ (Mbps)	N (pkt)	PLR (%)	T (msec)
S_1	0.8	1	4	0	8.56
S_2	1.6	2	4	0	4.28
Total	2.4	N/A	N/A	N/A	N/A

Table 4. Theoretical results in the case one queue per VPN and per class

We can notice thereafter that using two systems increases the average delays (Table 3 and Table 4). This can be explained by the fact that service rates are smaller which results in increasing service rates. Moreover, as the average number of packets in the queue is constant in the systems, the total average delay is influenced only by the service rate. Finally, figure 5 shows the influence which can have μ on the delay when λ is fixed to $0.8 \times \mu$ and k is fixed to $0.2 \times \mu$.

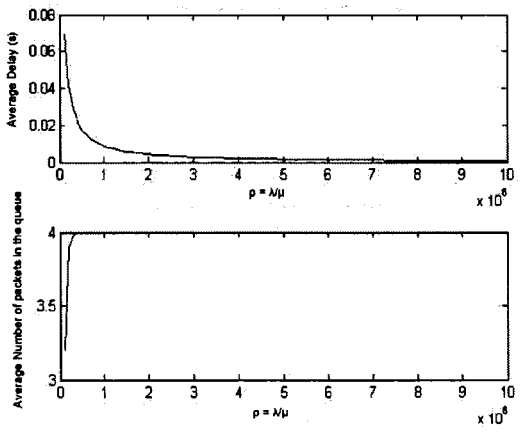


Figure 5. Average delay and blocking probability using fixed λ = 80% and variable μ

By considering that an adequate time for an LLQ queue would be lower than 5 msec, we can see that 2Mbps corresponds roughly to a minimal threshold so that the voice communications do not undergo too much delay in the LLQ queue. However, by considering the end-to-end treatment, only the equipment in the edge of the network (CE and PE for example) could use the independent queues per VPN and per class since the service provider will treat the data by aggregates, in the heart of the network, where only one queue is used to treat the entirety of the voice communications in each node (one queue for all VPNs per class). Consequently, the minimal threshold mentioned previously could be then decreased with 1Mbps to generate an average time of 10 msec.

5 Conclusion

The goal of this paper was to better understand the impact of traffic aggregation on conformance, in the context of VPN service built on top of the MPLS network when providing multiple classes of service. In order to avoid effect of non conforming traffics belonging to one VPN on the others in the same class of service, we have proposed to use different queues per VPN and per class of service. Results provided in the section 3 show the benefit of using such solution on the protection of conforming traffics against non conforming traffics. In the section 4, we analyzed the effect of using a separate queue for each VPN and class of service on the delay. Results obtained in this section, show that even if the delay is increased while using different queues per VPN and per class, some parameters could be fixed in order to provide an acceptable delay for voice communications.

References:

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. December 1998. RFC 2475.

[2] F. L. F. et al. Multi-protocol label switching (MPLS) support of differentiated services. May 2002. (online), http://www.ietf.org/rfc/rfc3270.txt.

[3] J. Guichard and I. Pepelnjak. Mpls and vpn architectures: A practical guide to understanding, designing and deploying mpls and mpls-enabled vpns. Aug 2002. Cisco Press, Indianapolis.
[4] E. Rosen and et al. Multiprotocol label switching architecture. January 2001. RFC3031.

BIBLIOGRAPHIE

[1] W. Stevens, "RFC 2001 : TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," IETF 1997.

[2] O. Truong, "Étude et développement d'outils d'optimisation de gestion de services dans les réseaux MPLS," in *Département de génie logiciel et des TI.* Montréal: École de technologie supérieure, 2006.

[3] M. Klerer and R. Ward, "Multiservice Switching Forum System Architecture Implementation Agreement," msforum 2000.

[4] C. Gallon and P. Drew, "Next-Generation VoIP Network Architecture," msforum 2003.

[5] P. Drew and S. Walker, "Release 2 Architecture," MSForum 2004.

[6] T. Ylonen and C. Lonvick, "RFC 4251: The Secure Shell (SSH) Protocol Architecture," 2006.

[7] P. Drew, "MSF Global Interop (GMI) 2004 Physical Test Scenarios," MSForum 2004.

[8] R. Stefic and N. Prib, "Measurement and analysis of users' perception of QoS for IP telephony service," presented at Telecommunications, 2003. ConTEL 2003. Proceedings of the 7th International Conference on, 2003.

[9] R. Braden, D. Clark, and S. Shenker, "RFC 1633 : Integrated Services in the Internet Architecture: an Overview," IETF 1994.

[10] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "RFC 2205 : Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification," IETF 1997.

[11] R. Yavatkar, D. Hoffman, Y. Bernet, F. Baker, and M. Speer, "RFC 2814 : SBM (Subnet Bandwidth Manager): A Protocol for RSVP-based Admission Control over IEEE 802-style networks," IETF 2000.

[12] A. Koubaa, A. Jarraya, and S. Ye-Qiong, "SBM protocol for providing real-time QoS in Ethernet LANs."

[13] S. Jha and M. Hassan, "SBM+: enhanced SBM for managing bandwidth in multiple access subnets," presented at Local Computer Networks, 2001. Proceedings. LCN 2001. 26th Annual IEEE Conference on, 2001.

[14] K. Nichols, S. Blake, F. Baker, and D. Black, "RFC 2474 : Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," IETF 1998.

[15] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "RFC 2597: Assured Forwarding PHB Group," IETF 1999.

[16] V. Jacobson, K. Nichols, and K. Poduri, "RFC 2598: An Expedited Forwarding PHB," IETF 1999.

[17] W. Odom and M. J. Cavanaugh, *IP Telephony self Study, Cisco QOS Exam Certification Guide*, Second ed: Cisco Press, 2005.

[18] A. Silberschatz, G. Gagne, and P. B. Galvin, *Operating System Concepts*, Seventh ed. Danvers: John Wiley & Sons, Inc., 2005.

[19] P. Kemper, D. Muller, and A. Thummler, "Combining response surface methodology with numerical models for optimization of class-based queueing systems," presented at Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on, 2005.

[20] D. Han, W. Cui, Y. Park, and S. An, "The design and implementation of a high performance management information tree for a TMN platform," presented at TENCON '98. 1998 IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control, 1998.

[21] M. Li and K. Sandrasegaran, "Network Management Challenges for Next Generation Networks," presented at Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on, 2005.

[22] "Information Technology - Open systems Interconnection - Common management information protocol : Specification," ITU-T X.711, 1997.

[23] A. Keller, "Tool-based implementation of a Q-Adapter Function for the seamless integration of SNMP-managed devices in TMN," presented at Network Operations and Management Symposium, 1998. NOMS 98., IEEE, 1998.

[24] "Information technology - Open Systems Interconnection - Common Information Management service," X.710, 1997.

[25] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "RFC 1157: A Simple Network Management Protocol (SNMP)," IETF 1990.

[26] J. Case, K. McCloghrie, M. Rose, S. Waldbusser, and J. Galvin, "RFC 1441-1452 : SNMPv2," IETF 1993.

[27] D. Harrington, R. Presuhn, B. Wijnen, J. Case, D. Levi, P. Meyer, B. Stewart, U. Blumenthal, B. Wijnen, and K. McCloghrie, "RFC 3411-3415," IETF 2002.

[28] W. Stallings, "SNMP and SNMPv2: the infrastructure for network management," *Communications Magazine, IEEE*, vol. 36, pp. 37-43, 1998.

[29] G. Jiang, "Multiple vulnerabilities in SNMP," *Computer*, vol. 35, pp. 2-4, 2002.

[30] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry, "RFC 2748 : The COPS (Common Open Policy Service) Protocol," IETF 2000.

[31] C. Gallon, "SIP IA for the interface between a SIP call agent and a bandwidth manager," MSForum 2004.

[32] C. Gallon, O. Schelen, A. Torger, and J. Johansson, "Implementation Agreement for Network Resource Control Protocol (NRCP)," MSForum 2004.

[33] F. Vallée, *Architecte logiciel: UML pour les décideurs*, Eyrolles ed. Paris, 2005.

[34] J. Simmonet, "Outil de surveillance dans un réseau MPLS," École de technologie supérieure, LAGRIT, Rapport de fin d'études 2005.

[35] P. Ladka, *ASP.NET for Web Designers*, New Ryders ed. Indianapolis: Dwyer, D., 2003.

[36] S. A. Hussain and A. Marshall, "Provision of quality of service using active scheduling," *Communications, IEE Proceedings-*, vol. 151, pp. 231-237, 2004.