TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AC	Admission Control
CBR	Constraint Base Routing
CE	Customer Edge
CLI	Command Line Interface
FT	Flow Termination
IETF	Internet Engineering Task Force
IOS	Internetwork Operating System
IP	Internet Protocol
LAGRIT	LAboratoire de Gestion de Réseaux Informatiques et de Télécommunications
LP	LAGRIT Poller
LSP	Label Switched Path
MPLS	Multi Protocol Label Switching
NS-2	Network Simulator 2
OSPF	Open Shortest Path First
PCN	Pre-Congestion Notification
PE	Provider Edge
PHP	PHP: Hypertext Preprocessor
RED	Random Early Detection
RIP	Routing Information Protocol
RSVP-TE	Resource Reservation Protocol - Traffic Engineering
RTG	Real Traffic Grabber
SNMP	Simple Network Management Protocol
TCL	Tool Command Language
TCP	Transmission Control Protocol
TE	Traffic Engineering
UDP	User Datagram Protocol

INTRODUCTION

Le protocole Internet (IP) a été développé dans le but d'échanger des données. Les réseaux IP fonctionnent en « best effort ». Cela signifie que l'acheminement des flux est effectué paquet par paquet sans tenir compte de l'ensemble. Lors de sa création, les services tels que la voix et la vidéo sur IP n'étaient pas encore prévus. La seule possibilité lorsque des paquets sont perdus est de réémettre ceux-ci. Or, l'émergence des nouveaux services temps réels (voix sur IP, vidéoconférence) nécessite de respecter des contraintes de trafic pour que leur qualité soit acceptable. Par exemple, la perte de paquets pour une communication voix doit être inférieure à 10 % sous peine d'avoir une communication dégradée. Lorsqu'un appel voix sur IP transite sur un réseau IP il est donc impossible d'en garantir la qualité.

Les réseaux utilisant le protocole IP sont difficilement modifiables pour être adapté à ces nouveaux besoins. En effet, il faudrait une migration complète des réseaux dans le but de changer la structure d'IP. Le protocole Multi Protocol Label Switching (MPLS) (Stallings, 1999) a donc été conçu pour tenter de résoudre ces problématiques sans changer la totalité du réseau. La venue des réseaux MPLS a permis d'ajouter des contraintes de service dans le trafic pour un moindre coût. De plus, ceux-ci sont moins dispendieux que les réseaux ATM qui permettent également d'assurer de la qualité de service et sont donc plus populaires aujourd'hui. Cependant, seule l'ingénierie de trafic (Traffic Engineering ou TE) implémentée sur MPLS permet de rajouter de la vraie qualité de service. Le protocole MPLS permet cependant un acheminement des paquets plus rapide dans le réseau. Les congestions se manifestent toujours au sein du réseau lorsque la demande devient trop importante ou que des rafales se produisent, mais la charge est mieux répartie. Selon le moment de la journée, on peut observer des périodes durant lesquelles le trafic est beaucoup plus important et où la qualité de service devient un élément crucial.

Les mécanismes de TE développés pour MPLS sont de plusieurs types. On retrouve les mécanismes proactifs (Oulai, Chamberland et Pierre, 2007), qui répartissent le trafic sur le

réseau pour limiter au maximum les congestions. Les mécanismes curatifs (Salvadori et Battiti, 2003) permettent eux de traiter les paquets une fois que la congestion a été détectée. La plupart des mécanismes préventifs essayent de répartir au mieux le trafic et n'acceptent plus les nouvelles demandes de flux une fois que la bande passante réservée est au maximum des capacités du réseau. Les mécanismes proactifs sont des mécanismes de reroutage des paquets, de diminution de la vitesse d'émission ou de changement de files d'attente.

Ce mémoire présente tout d'abord la problématique de la recherche en détail ainsi que les objectifs mis en place pour la résoudre. Les concepts généraux à connaître pour comprendre le travail effectué dans la maîtrise sont présentés dans le chapitre deux. Le chapitre trois présente l'étude des congestions effectuée pour ce mémoire. Le chapitre quatre définit l'algorithme développé dans le cadre de cette maîtrise. Le chapitre cinq détaille l'implémentation de l'algorithme de prédiction sur le banc d'essai ainsi que sur le simulateur de réseau. Le sixième chapitre présente les scénarios de validation de l'algorithme ainsi que les résultats de ces expériences. Enfin, des recommandations sont présentées pour la recherche future après la conclusion de ce projet.

CHAPITRE 1

PRESENTATION APPROFONDIE DE LA RECHERCHE

1.1 Problématique

Les méthodes de lutte contre la congestion dans les réseaux MPLS sont de deux types : proactives ou curatives.

Les méthodes proactives traitent les congestions bien avant qu'elles se produisent. Elles permettent de répartir la charge dans le réseau et d'éviter qu'une demande supplémentaire ne soit créée lorsque celui-ci atteint un certain niveau d'encombrement. Le problème avec les mécanismes proactifs c'est que ceux-ci limitent le trafic sur le réseau de façon importante. En effet, la plupart de ces techniques réservent une bande passante maximale pour des flux alors que celle-ci n'est pas toujours utilisée. Il est également possible que le niveau maximum de bande passante utilisée ne soit pas le niveau maximum offert. Certains mécanisme bloquent les flux lorsque 80 % de la bande passante offerte a été atteinte. Dans ces cas, beaucoup de demandes de clients peuvent être bloquées alors que le réseau n'est pas utilisé à son maximum.

Les mécanismes curatifs réagissent une fois que la congestion est arrivée. Cela veut dire que le délai s'est allongé de manière importante et que des paquets ont été détruits. Ces méthodes ne peuvent donc pas garantir la qualité de service pour des applications en temps réel telles que la voix ou la vidéo.

L'ensemble de ces techniques génère deux types de problèmes. Tout d'abord une mauvaise utilisation des capacités du réseau qui pourrait être plus chargé. Mais on peut aussi faire face à des dégradations de services. Ces techniques agissent donc trop tôt ou trop tard. Il est donc important de trouver une méthode agissant entre les deux et pouvant s'adapter au comportement du trafic.

1.2 Objectif

L'objectif de ce projet est de trouver un moyen de prévoir les congestions sur les réseaux MPLS pour utiliser le réseau au maximum afin de ne pas répartir le trafic trop tôt ou trop tard. Cela permettrait en effet d'utiliser un maximum de bande passante en limitant les pertes de paquets ou le délai lors d'un reroutage. L'hypothèse qui est faite ici est qu'il est possible de prévoir cette congestion à l'aide des informations de trafic contenues dans les équipements réseau. Par exemple, le nombre de paquets dans la file d'attente ou le débit demandé. Ces informations sont accessibles via le protocole de gestion de réseaux Simple Network Managment Protocol (SNMP).

L'objectif principal de ce mémoire peut être divisé en trois sous-objectifs :

- Le premier est de réussir à recréer un réseau MPLS sur une plateforme de test avec équipements réels et d'analyser le comportement des congestions. Cette plateforme devra correspondre aux architectures des opérateurs pour mieux s'approcher de la réalité. Il est également nécessaire de créer des congestions avec des outils existants pour s'approcher du comportement des congestions dans les réseaux MPLS.
- Le deuxième sous-objectif consiste à développer un algorithme basé sur les observations précédentes pour prévoir les congestions. Cet algorithme devra être léger pour ne pas surcharger le réseau. Il devra utiliser au maximum les informations disponibles via SNMP afin de pouvoir être déployé aisément.
- Enfin, le troisième sous-objectif consiste à appliquer l'algorithme sur le banc d'essai et dans un simulateur de réseau pour pouvoir valider son comportement et justifier sa conception. Il est également nécessaire de le comparer aux techniques existantes.

1.3 Limites du projet

Le travail effectué dans ce mémoire s'attache à l'amélioration des méthodes d'alerte de la congestion. Il ne traite pas des mesures à prendre pour l'éviter une fois que celle-ci est

détectée. Les mécanismes de TE permettent déjà beaucoup de types de réactions différentes. Le but est ici de trouver une méthode innovante pour prévenir ces mécanismes de l'apparition des congestions.

1.4 Méthodologie

Pour parvenir aux objectifs fixés dans le cadre de ce mémoire plusieurs outils sont mis à disposition. Tout d'abord les équipements du banc d'essai du LAGRIT avec diverses applications nécessaires à l'étude d'une congestion. Ces équipements permettront d'étudier le comportement de la congestion et d'en analyser les répercussions. Le simulateur Network Simulator 2 (NS-2) sera également utilisé dans le cadre de la recherche. Ce simulateur permettra de développer notre mécanisme de prévision de la congestion plus aisément.

CHAPITRE 2

ÉTAT DE L'ART

Ce chapitre présente l'ensemble des connaissances nécessaires à la compréhension de la problématique liée aux congestions dans les réseaux MPLS. Dans la première partie, les concepts généraux sur lesquels se basent le mémoire sont tout d'abord détaillés. Puis, les parties 2.2 et 2.3 présentent les protocoles SNMP et MPLS qui ont étés utilisés comme base pour le mécanisme développé. Enfin, les mécanismes utilisés pour lutter contre les congestions sont étudiés.

2.1 Définition des termes généraux

2.1.1 La bande passante

Dans ce mémoire, un des mots qui revient le plus souvent est « bande passante ». Sans explications supplémentaires, la bande passante peut se rapporter à plusieurs données totalement différentes. Les travaux présentés dans ce mémoire ne parlent que de la bande passante au niveau réseau. La bande passante offerte correspond au nombre d'informations (en bits) par seconde qu'un lien peut supporter. La bande passante utilisée représente le nombre moyen de données qui transitent effectivement dans le lien. L'unité de référence pour la bande passante est le bit. Il est également possible de parler en octets, mais cette unité sert plutôt pour la taille des données transmises. Un octet correspond à huit bits.

2.1.2 La congestion

La congestion est un évènement critique dans le fonctionnement des réseaux. Elle se produit lorsque trop de flux affluent vers une même interface et que le débit nécessaire à ces flux est plus grand que le débit disponible. On dit qu'une interface est congestionnée au moment où la file d'attente est pleine et que des paquets sont détruits. Lorsqu'une congestion se produit,

le délai de transmission s'allonge et certains paquets supprimés entraînent l'exécution de mécanismes de retransmission. Ces paquets peuvent donc empirer l'état du réseau si aucune solution n'est mise en place.

2.1.3 La qualité de service

La demande croissante de nouveaux services tels que la voix dans les réseaux IP actuels a entraîné de nouvelles contraintes dans la gestion des capacités du réseau. La qualité de service (QdS) représente la qualité de transmission d'un flux nécessitant le respect de certains critères. Elle n'est pas quantifiable en tant que tel, mais représente le fait de mettre en œuvre des moyens pour acheminer des services en respectant leur priorité. La qualité de service est représentée par plusieurs critères quantifiables :

- le délai : il représente le temps mis par un paquet pour passer d'un élément réseau à un autre. Il peut-être mesuré de bout en bout (de l'émetteur au récepteur) ou point à point sur un lien réseau (d'un routeur à un autre).
- la gigue : elle représente la variation du délai. Elle est plus généralement mesurée entre un émetteur et un récepteur. La valeur de la gigue est la moyenne de décalage du délai par rapport au délai moyen.
- le taux de perte : il représente le taux de paquets perdus par rapport au nombre de paquets envoyés.

Certains services critiques comme la voix sur IP nécessitent de respecter des contraintes basées sur ces variables (Hersent, Gurle et Petit, 2006). Pour un appel de voix sur IP de bonne qualité il est nécessaire d'avoir un délai inférieur à 150 ms de l'émetteur au récepteur, une gigue inférieure à 20 ms et un taux de perte inférieur à 10 %. Des valeurs supérieures à ces contraintes peuvent causer un décalage, de l'écho, des pertes de phrases ou la perte complète du signal.

2.1.4 La gestion de réseaux

Le terme gestion de réseaux désigne le fait de surveiller et gérer les équipements contenus dans un réseau. Ces équipements peuvent être des équipements réseau tels que des routeurs ou des commutateurs, mais aussi des serveurs ou des ordinateurs personnels. La gestion de réseau a pour but d'analyser le comportement de celui-ci pour aider au débogage et à la prévention des pannes. C'est un aspect important de la vie du réseau qui comprend l'analyse ainsi que la modification des configurations automatisées ou non. La gestion de réseaux est effectuée pour localiser les pannes le plus rapidement possible. Elle permet ainsi de rétablir rapidement les problèmes au sein des réseaux d'opérateurs qui comptent des milliers d'équipements. Le but de la gestion de réseaux est de localiser et de corriger les pannes avant que le client ne puisse ressentir les effets de celles-ci.

2.2 La gestion de réseaux avec SNMP

Le protocole SNMP pour Simple Network Management Protocol est le protocole le plus communément utilisé pour la gestion des réseaux informatiques. Il permet facilement d'obtenir des informations contenues sur des équipements réseau. Ces données peuvent être de diverses natures telles que le taux d'utilisation du processeur, l'état d'une interface ou la version logicielle de l'équipement. L'implémentation du protocole SNMP nécessite plusieurs éléments (*Voir* Figure 2.1) ayant des rôles distincts dans le traitement et l'organisation de ces données (Stallings, 1999).

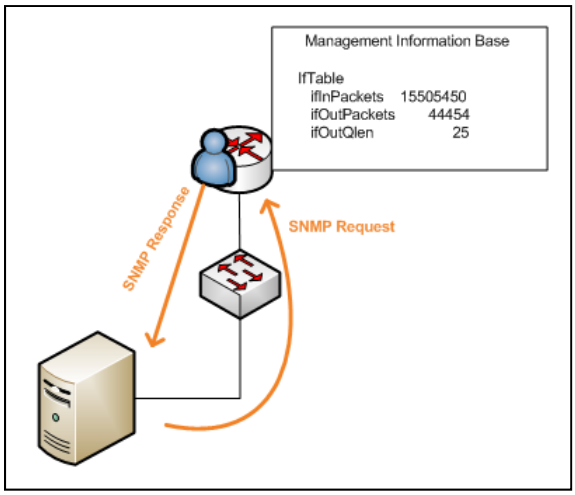

Figure 2.1 Architecture SNMP.

2.2.1 La station de gestion

Une station de gestion est un serveur utilisé pour centraliser la gestion des équipements d'un réseau au travers de SNMP. Les fonctionnalités nécessaires sont les applications pour l'analyse du réseau et pour les corrections de pannes. La station doit avoir une interface réseau par laquelle elle peut contrôler les équipements ainsi qu'être contrôlée par le gestionnaire. Les applications pour l'analyse du réseau permettent généralement une analyse automatique du réseau. Les pannes ou les prémices d'une panne sont rapportées à l'administrateur de réseau qui décide de la marche à suivre.

2.2.2 Agent de gestion

L'agent de gestion est placé sur les équipements qui doivent être surveillés, c'est lui qui permet d'extraire des données de l'équipement et de les transmettre à la station de gestion. L'agent est souvent installé sur les équipements réseau par la configuration d'usine. Sur certains équipements comme les serveurs, il peut être installé par la suite par l'administrateur réseau.

2.2.3 La base de données d'information

La base de données d'informations (Management Information Base ou MIB) permet de hiérarchiser toutes les informations des équipements disponibles avec SNMP. Il existe deux MIB, MIB-I et MIB-II qui est une extension de MIB-I. Dans ces MIB, chaque donnée est représentée par un objet et possède un identifiant unique (Object Identifier ou OID). Ces objets sont hiérarchisés par type et leurs identifiants sont relatifs aux branches dans lesquelles ils se trouvent (*Voir* Figure 2.2). Les informations stockées sont de multiple nature (*Voir* Tableau 2.1). Il existe quatre formes de stockage pour ces valeurs : un compteur (Counter), une jauge (Gauge), un chiffre (Integer) ou du texte (String). Les compteurs et les jauges peuvent être de 32 ou de 64 bits. La particularité du compteur est que celui-ci fait des tours complets; lorsqu'il atteint la valeur maximale il repasse par zéro. La jauge, lorsqu'elle atteint le maximum, ne peut que descendre ou rester au maximum (par exemple : le pourcentage d'utilisation du processeur). Généralement la base de données est contenue dans la station de gestion.

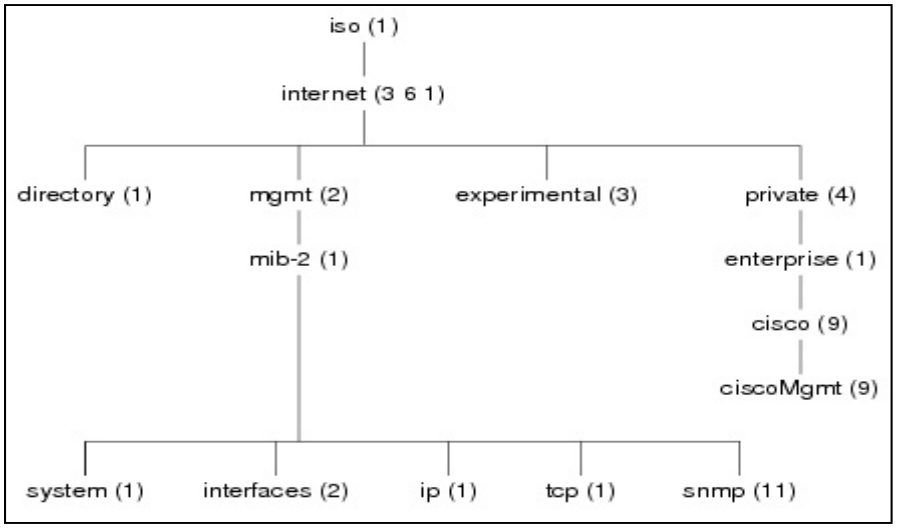

Figure 2.2 Exemple de hiérarchisation.
Extrait du site Internet cisco.com

Tableau 2.1 Exemples d'objets SNMP

Nom	OID	Accès	Type	Description
sysDescr	.1.3.6.1.2.1.1.1	Lecture	String	Texte décrivant le système, contient le nom et la version d'identification matériel
ifNumber	.1.3.6.1.2.1.2.1	Lecture	Integer	Nombre d'interface réseaux
ipOutDiscards	.1.3.6.1.2.1.4.11	Lecture	Counter	Nombre de paquets IP n'ayant été supprimés sans erreurs de traitements (exemple : file d'attente pleine)
ifAdminStatus	.1.3.6.1.2.1.2.2.1.7	Lecture-écriture	Integer	État d'une interface réseau. Trois possibilités : (1) up, (2) down, (3) testing.

2.2.4 Le protocole SNMP

Ce protocole permet la communication entre les agents de gestion installés sur les équipements réseau et la station de gestion. SNMP est situé à la couche application du modèle TCP/IP. Il se base sur des messages UDP pour échanger des informations. SNMP a été conçu pour récupérer les informations des équipements du réseau avec une possibilité de modifier certains paramètres.

Le protocole SNMP utilise trois principaux types de messages pour échanger des données entre la station de gestion et les agents :

- GET : est envoyé par la station de gestion à l'agent pour récupérer les valeurs d'objets se trouvant dans l'équipement.
- SET : sert à paramétrer la valeur d'un objet dans un équipement. Le message SET est initié par la station de gestion.
- TRAP : est un message envoyé par l'agent à la station de gestion pour la notifier d'un événement. Les TRAP sont généralement des alertes qui sont paramétrées dans l'agent de la station.

2.3 MPLS

MPLS est un protocole de niveau 2.5, il se situe entre la couche liaison de données et la couche IP. Comme son nom l'indique, MPLS est un protocole de commutation de paquets. Contrairement à IP qui traite les flux paquets par paquets, le protocole MPLS traite les flux dans leur ensemble. En effet, dans les réseaux IP, les paquets appartenant aux même flux peuvent emprunter des chemins différents dans le réseau et ceci de manière aléatoire. Dans les réseaux MPLS chaque flux est transmis au travers du réseau en suivant un chemin réservé à l'avance. Le routage n'est pas effectué en fonction de l'adresse IP du destinataire, mais grâce à une table de commutation stockée dans chaque routeur MPLS.

2.3.1 Architecture MPLS

Les réseaux MPLS sont constitués de routeurs de bordures (Label Egress Router ou LER) qui font la liaison entre les réseaux IP et les réseaux MPLS (*Voir* Figure 2.3). Ces routeurs s'occupent de mettre ou d'enlever l'en-tête MPLS sur les paquets. À l'intérieur du réseau, on retrouve les routeurs de cœurs (Label Stack Router ou LSR) qui ne s'occupent que de commuter les paquets MPLS.

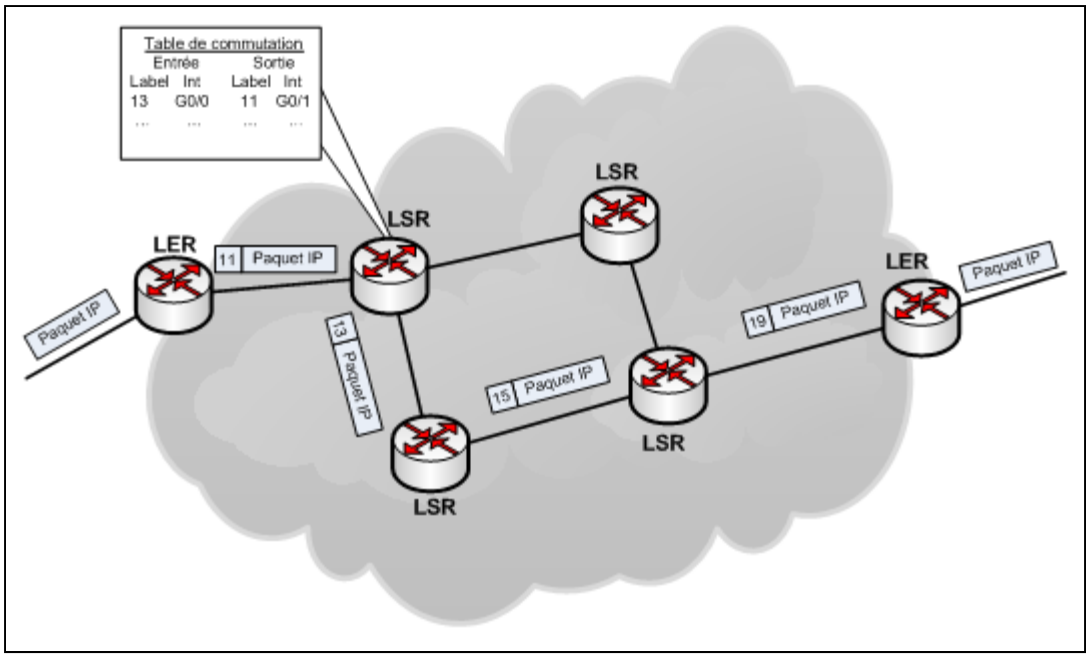

Figure 2.3 Architecture MPLS.

2.3.2 Établissement d'un lien

Pour établir un lien à travers un réseau MPLS il est nécessaire de suivre plusieurs étapes. Tout d'abord, le LER émet un message PATH à l'entrée du réseau et les routeurs LSR du réseau MPLS trouvent, via le protocole de routage, la route par laquelle passer pour atteindre la sortie du réseau MPLS. Ils transfèrent ce message PATH au routeur LER de sortie du réseau. Ensuite le LER renvoie un message RESV pour confirmer la réservation du chemin d'accès à travers le réseau. Pour définir les numéros de label qui seront utilisés, le protocole de distribution de label (Label Distribution Protocol ou LDP) est utilisé. Les chemins MPLS sont à sens unique il est donc nécessaire de refaire la manoeuvre dans l'autre sens si la connexion est établie pour un flux à double sens. Le nom spécifique aux chemins MPLS est « Label Switched Path » (LSP).

2.3.3 Regroupement des flux

Les réseaux MPLS ont une particularité qui permet le regroupement des flux de mêmes types dans une classe d'équivalence (Forward Equivalency Class ou FEC). Le regroupement en classe permet aux réseaux de limiter le nombre d'entrées dans les tables de commutations et de rendre l'acheminent des paquets plus rapides. Pour que deux flux appartiennent à la même classe il faut que ceux-ci suivent le même trajet dans le réseau et soient de même nature. Par exemple deux flux voix sur IP d'un réseau A à un réseau B.

2.3.4 Ingénierie de trafic

De façon native MPLS n'intègre aucun mécanisme de qualité de service. Cependant, il peut être utilisé avec les mécanismes IntServ et DiffServ. Ces mécanismes permettent la différenciation du trafic en différentes classes de service. L'ingénierie de trafic (Traffic Engineering ou TE) a été développée pour permettre à MPLS d'améliorer la qualité de service. Les techniques de TE sont multiples, certaines répartissent au mieux les charges dans les réseaux et d'autres réservent des bandes passantes minimales aux flux prioritaires.

L'idée la plus importante pour les réseaux MPLS avec TE est que le trafic des données IP peut passer par des chemins qui ne sont pas forcément les routes définies par des algorithmes de routage selon un plus court chemin. En effet avec le routage basé sur les contraintes (Constraint Base Routing) il est possible de réserver un chemin pouvant satisfaire les spécificités du trafic à router. Les contraintes qu'il est possible de prendre en compte sont : la bande passante, le délai et la gigue.

2.4 Mécanismes de lutte contre la congestion

Les mécanismes pour lutter contre la congestion peuvent être classés en deux types. Les premiers sont les mécanismes curatifs. Ceux-ci entrent en action une fois qu'une congestion a

été détectée. Pour éviter d'avoir des congestions dans le réseau, il existe des solutions proactives. Celles-ci acheminent le trafic dans le réseau de manière à ce que les congestions n'aient que très peu de chance de se produire.

2.4.1 Mécanismes curatifs

Il existe beaucoup de mécanismes curatifs qui peuvent s'appliquer à MPLS. On distingue trois types de comportements : la redirection complète du tunnel (Zhiqun, Xu et Wei, 2001), un partage de charge (Salvadori et Battiti, 2003) ou une diminution de la vitesse d'envoi (Dekeris et Narbutaite, 2004). Certaines méthodes utilisent plusieurs techniques selon les caractéristiques du trafic. La congestion est détectée via plusieurs données. Certains articles utilisent la bande passante moyenne ou restante sur un lien et d'autres la moyenne des paquets perdus. Mais dans la plupart des recherches, les niveaux utilisés lors des tests ne sont pas spécifiés. Les auteurs laissent le soin aux administrateurs de réseaux de les définir selon leurs besoins. Ils ne sont donc pas exclusivement curatifs, mais peuvent être utilisés comme mécanismes proactifs si les niveaux sont choisis en ce sens. L'inconvénient de ce type de solution est la définition des niveaux d'agissements. Si ceux-ci sont trop bas, alors le mécanisme agira de manière proactive et n'utilisera peut-être pas la bande passante de manière efficace. Si les niveaux sont trop hauts, la congestion pourrait perturber le réseau énormément ou ne pas être détectée par le mécanisme. Ces solutions nécessitent donc une très bonne connaissance de tout le réseau et le choix des niveaux doit être effectué de façon très précise.

En plus des mécanismes d'ingénierie de trafic il existe des mécanismes qui agissent directement sur les files d'attente. L'un des mécanismes les plus connus est Random Early Detection (RED). RED a été normalisé par la RFC2309 (B. Braden, 1998).Ce mécanisme supprime des paquets ou les marque lorsqu'un certain niveau de criticité est atteint. Pour les systèmes RED, l'administrateur de réseaux doit définir deux limites pour la détection des congestions. Ces limites sont définies en fonction de la taille des files d'attente.

Généralement les limites définies sont de l'ordre de 50 et 80 % de la taille totale de la file d'attente. Lorsque la moyenne du nombre de paquets atteint la limite basse, des paquets sont supprimés au hasard dans la file d'attente. Plus la moyenne augmente et plus le pourcentage de paquets supprimés augmente jusqu'à atteindre un maximum défini. RED rejette les paquets au hasard, mais en prenant garde à supprimer les paquets sur plusieurs sessions différentes et non pas uniquement la même. Si la limite haute est dépassée alors RED rejette tous les nouveaux paquets qui arrivent. L'inconvénient principal de ce mécanisme est qu'il ne limite pas la congestion, il ne fait que supprimer des paquets pour éviter que la file d'attente ne soit pleine. En supprimant les paquets, certains flux peuvent réémettre les paquets perdus et donc augmenter le phénomène de congestion.

2.4.2 Mécanismes proactifs

Dans l'article (Oulai, Chamberland et Pierre, 2007) l'auteur fait la synthèse des dispositifs qui existent pour le contrôle de congestion proactif à partir de routage basé sur des contraintes. Les dispositifs de ce type peuvent prendre des contraintes uniques ou multiples. Ces contraintes sont en général choisies entre le délai, la gigue, le taux de perte de paquets et la bande passante. Beaucoup de documents traitent de la bande passante, car celle-ci est plus facilement quantifiable. En effet, savoir quel est le délai ou la gigue d'un lien nécessite l'envoi de paquets de contrôle. L'information de débit est quant à elle stockée sur le routeur. Elle ne nécessite donc aucune modification du comportement de l'équipement pour être relevée.

Il existe deux types d'architectures, centralisée ou distribuée. Pour le mode centralisé, un serveur qui connaît l'architecture du réseau est utilisé pour bâtir les LSP (A. Capone, L. Fratta et Martignon, 2003). Alors qu'en mode distribué, c'est le LER d'entrée qui gère la création des routes (Reeves et Salama, 2000).

En plus de l'architecture, il existe plusieurs choix quant à la façon de procéder à l'ajout d'un LSP. Soit on choisit de ne jamais déplacer les LSP existants. Soit, pour optimiser au maximum la répartition du réseau, on admet que l'on peut rerouter certains LSP. Dans ce cas les LSP de priorité supérieure peuvent remplacer des LSP de moindre priorité.

2.4.3 Nouveau mécanisme de prévention

Cette section présente un nouveau mécanisme de prévention de la congestion : les systèmes de notifications pré-congestion (Pre-Congestion Notification ou PCN). Cette approche est récente, les premiers articles relatant PCN sur une architecture MPLS datent de 2008. L'IETF (Internet Engineering Task Force) a par ailleurs publié la RFC 5559 (Eardley, 2009) en juin 2009 pour les réseaux PCN. Dans leur article paru en Juin 2008, Xinyang Zhang et Anna Charny évaluent les performances des solutions PCN sur MPLS. Cet article est très intéressant pour toute personne souhaitant approcher cette nouvelle solution.

2.4.3.1 Architecture et hypothèses

Le mécanisme PCN est ajouté en complément de l'architecture MPLS. Les LSR ne servent plus simplement à commuter les paquets en fonction de leur Label. En effet ces routeurs ont en plus la charge de marquer les paquets. Ceux-ci sont marqués selon deux limites configurées par l'administrateur réseau. La première limite est la limite de contrôle d'admission (Admission Control ou AC) et la seconde et celle de terminaison des flux (Flow Termination ou FT).

La méthode PCN nécessite le respect d'une contrainte importante. Seuls des flux ne pouvant pas s'auto ajuster à la bande passante disponible peuvent être traités. Les solutions PCN ne peuvent donc pas traiter des flux TCP, mais sont plutôt réservées à des services tels que la voix sur IP ou la vidéoconférence.

2.4.3.2 Fonctionnement

Dans l'article (Arumaithurai *et al.*, 2009) les routeurs à l'intérieur du réseau utilisent les deux paramètres. Ils peuvent être en marquage AC ou FT (*Voir* Figure 2.4).

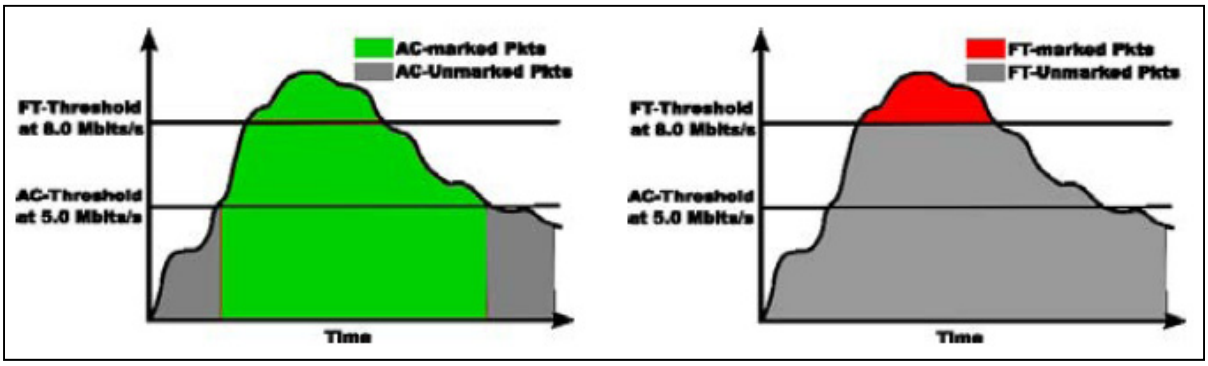

Figure 2.4 Limites de marquage du mécanisme PCN.
Extrait de (Arumaithurai *et al.*, 2009)

Le mode AC est déclenché dès que le trafic dépasse la limite définie. Lorsque ce mode est en cours, le routeur concerné marque tous les paquets. Si une demande de nouveau trafic est envoyée et passe par ce routeur, le paquet est donc marqué. Le routeur de sortie du réseau reçoit alors une demande de nouveau flux qui est marquée par le mode AC. Celui-ci envoie donc une information au routeur d'entrée pour lui signaler que le flux n'est pas accepté, car la limite est dépassée. Le routeur d'entrée doit donc choisir un nouveau chemin par lequel passer.

Lorsque la limite de terminaison de flux est dépassée, le marquage change. Le routeur effectue alors un « excess marking ». Cela signifie que seuls les paquets entraînant le dépassement de la limite sont marqués. Tous les paquets qui auraient été marqués « AC » se voient enlever leur marquage. Le routeur de sortie du réseau décide alors combien de flux doivent être coupés en calculant le ratio des paquets marqués et non marqués.

Cette nouvelle méthode permet d'éviter les congestions. Il est cependant nécessaire d'étudier le comportement du réseau de manière raffinée pour choisir des niveaux de fonctionnement corrects. Une faille dans cette méthode est possible si un routeur marque ses paquets en AC et que ceux-ci passent dans un autre routeur qui marque en FT. Dans ce cas les paquets marqués AC seront démarqués et le premier passage en voie de congestion sera ignoré par le réseau.

2.5 Conclusion

Dans ce chapitre les méthodes de lutte contre la congestion ont été présentées. Ces différentes approches montrent deux types de problèmes. Soit elles essayent du mieux qu'elles peuvent d'éviter la congestion et ne permettent pas une utilisation optimale des liens. Soit elles attendent la congestion pour réagir et des paquets sont perdus. Dans ces deux cas, le réseau n'est pas utilisé de manière optimale et reste en proie à des pertes de paquets ou à une sous utilisation.

Pour le moment aucune solution appliquée aux réseaux MPLS ne permet de prévoir les congestions et de réagir uniquement lorsque celles-ci sont prévues. L'étude du comportement de la congestion est donc une nécessité. Cela permettra de savoir si une prévision est possible et si celle-ci donnerait de meilleurs résultats que les techniques existantes.

CHAPITRE 3

ÉTUDE DU COMPORTEMENT D'UNE CONGESTION

Dans ce chapitre le but est d'analyser le comportement d'une congestion dans le réseau. Cela doit permettre d'analyser quelles sont les informations qui peuvent être utilisées pour améliorer les solutions existantes. Les congestions seront étudiées sur un banc de test réel ainsi que sur un simulateur. Il est nécessaire d'effectuer cette comparaison pour savoir si le comportement du réseau est identique et si un rapprochement peut-être fait entre ces deux études. En effet, il est plus aisé de développer une solution sur simulateur. Il est donc nécessaire de savoir si une solution développée sur simulateur peut produire un effet identique sur un banc de test réel. La première partie présentera l'étude sur le banc de test réel. La deuxième partie présentera les tests sur simulateur. La première observation suite à ces tests consistera à valider l'utilisation du simulateur pour le développement d'une solution. Enfin, un regroupement des résultats sera effectué pour mettre en avant les informations avec lesquelles il est envisageable de développer une méthode de prédiction.

3.1 Mesures en environnement réel

3.1.1 Matériel

Le matériel mis à disposition pour les mesures en environnement réel est celui du LAboratoire de Gestion de Réseaux Informatiques et de Télécommunications (LAGRIT) situé à l'École de Technologie Supérieure de Montréal.

L'ensemble du matériel est composé de cinq routeurs Cisco 2821. La version IOS 12.4 est installée sur ces équipements. Ces routeurs possèdent chacun trois interfaces routables Gigabit Ethernet (1 Gbits/s) et neuf interfaces Fast Ethernet (100Mbits/s) fonctionnant comme un commutateur. Le banc de test comprend également quatre commutateurs Cisco Catalyst 2950. La version d'IOS configurée est la 12.1. Ces commutateurs disposent de 24 ports Fast Ethernet (100Mbits/s) et deux ports Gigabit Ethernet (1 Gbits/s). Le banc d'essai

possède également dix serveurs Dell PowerEdge 1850 avec des processeurs de 3Ghz disposant de quatre interfaces Fast Ethernet (100Mbits/s). Ces serveurs ont deux systèmes d'exploitation, Windows server 2003 et CentOs version 4.4. Enfin, le LAGRIT dispose d'un serveur de voix sur IP Cisco Call Manager avec cinq téléphones IP. Le réseau du banc d'essai est complètement séparé du réseau de l'ÉTS ainsi que de l'Internet. Il est ainsi impossible de menacer l'intégrité du réseau de l'ÉTS avec une faute de configuration ou une faille dans les applications installées.

3.1.2 Applications

Sur le banc de test, en plus des équipements, trois applications ont été installées. Celles-ci sont déterminantes pour réaliser les expériences et analyser les résultats.

3.1.2.1 Application Quagga

Quagga est un logiciel libre développé par Kunihiro Ishiguro (Ishiguro, 2010) qui s'installe sur plusieurs distributions Linux. Elle permet de transformer un serveur en routeur. En effet Quagga permet le routage statique ou dynamique avec divers protocoles tels que RIP (Routing Information Protocol) ou OSPF (Open Shortest Path First). Ces protocoles sont directement inclus dans Quagga et il n'y a aucune modification à effectuer si l'ont veux disposer d'un des six protocoles de routage disponibles. Il est donc possible grâce à Quagga d'augmenter le nombre de routeurs utilisés pour la plateforme de test en utilisant des serveurs. Cette application permet également de se connecter via Telnet sur une interface « commande en ligne » (Command Line Interface ou CLI) semblable à celle utilisée dans les équipements Cisco. Lors de l'administration sur un serveur il est possible de configurer Quagga directement via le fichier de configuration ou via une interface CLI locale. L'application a été installée sur trois serveurs Dell présents sur le banc d'essai afin de remplir le rôle de passerelle des réseaux locaux. La version de Quagga installée est la 0.99.12 datant du 10 mars 2009.

3.1.2.2 Application Iperf

Iperf est développée par le National Laboratory for Applied Network Research (NLANR, 2010). C'est une application qui permet de générer des trafics réseau entre une source et une destination. Il génère des rapports sur les paquets perdus ainsi que sur la vitesse d'émission moyenne. Iperf possède un grand nombre de paramètres modifiables par l'utilisateur. Il est possible de choisir la durée d'émission, la taille des paquets à envoyer ainsi que la bande passante à utiliser. Il est également possible de choisir un mode d'émission UDP ou TCP. Cette application est très polyvalente, elle permet de mesurer les capacités maximales d'un lien réseau mais aussi de simuler un trafic bien précis. Cette application est très pratique pour les créations de congestions, car elle permet de générer plusieurs clients simultanément depuis un même serveur. C'est Iperf qui servira intégralement pour générer des congestions dans le réseau. Ce choix a été fait, car Iperf est gratuit et s'installe facilement sur une multitude de systèmes tels que Windows ou Linux.

3.1.2.3 Application Lagrit Poller

J'ai spécialement développé l'application Lagrit Poller (LP) pour les besoins de ce mémoire. En installant les applications de surveillance CACTI (Kundu *et al.*, 2009) et Real Traffic Grabber (RTG) (Beverly, 2002), les premiers tests ont montré que celles-ci ne permettaient pas la personnalisation des informations recueillies. Le niveau des files d'attente sur les liens et le taux de processeur utilisé par les routeurs ne pouvaient pas être relevés. En plus de ne pas remplir les besoins de ce mémoire à 100 %, ces logiciels sont difficilement modifiables car ils contiennent beaucoup de fichiers et de lignes de codages. Il était donc plus aisé de développer une nouvelle application en prenant en compte les besoins pour ce projet.

LP a été développée en PHP et permet de choisir les informations à relever ainsi que l'intervalle de mesure entre lesquels elles sont recherchées. L'application utilise les fonctions

de requêtes SNMP contenues dans PHP. Ces informations permettent d'obtenir les données des agents situés sur les routeurs. L'application interagit avec une base de données MySQL. Celle-ci contient la liste des objets et des équipements. Cette liste est entièrement définie par l'utilisateur. Le logiciel LP s'appuie sur ces informations pour interroger les équipements. Les informations recueillies sont stockées de manière brute. Il n'y a pas de calculs effectués pour les mettre en forme. Cela permet de traiter les résultats par la suite et de garder une trace de valeurs obtenues avec les requêtes de base. L'annexe III contient un supplément d'information pour la compréhension du fonctionnement de Lagrit Poller, notamment une figure récapitulative.

LP permet aussi de tracer des graphiques définis par l'utilisateur concernant des requêtes SNMP spécifiques et sur un intervalle de temps donné. Tous les résultats des requêtes effectuées par l'application sont sauvegardés et indexés dans la base de données MySQL. C'est donc un bon outil pour relever les résultats des expériences et les stocker. Pour tracer les graphiques, LP utilise l'application Artichow (Artichow, 2006). Artichow est une application dont les droits d'exploitation et de modification sont sous licence Creative Commons. Cela signifie qu'il est possible de l'utiliser ou de le modifier à des fins non commerciales.

3.1.3 Configuration du banc d'essai

La figure 3.1 représente la topologie configurée sur le banc d'essai. La particularité de ce banc de test est que les équipements sont accessibles via un réseau d'administration séparé physiquement du réseau où le trafic de test passe. Cela permet de garder le contrôle sur les équipements même si un test de bris de lien ou de congestion est effectué. Le réseau d'administration n'apparaît pas dans la figure pour en simplifier la compréhension.

Cette topologie est typique des réseaux MPLS d'opérateurs. En effet ces architectures comprennent les routeurs situés chez les clients, les « Customer Edge » (CE). Ces routeurs

appartiennent généralement aux opérateurs et permettent de connecter le client au cœur du réseau. Ceux-ci sont reliés à des routeurs plus gros communément appelés les « Provider Edge » (PE). Ces routeurs sont situés dans le cœur du réseau de l'opérateur. Ce sont généralement des équipements performants sur lesquels sont connectés plusieurs CE.

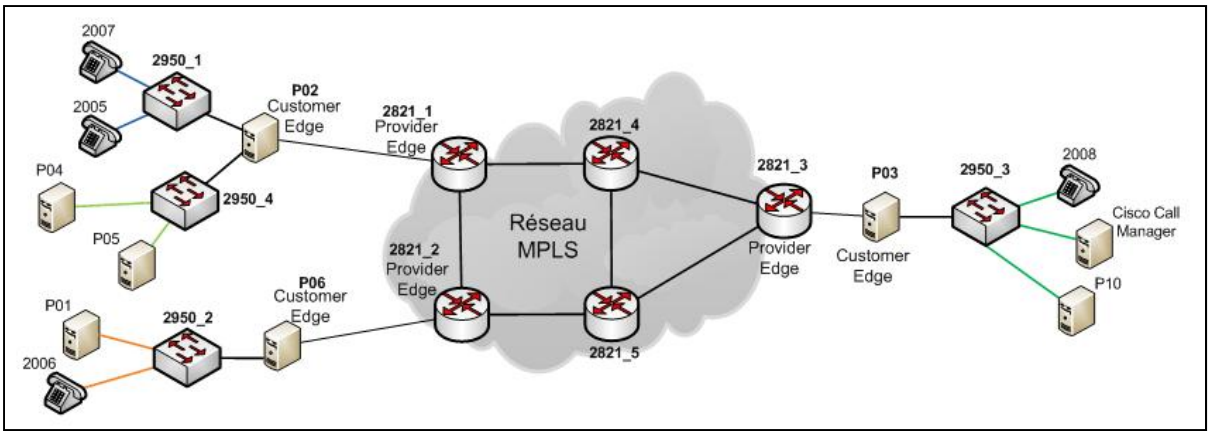

Figure 3.1 Topologie du banc de test.

L'architecture configurée dans le banc de test représente un client avec trois réseaux séparés en termes physiques. Ces réseaux pourraient être séparés par des dizaines ou des centaines de kilomètres. Mais dans le réseau MPLS, ils sont reliés comme s'ils étaient à quelques mètres. La principale différence entre cette infrastructure et celle d'un réseau d'opérateur est le délai. En effet, si les sites sont très éloignés le délai sera plus important que sur le banc d'essai. Cependant, cela ne devrait pas changer le comportement des équipements lors des congestions.

Des LSP sont préconfigurés pour que le réseau derrière P02 et celui relié à P06 passent par un goulot d'étranglement pour arriver au réseau connecté à P03. Le chemin pour aller de P02 à P03 passe par les routeurs 2821_1, 2821_4, 2821_5 et 2821_3. Le chemin utilisé pour faire le trajet de P06 à P03 passe par 2821_2, 2821_5 et 2821_3. Les bandes passantes des liens sont de 10 Mbits/s pour tous les liens entre les routeurs. Cela permet de générer des congestions plus facilement et sans trop surcharger les serveurs. La congestion doit se

manifester sur l'interface Gigabit Ethernet 3 du routeur 2821_5. Pour mieux comprendre les méthodes de configuration, les annexes I et II présente des extraits des configurations du routeur 2821_5 et du serveur P02.

3.1.4 Étude de la congestion

Pour étudier les effets de la congestion, plusieurs tests ont été réalisés avec des données différentes. Pour illustrer ces tests et justifier les observations qui en découlent, un exemple de scénario est présenté ici. Le test commence avec un client de 6Mbits/s (envoyant des paquets de 1200 octets) généré par Iperf. Ce client est émis depuis le serveur P01 vers le serveur P10. Il reste constant durant toute la durée du test. Ce flux est créé pour limiter le nombre de clients à générer et raccourcir la durée des relevés. Puis viennent s'ajouter des clients de 100 Kbits/s espacés de 0,4 seconde. Ils sont émis depuis P04 vers le serveur P10. Les espacements sont réalisés via un script écrit en bash qui appel un nouveau client Iperf toutes les 400 ms. Cela engendre théoriquement une congestion à 40 clients soit 8 secondes après la connexion du premier client.

Le but est ici d'analyser le comportement du réseau au moment de la congestion. Il n'est donc pas nécessaire de la maintenir plus de quelques secondes. La simulation est arrêtée 5 secondes après les premières pertes de paquets. L'analyse doit concerner les quelques secondes qui précédent la congestion pour voir s'il existe des signes avant-coureurs.

3.1.5 Méthode d'analyse

Pour pouvoir analyser les données, il est nécessaire de les recueillir avec l'application Lagrit Poller. La première étape consiste à rassembler plusieurs informations pour voir lesquels peuvent être utilisées pour prévenir une congestion possible.

Les informations relevées sont des objets SNMP. Le tableau 3.1 présente la liste des informations de trafic recensées pour analyser le comportement de la congestion sur le banc d'essai. Les informations stockées ont été sélectionnées parmi une liste de plus d'un millier d'objets SNMP. Elles ont été sélectionnées, car elles contiennent des données d'informations significatives lors des congestions comme la vitesse d'émission, la perte de paquets ou le remplissage des files d'attente. Les premiers tests pour l'étude de la congestion ont enveloppé plus d'objets, mais l'analyse des résultats a permis de limiter le nombre d'informations à observer. Ces informations sont uniquement recensées sur le routeur 2821_5 qui est le point de congestion. Les tests ont montré que les autres équipements ne sont pas directement impactés lors de la congestion ce qui justifie ce choix.

Tableau 3.1 Données recencées lors des tests

Objet	OID	Type	Description
ifOutUcastPkts	.1.3.6.1.2.1.2.2.1.17.2	Compteur	Nombre de paquets envoyé en sortie pour des flux Unicast
cpmCPUTotal5secRevi	enterprises.9.9.109.1.1.1.1.6.1	Jauge	Moyenne du taux d'utilisation de processeur durant les 5 dernières secondes. Ceci est un Objet spécifique à Cisco.
ifInOctets.1	.1.3.6.1.2.1.2.2.1.10.1	Compteur	Nombre d'octets entrants dans l'interface 0/1
ifInOctets.12	.1.3.6.1.2.1.2.2.1.10.12	Compteur	Nombre d'octets entrants dans l'interface 0/3/0
ifOutQLen.2	.1.3.6.1.2.1.2.2.1.21.2	Jauge	Nombre de paquets dans la file d'attente de sortie de l'interface 0/1
ifOutDiscards.2	.1.3.6.1.2.1.2.2.1.19.2	Compteur	Nombre de paquets supprimés en sortie de l'interface 0/1
ifOutOctets.2	.1.3.6.1.2.1.2.2.1.16.2	Compteur	Nombre d'octets sortants de l'interface 0/1

3.2 Mesures sur simulateur

Le logiciel utilisé pour réaliser les tests est Network Simulator 2 (NS-2). Il a été choisi, car il est plus largement utilisé dans le domaine de la recherche. On retrouve le protocole MPLS ainsi que le protocole de réservation de ressources RSVP-TE (Adami *et al.*, 2007). Ces tests doivent permettre de voir s'il existe une vraie similarité entre le simulateur et les conditions réelles. Cela permet par exemple d'affiner les mesures pour analyser le comportement des files d'attente de manière plus détaillée sur le simulateur. L'avantage est également qu'il est plus facile de développer un nouvel algorithme sur un simulateur avant de le transposer sur le banc d'essai.

3.2.1 Présentation de NS-2

Ce simulateur utilise deux langages de programmation. Le C++ pour les configurations globales du logiciel qui changent rarement. Le TCL pour les configurations qui ont tendances à changer rapidement et la rédaction des scénarios de test. Les deux langages sont interfacés via des classes comprises dans le code source du programme. Certaines variables sont de ce fait disponibles en TCL et en C++.

3.2.2 Architecture

L'architecture mise en place sur le simulateur NS-2 (*Voir* Annexe IV) est semblable à celle configurée sur le banc de test. Elle compte neuf noeuds dont quatre jouent le rôle d'extrémités du réseau MPLS. La vitesse de chaque lien est de 10 Mbits/s et le délai de ceux-ci est de 10 millisecondes. La bande passante a été choisie pour simplifier le traitement des simulations ainsi que le taux de données à analyser. Pour une plus grande bande passante il faudrait des paquets plus gros et émis plus souvent ce qui ralentirait beaucoup la simulation. C'est donc pour avoir des simulations plus rapides et simplifier l'analyse des données que cette vitesse a été retenue. Le routage est effectué de façon dynamique.

La configuration MPLS utilisée est semblable à celle du banc d'essai. Les routeurs du réseau sont configurés en nœuds MPLS avec le module inclus de base dans le simulateur NS-2. Le module de réservation de ressource et d'ingénierie de trafic n'a pas été utilisé pour les tests finaux. En effet, il ne permet pas de nouvelles fonctionnalités à part des messages échangés entre les équipements et est à l'origine de plusieurs bogues de l'application. Il a donc été désinstallé pour l'étude de la congestion.

Pour pouvoir comparer les résultats, la configuration des flux a été faite de manière semblable au banc de test. Deux noeuds (n0 et n1) envoient des données vers deux

destinations différentes (n7 et n8) et les flux prennent par défaut une portion de lien en commun (n5 vers n6). Pour congestionner le lien, il faut que l'ensemble des deux flux soit supérieur à la bande passante disponible. Il reste cependant un chemin alternatif que le noeud zéro peut emprunter pour éviter la congestion. La figure 3.2 représente l'architecture configurée.

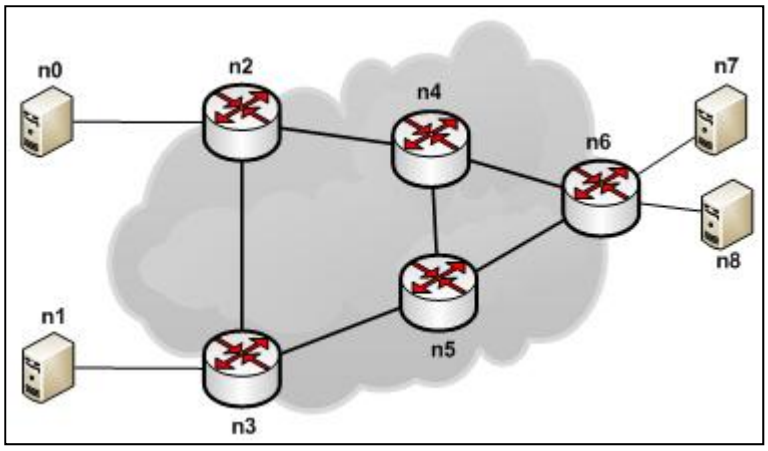

Figure 3.2 Architecture de test sur NS-2.

3.2.3 Étude de la congestion

Comme pour les mesures effectuées avec le banc d'essai, la congestion est testée avec un trafic constant de 6Mbits/s (avec des paquets de 1 220 octets). Pour saturer le lien on ajoute des clients de 100Kbits/s toutes les 0,4 secondes. Ces clients envoient des paquets de 300 octets. Le nombre de clients théoriquement suffisant à la congestion est de 40. Dans le cadre de cette expérience, 50 clients sont envoyés pour une durée de 25 secondes chacun afin de dépasser le point de congestion.

3.3 Résultats

3.3.1 Résultats du banc d'essai

Les figures 3.3 et 3.4 représentent les données recueillies lors de ces tests. L'intervalle entre chaque relevé est de 200 ms pour chacune de ces données. La valeur des files d'attente représente le nombre de paquets dans la file au moment de la mesure. Les valeurs de bandes passantes sont calculées en convertissant le nombre d'octets relevés dans un intervalle de mesure en Mbits/s. Le nombre de paquets supprimés est le nombre de paquets perdus pendant l'intervalle de mesure.

Les relevés montrent que la congestion commence à 28,4 secondes avec un débit reçu sur l'interface de 10,32 Mbits/s et une transmission en sortie de 10,23 Mbits/s. La congestion intervient 14 secondes après le premier client de 100 Kbits/s ajouté au réseau. Le nombre de paquets supprimés dans ce premier intervalle est de 8 paquets. En tout, durant la phase de congestion, 3 775 paquets ont été supprimés sur 29 512 paquets générés, soit une moyenne de 12,8 % de perte. Le nombre maximal de paquets supprimés en un intervalle est de 135 à 33,8 secondes soit 5,4 secondes après le début de la congestion.

La figure 3.3 montre une évolution du trafic ponctuée de variations aléatoires dues à la variation du délai lors de l'émission. Une partie de ces variations est également due au fait que les clients Iperf ne restent pas stables lorsqu'ils sont plus de 20 en simultanés sur un serveur. La figure 3.4 montre que la valeur de la file d'attente ne bouge pas tant que la congestion ne s'est pas manifestée. En effet on voit les premiers paquets de la file d'attente après que des paquets soient détruits. Ceci peut être dû au fait que le relevé de la file d'attente représente la valeur au moment de la requête. Si elle fluctue beaucoup et passe souvent par zéro il se peut que les relevés ne révèlent pas la nature du comportement de la file. Cependant, des tests avec des mesures plus rapprochées (50ms) ont démontré que le

comportement était environ identique. Les seuls moments où des chiffres supérieurs à zéro ont été obtenus résident à un maximum de 400 ms de la première suppression de paquets.

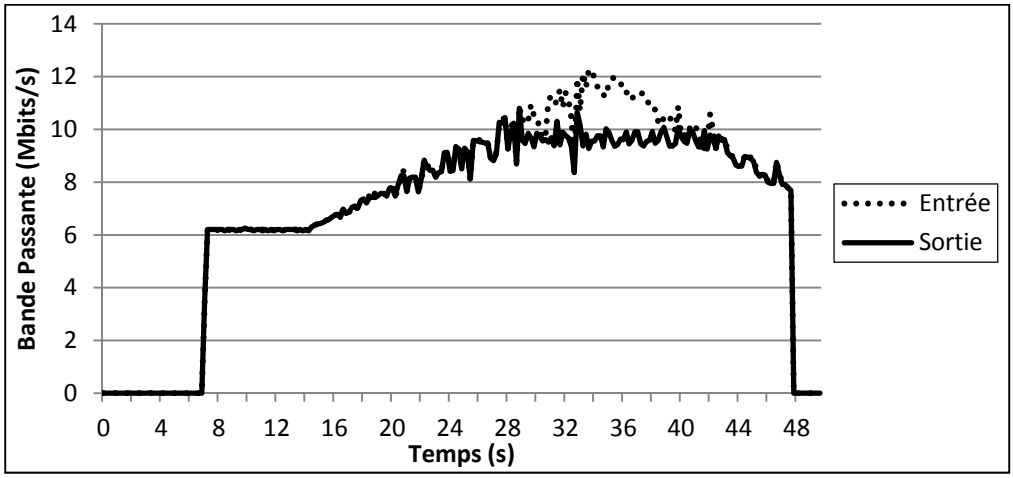

Figure 3.3 Comparaison du trafic en entrée et sortie de l'interface.

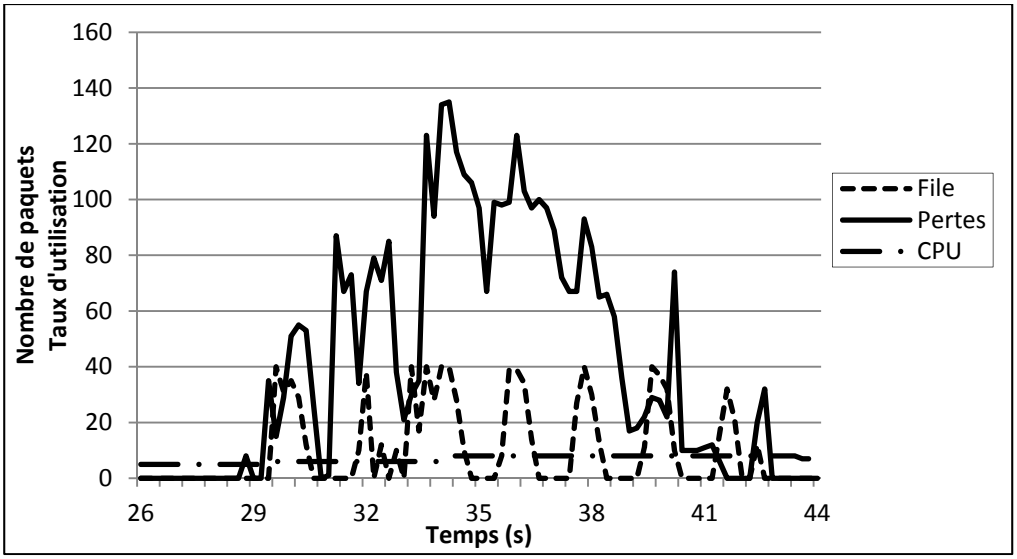

Figure 3.4 Comportement de la file d'attente, suppression des paquets et charge processeur.

Le taux d'utilisation du processeur est très bas. Ceci est principalement dû au fait que la capacité des liens n'est pas au maximum physique de l'interface. Avec des liens à 1 Gbits/s le taux de charge serait plus élevé. Cependant pour générer 1Gbits/s de données il faudrait

plus de serveurs pour créer la congestion et le LAGRIT a mis à disposition sept serveurs pour ce projet, dont trois en routeurs et un en tant que station de gestion. Il faudrait donc plus de serveurs ou des serveurs plus performants.

Les mesures avec un cycle plus court, même si elles peuvent sembler nécessaires, sont moins précises. En effet, comme le délai des requêtes SNMP varie, plus on rapproche les mesures et plus cette variation de délai est perturbante. Avec des mesures plus courtes, la variation de la bande passante calculée est beaucoup plus aléatoire et il est difficile d'obtenir des résultats cohérents.

3.3.2 Résultats sur simulateur

Les figures 3.5 et 3.6 présentent les résultats obtenus lors du test sur le simulateur. Les données sont également recueillies toutes les 200 ms. Les informations de bande passante et de paquets supprimés sont calculées de la même manière que pour les calculs en banc d'essai. Les données de la file d'attente représentent le nombre de paquets dans la file au moment du relevé. La seule donnée qui n'est pas présentée par rapport aux données recueillies sur le banc d'essai est la charge du processeur qui n'est pas disponible sur NS-2.

Les relevés d'informations sur la bande passante montrent que la première suppression de paquets se produit avec une bande passante de 9,7 Mbits/s à 18,8 secondes du début de l'expérience. La congestion intervient donc 13,8 secondes après l'émission du premier client de 100 Kbits/s. Au point de congestion, la file contient 28 paquets et neuf paquets sont supprimés. Neuf secondes avant ce point de congestion la file d'attente contient déjà des paquets. Durant la congestion 59 095 paquets ont été envoyés et 7 947 paquets sont supprimés, soit un taux de perte de 13,4 %. Le nombre maximal de paquets supprimés en 0,2 seconde est de 153 paquets à 29 secondes soit 9,8 secondes après le début de la congestion.

Ces tests montrent que sur le simulateur il serait possible de prévoir la congestion grâce au comportement de la file d'attente. En effet en approchant de la congestion les relevés n'affichant aucun paquet dans la file sont de plus en plus rares. Proche de la congestion tous les relevés mentionnent des paquets dans la file et le nombre de ceux-ci croît rapidement. Le niveau d'utilisation de la bande passante atteint montre que la congestion s'approche des 100 % de la bande passante. Le comportement du simulateur est identique à la définition théorique des congestions : la file d'attente se remplit, puis, lorsqu'elle est pleine des paquets sont supprimés. Cependant dans la théorie, le flux de données en sortie ne stagne pas durant la congestion. Tous les paquets sont éliminés de la file d'attente une fois que celle-ci est pleine et la vitesse des données envoyées à la destination varie beaucoup.

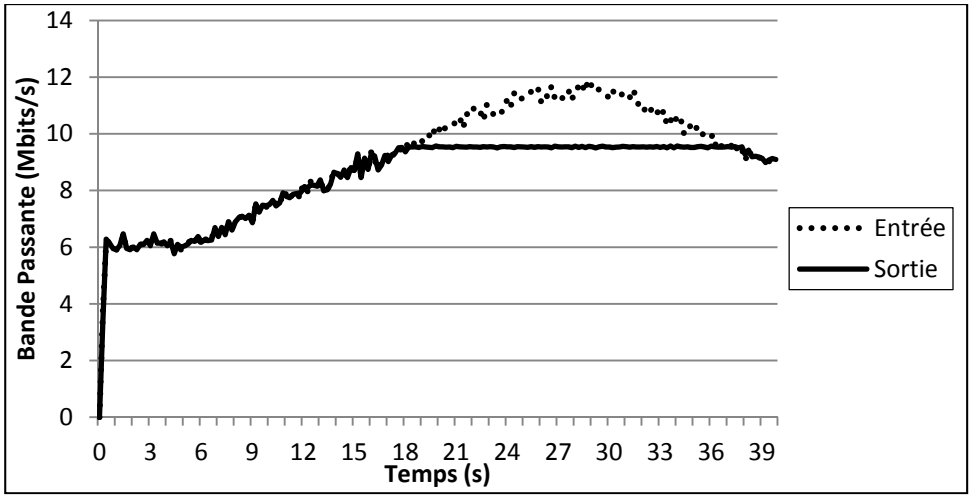

Figure 3.5 Flux d'entrée dans l'interface.

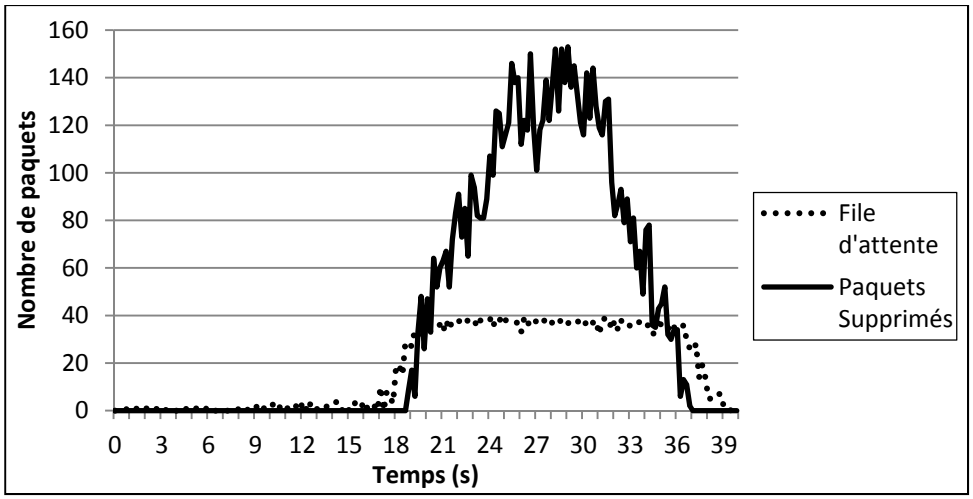

Figure 3.6 Comportement de la file d'attente sur simulateur.

3.3.3 Comparaison des résultats

Dans les parties précédentes, le comportement du réseau lors d'une congestion a été testé et mesuré. Il est maintenant nécessaire de comparer les résultats du banc d'essai et du simulateur pour savoir s'ils sont équivalents ou s'ils divergent. De plus, il faut analyser les données pour voir si les premiers éléments pour la prédiction de la congestion en ressortent.

Dans l'ensemble, les résultats du simulateur s'approchent des résultats du banc d'essai. La suppression des paquets est dans les deux cas proche d'un taux moyen de 13 %. Le moment où commence la congestion par rapport à l'émission du premier client est quasiment identique. En effet dans les deux cas la congestion intervient 15 secondes après le lancement des premiers clients de 100Kbits/s. La seule différence marquée concerne le comportement de la file d'attente. Le simulateur tend à montrer qu'il est envisageable de porter l'attention sur la file d'attente pour prédire une congestion. Cependant, les résultats obtenus sur le banc de test montrent que la file d'attente peut avoir un comportement qui ne permet pas de prévoir la congestion assez en avance. Le développement d'une solution sur simulateur est donc envisageable si celui-ci ne se base pas sur la valeur des files d'attente. La validation du simulateur est donc permise par ces données.

Pour agir lorsqu'une congestion il faudrait selon les articles parus sur la norme PCN (Arumaithurai *et al.*, 2009) environ 600 ms avant que les changements ne prennent effet. Ce temps est très court et pourrait s'avérer plus long selon la technique utilisée. Or, d'après les tests sur le banc d'essai, la file d'attente ne fournit des données exploitables que 400 ms avant la congestion. Cette valeur est un maximum et selon les cas il se peut qu'il n'y ait aucune information avant la première suppression de paquets. Il est donc clair que le nombre de paquets dans la file d'attente ne doit pas être un indicateur clé d'une prédiction de la congestion.

La donnée identique entre le simulateur et le banc de test est le taux d'utilisation de la bande passante. En effet, c'est l'indicateur le plus fort de la congestion. C'est d'ailleurs l'attribut le plus utilisé dans les applications de routage sous contraintes. En plus du taux d'utilisation de la bande passante la donnée critique est la perte de paquets. En effet, celle-ci est un compteur incrémental. Cela signifie qu'à n'importe quel moment il est réellement possible de savoir si des paquets ont été perdus depuis le dernier relevé.

Par conséquent, l'algorithme doit utiliser deux données du trafic : le taux d'utilisation de la bande passante et le nombre de paquets perdus. Les tests sur équipements réels se sont révélés fastidieux, car le relevé des informations par un serveur central est soumis au délai. Sur une architecture plus importante, il serait difficile de contrôler les informations de trafic à distance. Il est donc préférable de développer un algorithme qui pourrait se positionner directement sur les équipements en utilisant les informations de trafic disponible. L'ensemble de ces éléments montre qu'il est possible de développer une solution sur le simulateur qui pourra fonctionner de la même manière sur le banc d'essai. Cet algorithme est présenté dans le chapitre suivant.

3.4 Éléments problématiques des mesures

Les mesures en environnement réel ont montré des points faibles qu'il est nécessaire de connaître. Le développement et la validation d'une solution de prédiction des congestions ne peuvent se faire qu'en pleine connaissance de ces remarques.

3.4.1 Mesures du pourcentage d'utilisation de la bande passante

La donnée utilisée le plus fréquemment pour caractériser un lien est le pourcentage d'utilisation de la bande passante. La mesure de celle-ci pour une réaction rapide est délicate. Le maximum de bande passante que peut offrir un lien est une moyenne estimée sur une seconde. Mais cet intervalle de temps est trop grand et peut contenir une rafale et donc une congestion. Lorsque la mesure est effectuée sur une base de temps très petite elle est imprécise quelque soit l'environnement de travail. En effet les flux mesurés en Mbits/s ne sont pas envoyés en continu. Ce sont des paquets envoyés à un certain temps d'intervalle. Lors des simulations, si des paquets sont envoyés toutes les 20 ms et que la mesure s'effectue toutes les 30 ms alors elle mesurera un débit variable. Par exemple, entre 30 et 60 ms elle ne verra qu'un seul paquet alors qu'entre 60 et 90 ms elle en verrait deux. Cela fait donc osciller le débit mesuré. Plus l'intervalle de mesure est court et plus cette variation peut être importante. Cependant, si le temps de mesure est trop grand, une congestion peut apparaître dans le réseau avant de pouvoir être détectée.

3.4.2 Reproductibilité des expériences

Pour le travail sur simulateur, chaque scénario est reproductible et se produit exactement de la même manière. Les tests sur le banc d'essai ne permettent pas de reproduire les mêmes expériences plusieurs fois avec exactement les mêmes résultats. En effet la variation des délais au sein du réseau en est la première cause. Sur le simulateur, même si le délai est sélectionné en mode aléatoire, la série est toujours rigoureusement la même. De plus,

l'application Iperf nécessite un dialogue entre la source et la destination pour établir un flux de données, même en UDP. Ceci ne permet pas de lancer un trafic à la milliseconde près, ce qui fait varier l'instant auquel la congestion se produit.

3.4.3 Configuration de l'application Iperf

Durant les tests il est apparu que l'application Iperf mesure les données au niveau Applicatif. Cela signifie que les en-têtes IP ne sont pas comptabilisés dans les paramètres d'envoi de paquets. L'en-tête IP prend un minimum de 20 octets. Cette donnée n'est pas importante si les paquets envoyés sont très gros, par exemple 1500 octets. Par contre si les paquets envoyés sont de taille de paquets voix sur IP (en moyenne 200 octets), alors l'erreur est de 10 %. Sur une bande passante de 10 Mbits/s, l'erreur totale peut être de 1 Mbits/s. Sur le simulateur il n'y pas d'en-tête IP qui se rajoute, les données mesurées sont donc identiques à celles paramétrées. Afin de générer des clients avec précisément le même nombre d'octets pour le banc d'essai et le simulateur, il est nécessaire d'effectuer certains calculs avant de faire les tests. La formule générale pour calculer quels nombres doivent être utilisés pour configurer convenablement Iperf est la suivante :

$$BPI = \frac{BPS}{TPS \times 8} \times TPI \times 8 \tag{3.1}$$

Avec BPI : Bande Passante configurée sur Iperf (en bits/s)
BPS : Bande Passante Souhaitée dans le réseau (en bits/s)
TPIS : Taille de Paquet IP Souhaitée dans le réseau (en octets)
TPI : Taille de Paquet configurée sur Iperf (en octets)

3.4.4 Performance des serveurs

Pour générer des clients Iperf de petite taille sur une grande bande passante il est nécessaire d'avoir des serveurs très performants. Les serveurs Dell utilisés sur le banc d'essai ont un processeur de 3 GHz. Cependant dès la génération de plus de 20 clients lancés via des processus séparés, le débit envoyée varie grandement et de manière aléatoire. Les valeurs peuvent varier de plus de 1 Mbits/s en l'espace d'un cycle de mesure. Cet élément est très problématique pour la reproductibilité des mesures.

3.5 Conclusion

L'étude menée dans ce chapitre avait deux buts principaux. Le premier était de valider le comportement du simulateur par rapport à celui du banc d'essai. Cela avait pour but de permettre le développement d'une solution sur le simulateur plutôt qu'en situation réelle. Il est en effet plus facile de développer une nouvelle solution sur un simulateur. Le deuxième but était d'étudier le comportement de la congestion pour mieux la comprendre et connaître la réaction des équipements.

Les résultats ont permis de valider l'utilisation du simulateur. En effet le comportement entre le simulateur et le banc d'essai est très proche. Le ratio des paquets perdus est quasiment identique et les réactions aux congestions sont identiques. L'étude du comportement de la congestion a permis de mettre en avant le comportement aléatoire de la file d'attente. Elle a également montré que pour prévoir une congestion les données les plus fiables étaient le pourcentage de bande passante envoyé ainsi que le nombre de paquets perdus. Ces données serviront donc de base pour travailler sur une prédiction de la congestion.

Plusieurs points de vigilances ont aussi été détectés avec la réalisation de l'étude de la congestion. Notamment la non-reproductibilité des expériences sur le banc d'essai, ce qui rend très difficile le développement d'un algorithme sans simulateur. La mesure du

pourcentage utilisé de la bande passante est également un élément crucial qu'il ne faut pas négliger. Il est donc nécessaire de réfléchir à l'avance à la valeur des cycles de mesure.

CHAPITRE 4

DÉFINITION DE L'ALGORITHME DE PRÉVISION

Ce chapitre contient l'intégralité du comportement de l'algorithme développé pour prédire la congestion. L'algorithme a été réalisé après une analyse approfondie du comportement du réseau. La version présentée dans ce chapitre a été conçue après une série de tests qui a permis d'affiner son comportement. Le chapitre résume tout d'abord les caractéristiques générales de l'algorithme. Puis chaque partie est définie individuellement avant d'être replacée dans un contexte plus général.

4.1 Définition des termes et globalités

4.1.1 But de l'algorithme

L'algorithme développé dans le cadre de cette maîtrise doit respecter les règles suivantes :

- prédire les congestions : L'algorithme ne doit pas simplement alerter le système lorsqu'il y a une congestion. Il doit permettre de savoir si il va y avoir une congestion dans le futur ou non. L'algorithme ne vise que les prévisions de congestion à court terme. L'échelle de temps des prévisions est donc de quelques secondes.
- facilité d'implémentation : Il ne doit pas nécessiter de trop grosses modifications et utiliser tant que possible les informations déjà contenues dans les équipements.
- détecter les rafales : Il doit permettre de différencier les rafales sporadiques des congestions pouvant bloquer le réseau.
- être développé à l'échelle : L'algorithme doit pouvoir être développé à grande échelle. Il ne doit pas se limité à être déployé dans des petits réseaux.
- rôle d'avertisseur : L'algorithme ne décide pas de changer le trafic. Son but est de générer une alarme qui peut servir aux mécanismes d'ingénierie de trafic pour agir.

- modifiable : L'algorithme doit être développé en plusieurs parties pour être facilement modifiable par des recherches futures.

4.1.2 Informations utilisées

L'algorithme se base sur deux informations de trafic différentes pour chaque lien des équipements. Ces informations peuvent être retrouvées sur chaque routeur peu importe son constructeur ou son modèle. Pour plus de simplicité, nous réutilisons ici les termes employés par le simulateur NS-2 pour éviter la confusion au travers des chapitres :

- barrivals_ : le nombre d'octets arrivés dans la file d'attente entre deux mesures,
- pdrops_ : le nombre de paquets rejetés entre deux mesures.

Ces valeurs sont des compteurs incrémentaux. Cela signifie que la valeur augmente sans cesse jusqu'à atteindre le maximum du compteur, puis elle revient à zéro. Les types de stockage de ces valeurs ne changent pas selon l'environnement de travail, ils sont disponibles sur les équipements réels et dans les simulateurs. Si l'algorithme ne se base pas sur le nombre de paquets sur les files d'attente, c'est parce que cette information s'est montrée instable sur le banc d'essai. En effet durant la plupart des études de la congestion la valeur du nombre de paquets dans la file d'attente est restée à zéro tant qu'il n'y avait pas de suppression de paquets. Certains tests ont cependant fait apparaître une valeur non nulle dans le créneau juste avant la congestion (à 200 millisecondes). Ceci n'est pas suffisant pour se servir de cette information à titre d'avertisseur. Il est possible qu'il y ait des paquets dans la file d'attente. Cependant, l'objet SNMP ne représente pas une moyenne, mais une valeur au temps de la mesure. Cela signifie qu'entre les cycles de mesures il peut y avoir des paquets stockés dans la file d'attente que les expériences ne permettent pas de voir. Cette valeur est toujours relevée pendant les tests pour mieux analyser les congestions.

L'algorithme est composé de trois parties, la première mesure les données et les mets en ordre, la deuxième calcule en combien de temps une congestion peut arriver et la troisième établit les alarmes.

4.2 Le module de relevé

Le module de relevé des données reçoit les informations de trafic puis calcule les différents indicateurs. Ce module permet aussi le lissage des données selon plusieurs approches. Ce lissage est nécessaire pour que l'extrapolation soit plus efficace.

4.2.1 La mesure des données

Les données sont rafraîchies toutes les 200 millisecondes pour chaque information de trafic. Ce délai a été choisi car il permet une réaction rapide pour les trafics en temps réel, notamment la voix et la vidéo. Cependant, il reste raisonnable et évite de surcharger les équipements en calcul.

Pour pouvoir calculer le débit moyen durant ces intervalles on utilise l'information barrivals_. Le calcul pour avoir la bande passante en Mbits/s est le suivant :

$$bwarrival_{\_1} = \frac{barrivals\_ \times oct \times \frac{1}{t}}{Mbit} \quad (4.1)$$

oct : le nombre de bits dans un octet

$\frac{1}{t}$: le nombre de cycle dans une seconde, avec t le temps d'un cycle

Mbit : le nombre de bits dans un mégabit soit 1 048 576

bwarrival_1 : la donnée la plus récente calculée pour le débit moyen

Dix mesures du débit moyen sont conservées pour pouvoir effectuer des calculs par la suite. Il est cependant aisé de rajouter des valeurs supplémentaires si cela est nécessaire.

Le nombre de paquets supprimés dans un intervalle de mesure ne nécessite aucun calcul. En effet, ce nombre sert strictement d'indicateur pour savoir où est la congestion. Il est également utilisé pour alerter d'un stade critique si l'algorithme échoue. La valeur pdrops_ est donc utilisée sans aucune modification.

Enfin à chaque cycle de mesure, lorsque cela est possible, les compteurs sont remis à zéro ce qui permet de faciliter les calculs de bande passante. S'il n'est pas possible de remettre les valeurs à zéro, il suffit de conserver les deux dernières valeurs de barrivals_ ainsi que de pdrops_ et de faire la différence.

4.2.2 Lissage des données

La variation aléatoire du délai de transmission sur les liens entraîne une variation du débit mesuré à chaque cycle. Ces variations peuvent être plus ou moins importantes. Elles peuvent être la cause de rafales sporadiques, puisqu'elles génèrent une montée rapide, mais courte du trafic. Pour l'extrapolation, il est essentiel d'avoir une tendance qui se rapproche du trafic réel sans en avoir toutes les petites variations. Pour cela, l'algorithme est proposé avec quatre techniques de lissages semblables, mais aux résultats différents :

- Moyenne sur une seconde : Toutes les 200 ms on calcule la moyenne sur la dernière seconde.
- Moyenne pondérée sur une seconde : Toutes les 200 ms on calcule la moyenne sur la dernière seconde. Cependant, les données les plus récentes ont un poids plus fort dans le calcul.
- Moyenne sur deux secondes : Toutes les 200 ms on calcule la moyenne sur les dernières deux secondes.
- Moyenne pondérée sur deux secondes : Toutes les 200 ms on calcule la moyenne sur les deux dernières secondes. Comme pour la moyenne pondérée à une seconde les données les plus récentes ont une valeur plus forte que les plus anciennes.

Pour le traitement de ce lissage une nouvelle variable a été utilisée. Cette variable est BPValueUsed_. Elle permet de calculer le débit lissé et est utilisée par le module d'extrapolation comme si c'était le débit envoyé réellement dans l'interface. En tout, trois valeurs de BPValueUsed_ sont conservées pour les besoins de l'extrapolation.

La figure 4.1 représente la différence entre deux des techniques de lissage. Elle montre que les moyennes sans pondérations sont moins susceptibles aux variations que les moyennes sans pondérations. Cependant, elles génèrent un retard sur la mesure des données. La bande passante maximum est en effet atteinte plus tardivement. Les solutions avec pondérations sont légèrement plus susceptibles aux fluctuations, mais ont moins de retard sur le niveau de la bande passante utilisé.

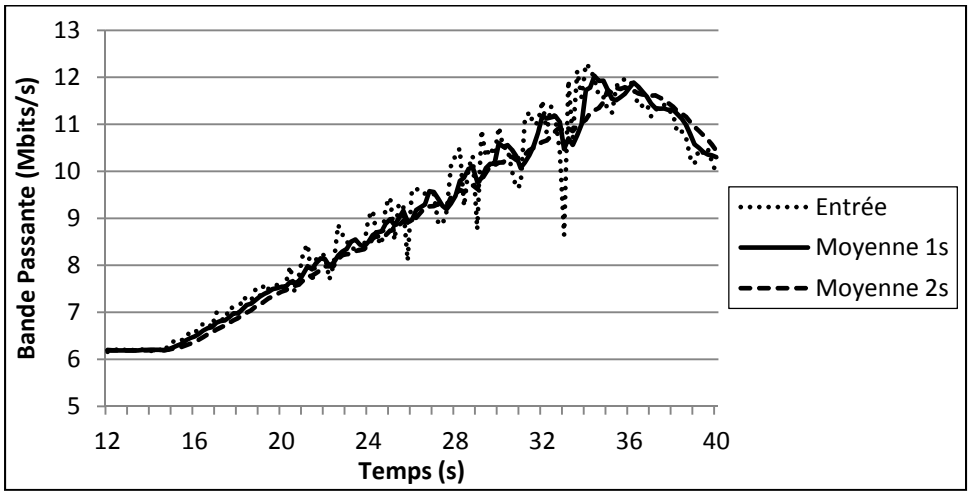

Figure 4.1 Méthodes de lissage des données.

4.3 Module d'extrapolation

Ce module se base directement sur les informations stockées par le module de mesure. Il utilise les dernières valeurs de bande passante pour calculer le temps avant une possible congestion. Le nombre de données utilisées pour le calcul dépend de la méthode utilisée. Dans notre étude nous utilisons quatre méthodes : l'extrapolation linéaire, logarithmique, un mélange des deux techniques ainsi que la méthode polynomiale de Lagrange.

Afin de pouvoir décider s'il peut y avoir congestion ou non il est nécessaire d'extrapoler les points obtenus via la partie mesure. On peut ainsi estimer le temps nécessaire à la congestion. Ce calcul est effectué lors des phases de croissance positive des arrivées. Plusieurs théories peuvent être appliquées sur ce module.

La variable définie dans cette partie de l'algorithme pour caractériser le temps restant avant la congestion est *predictbw_*.

4.3.1 Modèle linéaire

Pour ce modèle, on prend les deux dernières mesures de la bande passante afin de calculer la pente entre les deux points. Avec le coefficient directeur de la pente, on peut ensuite calculer à quel moment la congestion (100 % de la bande passante) peut arriver.
Le calcul global :

$$\alpha = \frac{BP_1 - BP_2}{t} \tag{4.2}$$

avec BP_1 : mesure du débit à l'instant t
BP_2 : mesure du débit à l'instant t – 1
α : coefficient directeur de la pente

$$predictbw\_ = \frac{(BP_{max} - BP_1)}{\alpha} \tag{4.3}$$

avec BP_{max} : bande passante maximum du lien
predictbw_ : le temps (en secondes) restant avant la congestion

Le modèle linéaire se base sur deux points de mesure. Il est donc très sensible aux variations du trafic. Il convient aux évolutions de trafic ayant des formes linéaires prononcées et dont les fluctuations aléatoires ne sont pas nombreuses. Si l'évolution du trafic fluctue beaucoup, le modèle linéaire ne permettra pas d'avoir de bons résultats. De plus, il est très sensible aux petites montées brusques qui pourraient engendrer des résultats faussés même si la bande passante est très basse.

4.3.2 Modèle logarithmique

Pour ce modèle on estime que plutôt qu'une courbe linéaire le trafic sera de tendance logarithmique (sa montée se fait progressivement moins rapide). On cherche donc ici à trouver un logarithme qui passerait par le dernier point de mesure. Il suffit d'effectuer une translation des axes afin de prendre comme point à x=1 la dernière mesure de bande passante et de trouver un logarithme qui passe par ce point. Cela permet d'estimer la suite du trafic à en utilisant une courbe logarithmique. Si la translation avait été choisie à x=0 ou inférieure, cela créerait une extrapolation beaucoup plus pessimiste. Pour trouver quel logarithme passe par le dernier point de mesure, il est nécessaire de trouver la base de celui-ci. En effet pour chaque point aux coordonnées réelles il existe un logarithme de base a tel que :

$$y = loga(x) = ln(x)/ln(a) \tag{4.4}$$

d'où :

$$exp(ln(2)/y) = a \tag{4.5}$$

avec $y = BP1$: dernière valeur de débit calculée

Il faut ensuite calculer à quel moment la congestion peut arriver. Avec la transposition des axes effectuée on cherche

$$x = exp(Yln(a)) \tag{4.6}$$

avec $Y = \mathrm{BP}_{\max}$ soit la bande passante maximum du lien

x : une seconde de plus que la prévision de congestion (en secondes)

Durant la détermination du logarithme, le dernier point de mesure a été placé à x=1 et l'extrapolation donne le temps entre le dernier point de mesure et le point de congestion. Pour calculer le temps restant avant d'atteindre la bande passante maximum on doit donc soustraire un à *x* et cela donne le temps restant avant la congestion.

L'extrapolation logarithmique minimise la montée du trafic. Elle est donc très optimiste pour la détection d'une congestion. Si les données de trafic ne forment pas de logarithme et croissent rapidement, les résultats de cette méthode ne seront pas adéquats. Elle peut cependant suffire à un trafic pour lequel la montée de la bande passante se ferait lentement. L'avantage de cette formule est que les rafales ne perturbent presque pas son fonctionnement.

4.3.3 Modèle polynomial de Lagrange

Pour utiliser le modèle de Lagrange on utilise les trois derniers points de mesures afin d'avoir un polynôme de degrés deux (une parabole). L'équation de base pour trois points avec le modèle de Lagrange est la suivante :

$$P(x) = y_0 \times \frac{x - x_1}{x_0 - x_1} \times \frac{x - x_2}{x_0 - x_2} + y_1 \times \frac{x - x_0}{x_1 - x_0} \times \frac{x - x_2}{x_1 - x_2} + y_2 \times \frac{x - x_0}{x_2 - x_0} \times \frac{x - x_1}{x_2 - x_1} \qquad (4.7)$$

avec x_i , y_i : des couples de points utilisés pour l'extrapolation
x_0 , y_0 : le point le plus ancien
x_2 , y_2 : le point le plus récent

Pour simplifier le calcul et ainsi augmenter la vitesse de l'exécution de l'algorithme, il suffit de faire une transposition des axes. Pour des points d'abscisse (19 ; 19,2 ; 19,4) on transpose à (0 ; 0,2 ; 0,4). On procède également de la même façon pour les ordonnées. Soit en globalités pour

$$x_0, y_0 \rightarrow x_0 - x_0, y_0 - y_0$$
$$x_1, y_1 \rightarrow x_1 - x_0, y_1 - y_0$$
$$x_2, y_2 \rightarrow x_2 - x_0, y_2 - y_0$$

De cette manière, le premier terme est annulé et les x_0 peuvent être enlevés de l'équation. De plus les termes x_1 et x_2 auront toujours la même valeur qui dépends du temps choisit pour les relevés (dans notre étude 200ms). Après transformation l'équation P(x) peut s'écrire :

$$P(x) = y_1 \times \frac{x}{x_1} \times \frac{x - x_2}{x_1 - x_2} + y_2 \times \frac{x}{x_2} \times \frac{x - x_1}{x_2 - x_1} \tag{4.8}$$

Cette équation donne un polynôme de degrés deux soit sous la forme $P(x) = ax^2 + bx + c$. Il faut donc tout d'abord développer l'équation pour séparer les termes en x^2, en x et la constante pour faire une identification. Afin d'arriver à cette séparation les x_1 et x_2 sont remplacés par leur valeur soit dans notre étude 0,2 et 0,4. Il faut ensuite développer la solution 4.8 avec ces valeurs afin de séparer les termes de degrés différents. Avec le développement de la formule 4.8 avec les valeurs x_1 et x_2 mentionnées on obtient :

$$P(x) = \frac{x^2(y_2 - 2 \times y_1) + x(-2 \times (-0{,}4) \times y_1 - y_2 \times 0{,}2)}{0{,}08} \tag{4.9}$$

Dans l'équation 4.9, le 2 est le coefficient qui a servi à obtenir un dénominateur commun antre les deux divisions. Le 0,08 est le dénominateur commun trouvé entre les deux divisions. Dans notre cas cela représente $[x_2 \times (x_2 - x_1)]$ ou $[-2 \times x_1 \times (x_1 - x_2)]$. Tous les chiffres sont liés au temps que dure un cycle de relevé. Si la durée change, il est nécessaire de refaire les calculs. Une fois le développement terminé et les termes regroupés, il suffit de faire une identification pour trouver les valeurs de *a*, *b* et *c*. Il n'y a pas de constante dans notre équation, car tous les termes dépendent de *x* ou de x^2, cela signifie qu'il n'y a pas de *c*.

Après avoir effectué les calculs nécessaires à partir de l'équation 4.8, on obtient :

$$a = \frac{Coef_{a1}y_1 + Coef_{a2}y_2}{D} \quad (4.10)$$

$$b = \frac{Coef_{b1}y_1 + Coef_{b2}y_2}{D} \quad (4.11)$$

avec y_1 et y_2 : les deux derniers relevés de débit moins le débit le plus ancien

D : représente le dénominateur commun trouvé entre les deux divisions (ici 0,08)

$Coef_{a1}$: coefficient de y_1 pour l'équation de a (avec nos valeurs : - 2)

$Coef_{a2}$: coefficient de y_2 pour l'équation de a (avec nos valeurs : 1)

$Coef_{b1}$: coefficient de y_1 pour l'équation de b (avec nos valeurs : 0,8)

$Coef_{b2}$: coefficient de y_2 pour l'équation de b (avec nos valeurs : - 0,2)

Lorsque l'on veut extrapoler les données, il s'agit de trouver les solutions à l'équation : $BP_{max} = ax^2 + bx + c$. On cherche donc le déterminant Δ pour savoir quelles sont les solutions possibles. On gardera uniquement les solutions positives. En effet si elles sont négatives cela signifie que la solution pour une congestion est dans le passée. Pour connaître le temps restant avant la congestion il suffit de soustraire $2 \times t$ (avec t étant l'intervalle des relevés) à la solution.

L'utilisation des trois derniers points de mesure permet une meilleure extrapolation des données. Plus le nombre de points utilisés est élevé et plus la courbe d'extrapolation donnera des données proches. Cependant avec des points supplémentaires l'équation devient également beaucoup plus complexe à calculer. Le degré de l'équation augmente d'un pour chaque point supplémentaire. Avec un point supplémentaire, il est nécessaire de résoudre une équation du troisième degré en utilisant des nombres complexes. Or il n'existe pas de formules préétablies pour résoudre ces équations. Le calcul avec trois points est le meilleur compromis entre rapidité et précision, ce qui correspond à nos critères.

4.3.4 Illustration des modèles d'extrapolation

La figure 4.2 présente un exemple d'utilisation des techniques d'extrapolation. Les points marqués par des losanges représentent les points de mesures utilisés pour tracer les différentes courbes d'extrapolation. Le trait continu représente l'entrée, c'est-à-dire le trafic en entrée de l'interface (en Mbits/s). On peut voir sur cette figure que le modèle logarithmique est plus optimiste que les autres et le modèle polynomial est le plus pessimiste.

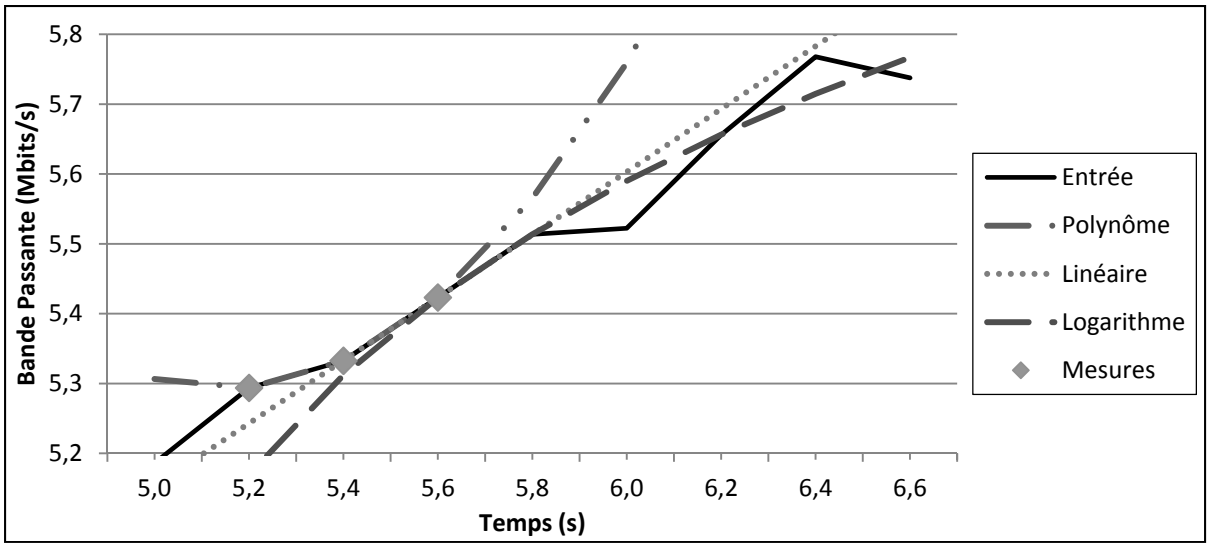

Figure 4.2 Illustration des modèles d'extrapolation.

4.3.5 Choix du modèle d'extrapolation

L'ensemble des tests qui ont été menés pour développer cet algorithme ont montré que l'extrapolation polynomiale de Lagrange est le modèle le plus polyvalent et le plus efficace. Celui-ci réagit mieux que les autres lorsque les données varient aléatoirement. Or le trafic Internet a une forme aléatoire sur des échelons à court terme. Même si à long terme, il a une tendance facilement identifiable. C'est l'extrapolation la plus pessimiste, mais elle a montré les meilleurs résultats globaux dans toutes les conditions d'essais. Ce choix sera justifié dans les résultats présentés dans le chapitre six.

4.4 Module décisionnel

Après avoir calculé le temps avant la congestion, et selon le niveau de criticité du trafic l'interpréteur a pour rôle de prévenir d'une éventuelle congestion ou d'une rafale. Cette interprétation est basée sur une gestion à plusieurs niveaux qui sera expliquée dans les parties suivantes.

4.4.1 Définition des niveaux

Le module décisionnel se base sur une échelle à 5 niveaux de criticité. Plus le niveau est élevé et plus la congestion est proche. Les paliers de ces niveaux (A_1, A_2, A_3) sont calculés dynamiquement en fonction de la bande passante des paquets arrivants dans l'interface. Le niveau d'alerte est donné en fonction de la valeur de *predictbw_* soit la valeur calculée par le module d'extrapolation. Les niveaux définis dans ce module sont les suivants :

- Niveau 0 : Temps de congestion prévu *predictbw_* > A_1
- Niveau 1 : Temps de congestion prévu à A_2 < *predictbw_* <= A_1
- Niveau 2 : Temps de congestion prévu à A_3 < *predictbw_* <= A_2
- Niveau 3 : Temps de congestion prévu *predictbw_* <= A_3 ou BP > 0,8 x BP_{max}
- Niveau 4 : Bande passante dépassée et paquets perdus

Les paliers sont calculés par la formule 4.11 où *BandNorm* représente le pourcentage de bande passante utilisé par rapport à la bande passante maximum.

$$A_3 = \frac{e^{\beta \times BandNorm}}{\alpha} + \delta = {}^{A_2}/_{2} = {}^{A_1}/_{3} \tag{4.12}$$

A_3 : niveau d'alerte 3

BandNorm : pourcentage de la bande passante utilisé

α : paramètre de décalage du point de montée

β : paramètre permettant de varier l'intensité de la courbe exponentielle

δ : constante permettant de définir un paramètre supplémentaire

Les paramètres permettant de faire varier la courbe exponentielle pour les niveaux permettent de régler deux : l'intensité et le point de monté de l'exponentielle. La pente peut-être plus ou moins raide grâce au réglage d'intensité. Le point auquel la courbe va monter est régler grâce au terme α. Dans notre étude α et β sont respectivement à 2980 et 8. Cela permet d'avoir une courbe logarithmique variant entre 0 et 1 pour des valeurs *BandNorm* entre 0 et 1. De plus le point de montée de la courbe se situe vers 0,5, ce qui permet d'éviter les erreurs de détection lorsque le débit est sous les 50% de la bande passante. De plus le paramètre δ est une constante permettant de définir un minimum pour le palier de niveau 3. Dans notre étude nous avons paramétré δ à 0,2 soit 200ms.

Ces niveaux avec paliers dynamiques permettent d'être plus stricte lorsque la bande passante utilisée est faible. En effet lorsque celle-ci est faible il est peu probable qu'une congestion se manifeste. Pour éviter qu'une petite variation ne déclenche l'alerte de congestion, le niveau d'alerte critique est plus bas. En revanche lorsque la bande passante utilisée est importante (supérieure à 80 %) le niveau sera progressivement plus élevé et l'alerte sera donnée plus tôt. Le niveau A_3, qui est le niveau le plus critique, varie de 0,2 seconde pour 0 % de la bande passante maximum, à 1,2 seconde pour 100 % de la bande passante (*Voir* Figure 4.3).

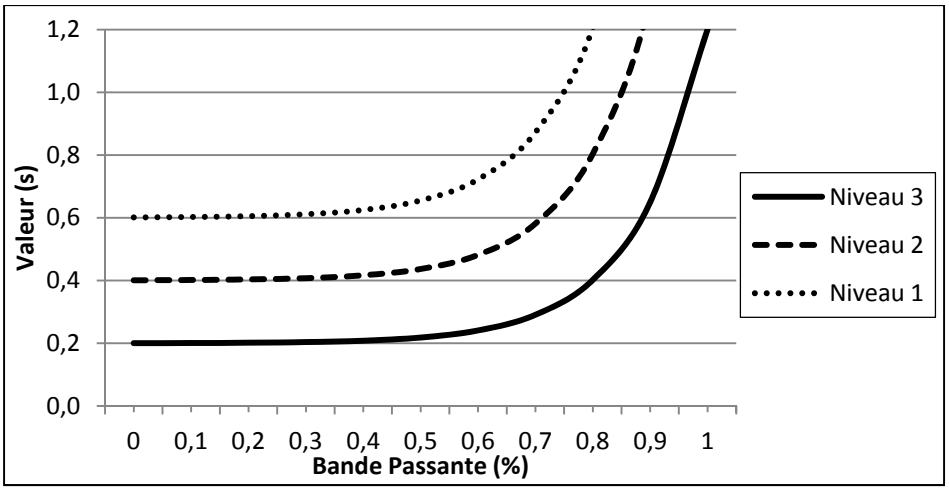

Figure 4.3 Variation des niveaux d'alertes.

Les 3 derniers changements de niveaux sont conservés afin de différencier les rafales sporadiques d'une montée normale du trafic. Le temps auquel ces changements d'état ont été faits est également gardé en mémoire. Pour stocker ces valeurs, six variables sont utilisées :

- alert_1, alert_2, alert_3 : les trois derniers niveaux d'alertes avec alert_1 étant la plus récente ;
- time_alert_1, time_alert_2, time_alert_3 : les trois derniers temps auxquels les changements d'état se sont produits.

4.4.2 Comportement du module

Le module a plusieurs réactions possibles selon le niveau auquel il se situe ainsi que le comportement du trafic. On différencie trois types de comportements.

4.4.2.1 Trafic croissant

Lorsque le trafic augmente et que la congestion se rapproche il n'est possible de monter que d'un niveau à la fois, même si les prédictions nous plaçaient dans un niveau supérieur. Si le trafic monte très rapidement il faut donc plusieurs cycles de mesures pour arriver à un niveau

critique. Une alerte est envoyée lorsque deux conditions sont remplies. Il faut tout d'abord que le niveau d'alerte soit à trois avec une prédiction de congestion inférieure au seuil A_3. Ensuite, les derniers changements d'état doivent avoir été effectués à une moyenne supérieure à 0,3 seconde. Cela signifie que l'état n'a pas changé à chaque cycle de mesure. Cette dernière condition permet d'éviter une réaction trop brusque lors d'une rafale sporadique. En effet celle-ci pourrait faire passer rapidement le niveau d'alerte à 3 alors qu'aucune congestion ne va réellement se produire. Le calcul est effectué de la manière suivante :

$$moyenne = \frac{now\_ - time_{alert_3}}{\mu} \tag{4.13}$$

now_ : le temps au moment du calcul

$time_{alert_3}$: le temps de changement de niveau le plus ancien en mémoire

μ : le nombre de cycle pour la moyenne (ici 3)

Ce calcul permet de définir la moyenne passée dans les trois derniers changements d'état. Elle prend en compte le temps au moment de la mesure plutôt que le dernier réel changement d'état. Ceci est fait pour éviter qu'une rafale sporadique ne puisse congestionner le réseau si elle dure longtemps et si elle n'est pas identifiée comme telle.

La seule exception à ces règles est lorsque le trafic est supérieur à 80 % de la bande passante. Dans ce cas il n'y a plus de contrôle sur les rafales car celles-ci peuvent facilement congestionner le réseau et engendrer une perte de paquets. L'alerte est alors envoyée lorsque le temps donné par le module d'extrapolation est inférieur à A_3. Ceci permet d'agir plus rapidement lorsque le taux d'utilisation du lien devient important.

4.4.2.2 Détection des rafales

Une rafale est caractérisée par une montée très rapide du trafic, donc un changement de prévision de la congestion très rapide. Les rafales peuvent être facilement détectées par le système. En phase de croissance le changement d'état ne peut se faire qu'un niveau à la fois. Lorsqu'une rafale se manifeste, si elle ne dure qu'un cycle de mesure ou moins elle ne perturbera donc pas le fonctionnement de la détection puisque que seul un niveau au maximum sera franchit. Le seul cas problématique ou une rafale sporadique peut provoquer une alerte est si la détection est à un niveau d'alerte deux et qu'une rafale suffisamment importante se produit durant un seul cycle de mesure. Dans ce cas une alerte à la congestion peut être faussement envoyée.

Si la rafale dure plus d'un ou deux cycles de mesure la modification des niveaux se fera de manière très rapide. La moyenne des changements d'état sera donc inférieure aux 300 ms nécessaires au déclenchement d'alerte. La rafale ne déclenchera donc pas d'alerte. Cependant si elle dure plus de quatre cycles et devient un risque pour le réseau, la moyenne des temps de changements augmentera et une alerte sera donnée.

La détection des rafales est mise en place pour éviter qu'une montée brusque du trafic, mais non menaçante, ne déclenche une alerte. Par exemple si de nouveaux flux s'ajoutent en simultanés sur un lien, portant la capacité utilisée de 30 à 60 %, sans détection des rafales une alerte serait émise faussement.

4.4.2.1 Trafic Stagnant ou décroissant

Dans le réseau le trafic n'est pas uniquement croissant ou décroissant. Il y a plusieurs phases successives de plus ou moins courtes durées. Certains tests ont montré que la congestion pouvait arriver rapidement après une courte phase de décroissance. De ce fait, lors des phases de décroissance, l'algorithme ne fait plus d'extrapolation et le dernier état d'alerte est

conservé. Si le trafic décroît rapidement après une forte hausse puis remonte légèrement, le niveau d'alerte peut redescendre à une criticité faible. Ce niveau sera établi selon les valeurs d'extrapolation. Par contre si le trafic est à un niveau d'alerte deux et devient décroissant puis passe à un niveau d'alerte trois, une alerte de congestion est émise. Pour les cas où le trafic stagne, le traitement des alertes est identique au traitement en cas de trafic décroissant.

4.5 Résumé de l'algorithme

La figure 4.4 représente en globalité les interactions de l'algorithme. La première action est effectuée par le module de relevés. Il importe les données contenue sur les équipements puis il calcule les informations de débit, la valeur lissée ainsi que le nombre de paquets perdus. Ensuite le module d'extrapolation utilise les données issues du lissage pour prévoir le temps restant avant la congestion. Puis, le module décisionnel utilise les informations de paquets perdus, de bande passante et d'extrapolation pour définir le niveau d'alerte courant. Enfin, il se base sur l'historique des mesures pour décider ou non d'émettre une alerte de congestion.
Plusieurs variables sont utilisées dans cet algorithme. Dans le module de relevé, les tableaux *bwarrival_[10] et BPValueUsed_[3]* représentent respectivement les 10 dernières valeurs de bande passante calculées par le module de mesure et les trois dernières valeurs calculées par le lissage. La variable *pdrops_* équivaut au nombre de paquets perdus dans le dernier cycle de mesure. Dans le module d'extrapolation, la valeur *predictbw_* correspond à la dernière prédiction faite pour le temps de congestion. Dans l'interpréteur, *alert_[3]* correspond aux trois derniers changements de niveaux d'alertes constatés. Le tableau *time_alert_[3]* stocke les temps de chaque changement de niveau d'alerte.

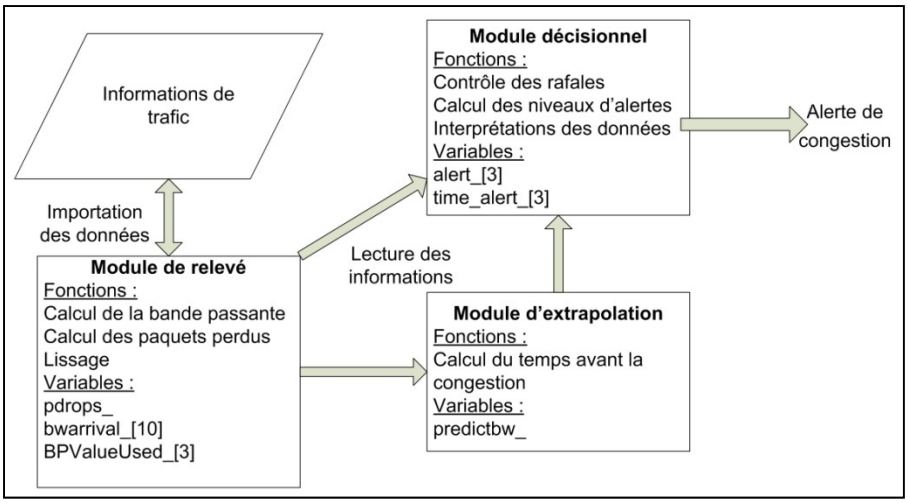

Figure 4.4 Interactions de l'algorithme.

Le tableau 4.1 résume le comportement de l'algorithme pour chaque niveau d'alerte. La première et la deuxième colonne définissent respectivement les niveaux d'alerte et les valeurs de prédiction pour lesquels ils sont valides. La dernière colonne résume le comportement que l'algorithme peut avoir à chaque niveau d'alerte selon la valeur de prédiction ou le comportement du trafic. La valeur BP représente la bande passante arrivant en entrée et BP_{max} représente la valeur de bande passante offerte par le lien.

Tableau 4.1 Changements d'état d'alerte

Niveau	predictbw_	Comportement
0	$>A_1$	- predictbw_ < A_1, passage au niveau 1 - trafic décroissant ou stagnant, l'état est conservé
1	$<A_1$ ET $>A_2$	- predictbw_ < A_2, passage au niveau 2 - predictbw_ > A_1, passage au niveau 0 - trafic décroissant ou stagnant, l'état est conservé
2	$<A_2$ ET $>A_3$	- predictbw_ < A_3, passage au niveau 3 - predictbw_ > A_2, passage au niveau 1 - predictbw_ > A_1, passage au niveau 0 - trafic décroissant ou stagnant, l'état est conservé
3	$<A_3$ OU BP > 80%	- predictbw_ < A_3 si moyenne > 300ms (ou BP <80%) → émission d'une alerte de congestion - predictbw_ > A_3, passage au niveau 2 - predictbw_ > A_2, passage au niveau 1 - predictbw_ > A_1, passage au niveau 0 - trafic décroissant ou stagnant, l'état est conservé
4	$BP \geq BP_{max}$ ET pdrops_ > 0	- alerte de congestion

CHAPITRE 5

IMPLÉMENTATION DE L'ALGORITHME

Dans le chapitre précédent, le fonctionnement de l'algorithme a été défini. Celui-ci n'a été présenté que de manière théorique. Pour pouvoir mesurer l'efficacité de cette nouvelle approche, il est nécessaire de l'implémenter sur une architecture de test. Le chapitre cinq présente les procédures d'installation de l'algorithme sur les deux infrastructures de test. Il présente tout d'abord la manière de laquelle l'implémentation est réalisée sur le banc d'essai. La deuxième partie explique l'implémentation de l'algorithme sur le simulateur.

5.1 Tests sur le banc d'essai

5.1.1 Implémentation de l'algorithme

Travaillant sur des routeurs propriétaires il n'est pas possible de rajouter notre algorithme au sein de ceux-ci pour pouvoir le tester. Le banc de test contient des routeurs Linux qu'il est possible de modifier; cependant ils ne sont pas compatibles avec le MPLS des routeurs Cisco. Le MPLS sous Linux est encore en développement et les fonctionnalités développées ne permettent que l'établissement de trajets MPLS entre routeurs Linux. Pour pouvoir développer l'algorithme sur des équipements réseau, il serait donc nécessaire de disposer d'assez de serveurs pour créer un réseau MPLS au complet. Pour pallier à ce problème, l'algorithme est testé de manière théorique grâce aux informations de trafic recueillies. Ces informations sont stockées sur une base de données grâce à l'application Lagrit Poller. Il est ensuite aisé de les transférer sur un fichier Excel. Les effets de l'algorithme sont donc calculés de façon théorique dans un tableur. Les calculs sont effectués via des formules et l'interprétation des résultats est effectuée manuellement. Le comportement de l'algorithme étant complètement développé par l'utilisateur, une utilisation en implémentation ou de manière théorique ne diffèrera pas grandement. La seule différence entre un développement sur les équipements ou une requête via l'application Lagrit Poller, réside dans la précision

des mesures. Lors d'un développement sur équipements les réactions peuvent être générées en temps réel et les mesures sont effectuées à des intervalles précis. Or, avec une mesure par interrogation SNMP le délai des requêtes varie, ce qui fait varier l'écart entre chaque requête. La bande passante calculée peut ainsi osciller en fonction de la différence de délai.

5.1.2 Cheminement de l'analyse

Étant donné que les données récupérées par SNMP sont uniquement des valeurs d'informations de trafic, l'analyse n'est pas possible directement. Il est nécessaire de les mettre en forme et d'effectuer certains calculs avant de les analyser. Pour effectuer la totalité de l'analyse, il est nécessaire de passer par plusieurs étapes :

- Copie des informations de trafic : pour chaque type d'information de trafic (chaque objet SNMP) il faut copier la totalité des relevés de l'expérience dans un feuillet Excel.
- Calculs des données nécessaires à l'algorithme : après avoir transféré toutes les données brutes, il est nécessaire de calculer les données essentielles pour l'algorithme. Par exemple, la bande passante moyenne sur chaque cycle, calculée grâce aux valeurs des compteurs d'octets.
- Lissage des données : pour chaque type de lissage, un feuillet différent rassemblant l'ensemble des données nécessaire à l'analyse a été créé. Ces données sont :
  (a) le temps écoulé de la simulation,
  (b) la bande passante envoyée à l'interface congestionnée,
  (c) cette bande passante lissée,
  (d) la bande passante en sortie de cette interface,
  (e) la taille de la file d'attente,
  (f) le nombre de paquets supprimés par intervalle.
- Formules d'extrapolation : il est ensuite nécessaire de calculer les valeurs données par les différents modules d'extrapolation et d'y intégrer le calcul du niveau d'alerte Ta3 pour chaque cycle de mesure.

- Analyse des données : après les phases de calculs, il faut analyser les données pour connaître le moment exact de la congestion. Si cela n'est pas dû à une rafale sporadique, la congestion est le moment où le premier paquet est supprimé. La phase d'analyse suivante consiste à trouver à quel moment l'algorithme se déclenche. Pour cela il faut suivre l'évolution des niveaux d'alertes. Les résultats sont résumés dans un tableau contenant plusieurs informations sur le test. Par exemple le temps au moment du déclenchement de l'alerte, les paquets perdus, et la bande passante atteinte au déclenchement. Il est ensuite possible de procéder à la comparaison des différentes méthodes.

Lorsque le premier fichier Excel est créé, il est très facile de reproduire les formules pour toutes les expériences. Cela permet de ne pas perdre de temps lorsqu'il faut faire des tests avec des paramètres différents (taille de paquets, temps du cycle de mesures).

5.2 Test sur simulateur

5.2.1 Implémentation de l'algorithme

L'implémentation de l'algorithme sur NS-2 a été réalisée directement sur les fichiers TCL (*Voir* Annexe VI) du scénario. Cela permet un déploiement plus rapide pour tester le bien-fondé de l'algorithme développé dans le chapitre trois. Le but de cette simulation n'est pas de tester une quelconque réaction, mais de déterminer à quel moment l'algorithme déclenche l'alerte. Ainsi, les résultats des tests sont enregistrés dans un fichier rassemblant toutes les informations. Il est donc possible d'analyser l'efficacité de l'algorithme après les simulations. Pour plus de facilité dans la conception des scénarios et limiter le nombre de lignes dans un même fichier, l'algorithme a été développé dans un fichier indépendant qui est appelé dans chaque scénario de test. Il permet de comparer les quatre méthodes d'extrapolation qui sont : linéaires, logarithmique, mixte (moyenne des estimations linéaires et logarithmiques) et polynomiale. Il est également possible d'utiliser les différentes méthodes de lissage des données présentées dans le chapitre 3 décrivant l'algorithme.

5.2.2 Implémentation des modules d'extrapolation

Chaque module d'extrapolation a été développé de façon identique. Les deux dernières valeurs de prédictions sont gardées en mémoire pour pouvoir avoir une trace de l'évolution. La prévision est mise à jour selon le mode d'extrapolation choisi pour passer ensuite au module décisionnel. Les variables utilisées pour la partie extrapolation sont les suivantes :

- predictbw_1 : le temps (en secondes) après lequel on prévoit la congestion
- predictbw_2 : l'ancienne valeur de prévision en mémoire

La Figure 5.1 illustre le code utilisé pour l'extrapolation logarithmique. Pour simplifier la programmation en TCL il est nécessaire de passer par plusieurs étapes de calculs puis de les regrouper.

```
set predictbw1_2 $predictbw1_1
if {$bwarrivals_1 > 1.0} {
        set predictbw1_1 9999
} else {
        if {[expr $bwarrivals_1-$bwarrivals_2]>0} {
                set b 1.000
                set A [expr exp([expr log(2)/[expr $bwarrivals_1 - $bwarrivals_2]])]
                set Y [expr 1 - $bwarrivals_2]
                set X [expr exp([expr $Y * log($A)])]
                set predictbw1_1_1 [expr [expr $X-2] * $time]
                set predictbw1_1_2 [expr        [expr $b-$bwarrivals_1] / [expr [expr $bwarrivals_1-
$bwarrivals_2] / $time]]
                set predictbw1_1 [expr [expr $predictbw1_1_1 + $predictbw1_1_2]/2]
        } elseif {[expr $bwarrivals_1-$bwarrivals_2]==0} {
                set predictbw1_1 99
        } else {
                set predictbw1_1 -99
        }
}
```

Figure 5.1 Code d'extrapolation logarithmique.

5.2.3 Module décisionnel

Le module décisionnel est appelé une fois que le module d'extrapolation a fini de faire ses calculs. Cette partie du code contient en majeure partie des boucles et des conditions relatives

à chaque état par lequel l'algorithme doit pouvoir passer. Les variables déclarées pour cette partie sont :

- rafale_ : la variable rafale_ si elle est à 0 indique qu'il n'y a pas de rafale en cours. Si elle passe à 1 cela indique une rafale et le traitement effectué par le module décisionnel est modifié en conséquence.
- alerte_1, time_alerte_1 : ces variables donnent le dernier niveau d'alerte (celui en cours) ainsi que le moment depuis lequel ce niveau a été atteint.
- alerte_2, time_alerte_2, alerte_3, time_alerte_3 : ces variables servent à garder en mémoire les deux derniers changements d'état avant celui en cours pour caractériser le trafic.

Le traitement pour la partie interprétation est identique à celui mentionné dans le chapitre 4. Lorsque des alertes sont émises, elles sont stockées dans des fichiers spécifiques qui contiennent leur nature, mais aussi le temps auquel elles ont été émises. Cela permet par la suite de pouvoir analyser le comportement de l'algorithme. Il n'est donc pas nécessaire dans ce cas d'analyser les changements de niveau comme pour les tests sur le banc d'essai. Les expériences sur le simulateur sont de ce fait plus rapides et il est plus aisé de faire plusieurs modifications puisque les résultats sont automatiquement calculés et ordonnés dans un fichier Excel. Les seules analyses à effectuer concernent le calcul des données utilisées pour comparer les différentes méthodes (par exemple la perte de paquets et la bande passante atteinte).

5.3 Définition des données

Pour pouvoir comparer les différents paramètres de notre méthode entre eux ou les confronter à d'autres solutions, il est nécessaire de définir des données représentatives des expériences. Selon les expériences à effectuer, ces données sont différentes.

La première donnée à utiliser pour ces expériences est l'instant auquel l'alerte se déclenche. En effet notre solution étant un avertisseur il est intéressant de relever à quel moment celui-ci donne l'alerte en fonction de la méthode utilisée. L'efficacité des méthodes dépend du temps

de réaction pour un mécanisme d'ingénierie de trafic. Dans cette étude le temps de réaction estimé pour qu'un mécanisme d'ingénierie de trafic agisse est d'une seconde. La deuxième donnée importante lors des tests est la perte de paquets. Les paquets perdus sont additionnés jusqu'à exactement une seconde après l'alerte lancée par l'algorithme. Si le déclenchement s'effectue plus tardivement qu'une autre solution, cela permettra de savoir s'il y a une perte de paquet induite par ce délai ou non. La troisième donnée pouvant être utilisée est la bande passante atteinte au moment de l'alerte. Les autres mécanismes se basent sur des limites fixes soit dans le trafic soit sur la file d'attente, il est donc pertinent de comparer la bande passante atteinte au moment de l'alerte. Enfin, la dernière donnée est la différence du nombre d'octets transmis. En effet il est intéressant de noter les différences de données qui ont été émises. Cela permettra de voir si notre solution permet une meilleure utilisation des liens.

Ces données doivent être couplées les unes aux autres. Car savoir que l'algorithme a réussi à atteindre une bande passante importante sans savoir s'il y a eu des pertes n'est pas significatif. Les données doivent donc être analysées convenablement pour pouvoir mener à une explication complète des résultats.

CHAPITRE 6

SCÉNARIOS DE TESTS ET RÉSULTATS

Ce chapitre présente les tests nécessaires à la validation de l'algorithme conçu au travers du chapitre quatre. Il est en effet essentiel, pour justifier la conception de cet algorithme, de le tester dans diverses conditions et de le comparer à d'autres solutions. La première partie du chapitre présente les scénarios de chaque test effectué. Ces essais sont faits sur le banc d'essai ainsi que sur le simulateur. La deuxième partie présente les résultats de ces expériences ainsi que leur analyse.

6.1 Scénarios de tests

6.1.1 Scénario de congestion sur le banc d'essai

D'après les données présentées dans le mémoire (Owezarski, 2006), le trafic Internet évolue selon une forme croissante le matin, puis décroissante en soirée et quasi nulle durant un petit créneau de nuit. Les graphiques de ce mémoire montrent plusieurs rafales durant la journée. Pour le troisième trimestre 2009, Cisco a publié les résultats d'une étude du trafic Internet (Cisco, 2010). Les données sont collectées à travers 20 fournisseurs pendant tout le trimestre. Cisco a fourni une figure représentant le trafic moyen journalier. La forme générale du trafic correspond aux données publiées dans le mémoire cité plus haut.

Afin de simuler au mieux les réactions de l'algorithme dans un réseau réel, le comportement du trafic est inspiré de ces deux références. Le trafic est généré depuis les serveurs P01 et P09 vers le serveur P10. Le serveur P09 envoie du trafic à un taux de 5 Mbits/s avec des paquets de 1200 octets. Ce flux est utilisé comme trafic de données dans le réseau. Pour générer du trafic voix depuis P01, des clients de 80Kbits/s sont envoyés. Chaque client représente l'équivalent d'une communication voix sur IP avec le codec G.711. Les paquets mesurent 200 octets et sont envoyés toutes les 20ms. Chaque client a une durée de 20

secondes et ceux-ci débutent l'un après l'autre avec un espacement de 200 ms. En tout, 70 clients sont envoyés.

Pour déployer ce scénario sur le banc d'essai, il est nécessaire de configurer Iperf. Afin de générer des clients de 80 kbit/s il est nécessaire d'effectuer quelques ajustements. En effet la configuration d'Iperf ne prend pas en compte les en-têtes IP. De ce fait si la configuration est de 200 octets les paquets IP auront 220 octets et la bande passante ne sera pas de 80 Kbits/s mais de 90 Kbits/s. Sur un client cela ne représente pas beaucoup, mais sur l'ensemble des 70 clients la perturbation est d'environ 1 Mbits/s. Pour avoir un client de 80 Kbits/s dans le réseau, Iperf doit être configuré avec une bande passante de 70 Kbits/s et des paquets de 180 octets. Pour un client de 5 Mbits/s avec des paquets de 1200 octets, la bande passante envoyée doit être de 4.91 Mbits/s avec des paquets de 1180 octets.

6.1.1 Scénario de test sur le simulateur

Le scénario de test pour comparer les méthodes de l'algorithme est le même que sur le banc de test. Le trafic constant est de 5 Mbits/s avec des paquets de 1 200 octets. Après 5 secondes, des clients envoyant des paquets de 200 octets toutes les 20 ms sont ajoutés toutes les 200 ms pour atteindre la congestion. Chaque client a une durée de transmission de 20 secondes et le scénario dure 40 secondes. La fonction aléatoire incluse dans NS-2 permet de faire osciller le délai autour de la valeur définie sur les liens. De ce fait, la variation des délais se rapproche du comportement réel du trafic. Même si les valeurs sont aléatoires la série de chiffres reste la même ce qui permet d'avoir les mêmes caractéristiques de trafic entre chaque simulation. La version TCL du scénario utilisé dans NS-2 figure en détail dans l'annexe V.

6.1.2 Comparaison avec des méthodes différentes

Afin de voir la pertinence de cette nouvelle approche, l'algorithme est comparé à deux méthodes existantes. La première est Random Early Detection (RED) et la deuxième est « Pre-Congestion Notification » (PCN), une approche nouvelle dans MPLS. Les tests sur banc d'essai étant difficilement reproductibles, RED n'a été comparé que sur simulateur. La comparaison avec la solution PCN sera effectuée via une étude théorique. Elle sera donc faite pour le banc de test ainsi que le simulateur.

6.1.2.1 Random Early Detection

RED permet de supprimer des paquets pour éviter de remplir les files d'attente. Les aspects fondamentaux de cette méthode résident dans l'établissement de deux paliers concernant le taux de remplissage des files d'attente. Le système RED peut aussi marquer les paquets pour signaler une congestion. Dans cette dernière configuration, son comportement peut-être vu comme un avertisseur de congestion.

Pour comparer notre approche avec RED, le scénario est le même que lors des tests précédents. Les algorithmes testés sont ceux pour lesquels nos résultats ont été les meilleurs. La comparaison est donc effectuée entre RED et l'algorithme avec lissage pondéré à deux secondes et extrapolation logarithmique. Dans le simulateur, RED est activé par un changement de paramètre. La méthode de suppression de paquets est choisie car elle permet de savoir à quel moment RED est susceptible de se déclencher et de marquer des paquets.

6.1.2.2 Pre-Congestion Notification

Les systèmes PCN permettent d'alerter le réseau d'une possible congestion. Ces alertes sont émises après que le trafic ait dépassé un niveau fixé par les administrateurs de réseaux. Cette solution n'est disponible ni sur le banc de test ni sur le simulateur NS-2. Cependant, il est

possible d'étudier son comportement théorique. En effet dans l'article mentionné dans le chapitre deux (Arumaithurai *et al.*, 2009), les niveaux de marquage des paquets utilisés lors de l'évaluation du système sont donnés. De plus, les temps de réaction sont calculés. Pour comparer les systèmes PCN à nos déclenchements d'alerte, l'approche théorique est donc appliquée. Pour comparer les résultats du simulateur, les méthodes sont les mêmes que celles utilisées pour comparer notre solution à RED. Dans cette partie les résultats du banc de test sont également comparés à PCN. Pour cela les solutions comparées seront l'extrapolation linéaire et logarithmique avec un lissage sur une moyenne d'une seconde et l'extrapolation polynomiale avec un lissage avec une moyenne deux secondes.

Le tableau 6.1 présente les caractéristiques utilisées pour la comparaison des deux algorithmes. Les caractéristiques des paliers ont été extraites de (Arumaithurai *et al.*, 2009). Le temps de réaction pour le contrôle d'admission est de zéro seconde. En effet puisqu'un message de demande de flux est marqué dès que le trafic dépasse le palier, celui-ci est directement refusé. Il n'y a donc pas de délai puisqu'il n'y a pas de flux à couper. Le temps de réaction pour la terminaison de flux est également extrait de (Arumaithurai *et al.*, 2009). Ce temps a été mesuré par différents tests sur simulateur. Les fonctionnalités AC et FT seront testées de manière séparée. Le test doit montrer la différence de ces deux comportements dans le déclenchement d'alerte. Il sera donc possible de savoir l'impact d'un déclenchement à 60 ou 80 % de la bande passante maximale.

Tableau 6.1 Caractéristiques retenues pour PCN

Admission Control	Limite	60%
	Temps de réaction	0 seconde
Flow Termination	Limite	80%
	Temps de réaction	0,6 seconde

6.2 Résultats

6.2.1 Expérience sur le banc d'essai

La figure 6.1 montre le trafic en entrée de l'interface qui cause la congestion ainsi que ce qui devrait être la l'évolution théorique du débit générée par le scénario. Elle présente beaucoup de variations aléatoires dans le trafic généré réellement. Les variations de ce débit sont dues à trois éléments. Tout d'abord, le délai de transmission est aléatoire. Le cycle de mesure étant de 200 ms, la variation du délai peut influencer la valeur des mesures de bande passante. Ensuite, les clients Iperf ne réussissent pas toujours à se connecter cela créer donc un décalage pour le maximum de bande passante demandée. La non-stabilité du débit des clients générés par Iperf ajoute également de l'amplitude aux variations du trafic.

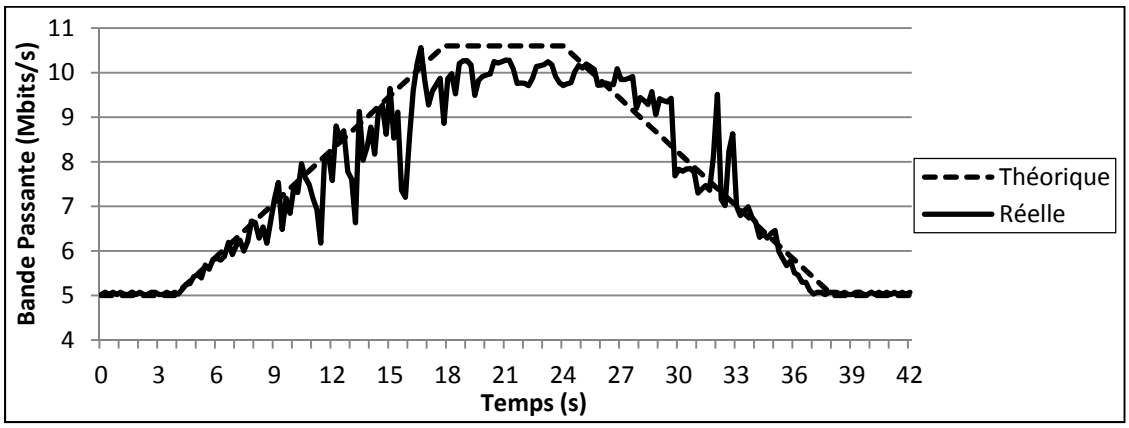

Figure 6.1 Trafic envoyé sur le banc de test réel.

Le tableau 6.2 résume les résultats de l'expérience sur le banc d'essai. Il présente les résultats pour les quatre méthodes d'extrapolation utilisées avec les quatre méthodes de lissage des données ainsi que sans lissage. Chaque méthode est comparée via quatre caractéristiques :

- temps alerte : le temps auquel l'alerte de congestion est émise par l'algorithme,
- prévision de congestion : le temps prévu par l'extrapolation pour la congestion,
- paquets perdus : la somme des paquets perdus jusqu'à une seconde après l'alerte,

- bande passante atteinte : la bande passante en arrivée au moment de la congestion, en pourcentage par rapport à la bande passante du lien.

La congestion commence à 16,4 secondes et en tout 1668 paquets sont supprimés. La comparaison entre le flux de données théorique et réel montre une nette différence quand au maximum de bande passante atteint. Ceci est dû au fait que les clients émis par Iperf ne fonctionnent pas toujours lors du lancement du script. En effet certains clients ne parviennent pas à se synchroniser avec le récepteur et échouent au moment de leur lancement.

En analysant le tableau, les résultats montrent que le lissage engendre un retard dans la détection. Pour la solution linéaire par exemple il passe de 11,6 secondes sans lissage à une détection ratée avec lissage de deux secondes. Le retard peut être assez important pour que le déclenchement de l'alerte ne se fasse qu'après détection de la perte de paquets. Les extrapolations linéaire, logarithmique et mixte sont les plus perturbées par ces méthodes de lissage. Elles sont en effet toutes les trois déclenchées par la perte de paquets dans le lissage avec une moyenne de deux secondes sans pondération. C'est ce qui explique que la bande passante maximum atteinte au moment de l'alerte est de 100,56 %. En effet la bande passante est celle en arrivée et lorsque celle-ci dépasse les 100 % du lien, la congestion apparaît et des paquets sont supprimés.

Les données montrent que l'extrapolation linéaire et logarithmique ont presque le même comportement global. Cependant, selon la croissance moyenne du trafic les deux modèles peuvent avoir un comportement semblable comme ici ou totalement différent. Si le trafic croît très rapidement et de manière plus linéaire l'extrapolation logarithmique devrait déclencher une alerte plus tardivement. Si le trafic fluctue beaucoup et de manière brutale la prévision linéaire émettra également une alerte plus tôt.

Pour l'extrapolation polynomiale, l'alerte est donnée quelques secondes avant la congestion. Entre le mode sans lissage et le mode avec lissage la détection est effectuée de 4,8 à 3

secondes en avance. De manière générale ces résultats montrent que la solution polynomiale est celle qui a le plus de succès malgré une prédiction donnée plus tôt. Elle permet néanmoins d'éviter la perte de paquets ce qui est crucial pour la qualité de service. Même si l'alerte est déclenchée tôt sur le plan temporel, les résultats montrent que la bande passante minimale pour l'alerte avec le mécanisme polynomiale est de 80,6 %. Avec 3 secondes d'avance sur la congestion, la bande passante est arrivée jusqu'à 91,3 %. Ces ratios d'utilisation montrent une bonne utilisation du débit du lien.

Tableau 6.2 Résultats du test sur le banc d'essai

Congestion (s)	16,4				
Lissage \ Extrapolation		Linéaire	Logarithmique	Mixte	Polynômial
Trafic Normal	Temps Alerte	11,6	11,6	11,6	**11,6**
	Prévision	11,92	11,74	11,83	**11,76**
	Paquets perdus	0	0	0	**0**
	BP (%)	80,6	80,6	80,6	**80,6**
Moyenne 1s Pondérée	Temps Alerte	15	15	15	**12,2**
	Prévision	15,61	15,55	15,58	**12,65**
	Paquets perdus	0	0	0	**0**
	BP (%)	96,4	96,4	96,4	**88**
Moyenne 1s	Temps Alerte	16,4	16,6	16,6	**12,2**
	Prévision	17,72	16,65	16,60	**12,78**
	Paquets perdus	114	147	147	**0**
	BP (%)	100,19	100,56	100,56	**88**
Moyenne 2s	Temps Alerte	16,6	16,6	16,6	**13,4**
	Prévision	18,24	19,86	19,05	**13,94**
	Paquets perdus	147	147	147	**0**
	BP (%)	100,56	100,56	100,56	**91,3**
Moyenne 2s Pondérée	Temps Alerte	16,4	16,4	16,4	**13,4**
	Prévision	17,26	17,38	17,32	**13,88**
	Paquets perdus	114	114	114	**0**
	BP (%)	100,19	100,19	100,19	**91,3**

Les résultats du tableau portent sur une seule expérience. Pour la comparer à une autre solution, il faut obligatoirement relancer une nouvelle expérience. Ceci est nécessaire car il faut modifier les paramètres sur les équipements. Dans les configurations du banc de test le même script de génération de trafic ne redonnera pas exactement les mêmes résultats. En effet les délais varient de manière aléatoire différente entre chaque expérience. De plus,

certains clients Iperf peuvent échouer sur une expérience et réussir à envoyer des données dans une autre. Cela change la bande passante utilisée et il est donc difficile de comparer le comportement entre deux algorithmes. Dans les tests sur simulateur, les paquets envoyés ainsi que les délais sont toujours rigoureusement les mêmes. Utiliser un simulateur est donc la solution la plus appropriée pour analyser les différences avec d'autres approches.

6.2.2 Expérience sur le simulateur

La figure 6.2 représente le trafic en entrée de la file d'attente de l'interface servant à la congestion ainsi que le trafic théorique. On peut voir sur cette figure que même si la fonction aléatoire a été activée, le trafic se rapproche beaucoup plus du scénario théorique que le trafic sur le banc d'essai.

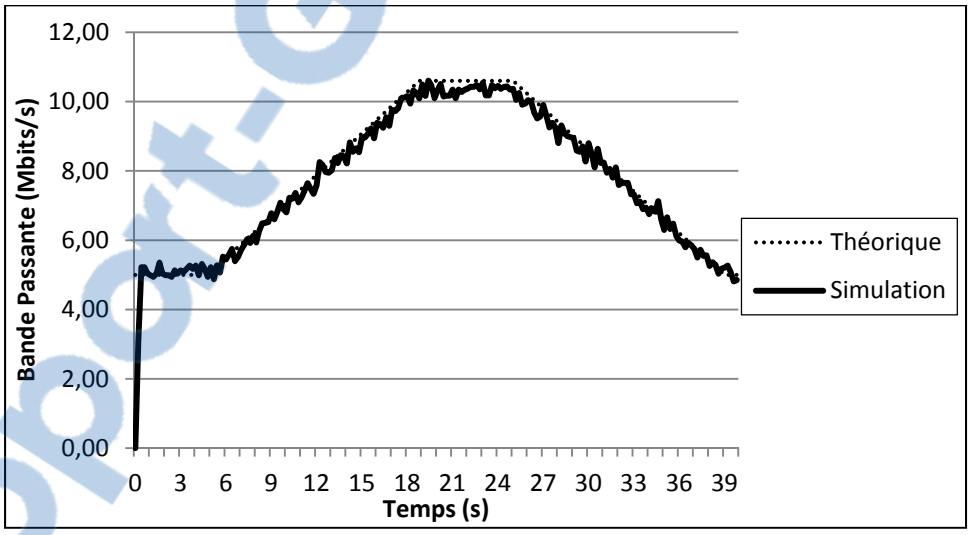

Figure 6.2 Scénario de test sur simulateur.

Le tableau 6.3 résume les résultats obtenus lors de la simulation. La congestion se produit dans l'intervalle de 17 à 17,2 secondes. Durant la durée de la congestion, 3112 paquets sont supprimés. La moyenne des paquets mesurés dans la file d'attente est de 35 paquets. Le maximum de bande passante en sortie atteint au moment de la congestion est de 9,57 Mbits/s.

Lors de cette expérience, la première donnée obtenue indique que sans lissage, les déclenchements d'alertes arrivent très rapidement pour la solution linéaire et la solution polynomiale. Cependant, les solutions logarithmique et mixte donne une estimation du temps de congestion relativement proche. La différence de prévision avec la réalité est d'environ une demi-seconde puisque la congestion se produit à 17 secondes et que les algorithmes prévoient une congestion à 16,65 et 16,46 secondes.

Avec le lissage, les alertes se déclenchent plus tardivement. Le décalage est trop grand pour les solutions linéaire, logarithmique et mixte puisque le déclenchement est effectué au moment de la congestion. Le débit en arrivée atteint notamment la valeur de 100,9 % de la bande passante offerte par le lien. Cela signifie qu'il y a congestion au moment ou l'alerte est déclenchée. Ceci est dû au fait que les valeurs du lissage prennent du retard sur les valeurs réelles. Ce retard est encore plus visible dans le lissage avec une moyenne de deux secondes où ces trois solutions voient leurs alertes déclenchées par la perte de paquets. Pour le lissage avec une moyenne des flux sur 2 secondes, la bande passante calculée arrive à 10 Mbits/s 1,2 seconde après que le trafic ait atteint cette valeur. Le retard est donc important.

Les résultats pour l'extrapolation polynomiale montrent que celle-ci déclenche des alertes dans les temps même avec un lissage important des données. Ceci est dû au fait qu'elle est très sensible aux variations de trafic. Le modèle d'extrapolation polynomiale est la solution la plus pessimiste. Elle permet cependant d'après ces expériences d'alerter le réseau d'une congestion avec plus d'avance et moins de risque de dépasser le point de congestion que les autres configurations.

Tableau 6.3 Résultats des tests sur simulateur

Congestion (s)	17				
Lissage \ Extrapolation		Linéaire	Logarithmique	Mixte	Polynômiale
Trafic Normal	Temps Alerte	14,2	16	16	**12,2**
	Prévision	14,59	16,65	16,46	**12,36**
	Paquets perdus	0	0	0	**0**
	BP max (%)	88,2	93,8	93,8	**82,6**
Moyenne 1s	Temps Alerte	17,4	17,6	17,6	**15**
	Prévision	18,06	19,29	18,68	**15,48**
	Paquets perdus	305	367	367	**0**
	BP max (%)	98,1	100,09	100,09	**82,6**
Moyenne 1s Pondérée	Temps Alerte	17	17,6	17,6	**14,2**
	Prévision	17,77	17,99	17,89	**14,65**
	Paquets perdus	207	367	367	**0**
	BP max (%)	97,6	100,09	100,09	**88,2**
Moyenne 2s	Temps Alerte	17,6	17,6	17,6	**16**
	Prévision	18,71	36,22	27,47	**16,41**
	Paquets perdus	367	367	367	**0**
	BP max (%)	100,09	100,09	100,09	**93,8**
Moyenne 2s Pondérée	Temps Alerte	17,6	17,6	17,6	**16**
	Prévision	18,15	19,91	19,03	**16,43**
	Paquets perdus	367	367	367	**0**
	BP max (%)	100,09	100,09	100,09	**93,8**

6.2.3 Comparaison des résultats entre banc d'essai et simulation

Les résultats obtenus pour les scénarios du banc d'essai et du simulateur montrent une nette différence dans le nombre de paquets perdus. Cela est dû à l'instabilité des flux sur le banc d'essai. En effet, ceux-ci ne maintiennent pas la congestion en tout temps. Certains cycles de relevés inclus dans la congestion ne montrent aucune suppression de paquets. Alors que dans le simulateur des paquets sont perdus à chaque instant de la congestion, ce qui finit par créer une différence de presque 100 %.

Les réactions de l'algorithme sont cependant du même ordre. Les déclenchements linéaire, logarithmique et mixte arrivent trop tard lorsque le lissage est employé. L'alerte émise par l'extrapolation polynomiale ne cause aucune perte de paquets car elle est toujours émise à au moins une seconde de la congestion. La seule différence pour l'extrapolation polynomiale est

que dans le banc d'essai avec toutes les variations présentes, l'alerte est émise plus tôt que sur le simulateur où les fluctuations sont moins importantes. De toutes les méthodes, la solution polynomiale se révèle donc être la plus intéressante à développer.

6.2.4 Comparaison avec RED

Le tableau 6.4 présente les résultats obtenus lors de cette comparaison. La congestion se produit à 17 secondes de manière identique au scénario précédent. Plusieurs caractéristiques sont comparées. Pour RED le temps d'alerte représente le temps auquel RED entre en action c'est-à-dire lorsqu'il supprime des paquets pour alléger la file d'attente. Les paquets perdus sont ici calculés pour l'ensemble de l'expérience en estimant qu'une seconde après le déclenchement d'une alerte, le trafic soit redirigé pour les solutions utilisant notre algorithme. Les octets transmis représentent la différence du nombre d'octets transmis entre le mécanisme déclenchant l'alerte le plus tardivement et les autres. Ces octets sont calculés entre les temps d'alertes. Enfin, BP_{max} représente la bande passante nécessaire au trafic d'arrivée au moment de l'alerte.

Tableau 6.4 Comparaison de RED avec notre solution

Congestion (s)	17				
Extrapolation	Linéaire	Logarithmique		Polynômiale	
Lissage	Sans Lissage	Sans Lissage	RED	Sans Lissage	Moyenne 2s
Temps Alerte (s)	14,2	16	17,2	12,2	16
Paquets perdus	0	0	2325	0	0
Octets Transmis (Mo)	2,05	4,05	5,7	0	4,05
BP max (%)	88,2	93,8	91,9	82,6	93,8

On peut voir que le mécanisme RED se déclenche tardivement. En effet il se déclenche seulement au moment de la congestion. Ceci est dû au fait que dans ce scénario de simulation la bande passante n'atteint le premier niveau de suppression de paquets qu'au moment de la congestion. En utilisant RED pour détecter une congestion, des paquets seraient perdus. Il

faut cependant remarquer que RED permet une diminution importante de la perte de paquets. En effet, les paquets perdus durant toute la congestion ne sont que de 2325 contre 3112 sans mécanisme de contrôle. Grâce à RED il y a eu une diminution des pertes de 26%.

6.2.5 Pre-Congestion Notification

Les tableaux 6.5 et 6.6 présentent les résultats obtenus lors de la comparaison avec PCN. Les données utilisées pour effectuer la comparaison sont identiques à celles utilisées pour la partie précédente.

Le premier élément ressortant de ces résultats est qu'un contrôle d'admission à 60 % bloque la montée du trafic très rapidement. En effet, l'alerte est émise à environ 9 secondes de la congestion dans les deux cas testés. Par rapport à nos algorithmes, la détection utilise la bande passante de manière peu efficace même si la congestion est évitée.

La fonction de terminaison des flux, même poussée à un niveau élevé de 80 % de bande passante utilisée, coupe le trafic à environ 5 secondes avant la congestion dans les deux études. Cette réaction est proche de celle générée par nos algorithmes sans lissage. Elle reste cependant très en avance sur les résultats où le lissage a été employé. La différence des octets transmis ainsi que la bande passante à laquelle l'alerte se déclenche montrent que notre solution permet d'utiliser la capacité des liens de manière plus optimale.

Ces tests révèlent que les solutions PCN permettent d'éviter les congestions. Cependant pour y arriver le trafic est bloqué à un niveau très bas.

Tableau 6.5 Comparaison de PCN avec les résultats du banc d'essai

Congestion (s)	16,4				
Extrapolation	Linéaire	Logarithmique	PCN		Polynômiale
Lissage	Moyenne 1s	Moyenne 1s	Contrôle d'admission	Terminaison de flux	Moyenne 2s
Temps Alerte (s)	15	15	7,6	11,6	13,4
Paquets perdus	0	0	0	0	0
Octets Transmis (Mo)	7,23	7,23	0	3,66	5,48
BP max (%)	96,4	96,4	62	80,9	91,3

Tableau 6.6 Comparaison de PCN avec les résultats du simulateur

Congestion (s)	17					
Extrapolation	Linéaire	Logarithmique	PCN		Polynômiale	
Lissage	Sans Lissage	Sans lissage	Contrôle d'admission	Terminaison de flux	Sans Lissage	Moyenne 2s
Temps Alerte	14,2	16	8,2	12,4	12,2	16
Paquets perdus	0	0	0	0	0	0
Octets Transmis (Mo)	8,2	7,78	0	3,76	3,55	7,78
BP max (%)	88,2	93,8	63,6	80,4	82,6	93,8

6.3 Discussion

Les solutions existantes génèrent toutes une réaction après l'atteinte d'un niveau fixe. Généralement ce niveau est défini de manière à éviter la congestion. Ces niveaux n'étant pas intelligents, c'est-à-dire qu'ils ne prennent pas en compte le comportement des flux, le trafic est souvent coupé très en amont. Les résultats des expériences réalisées dans les parties précédentes montrent que l'utilisation de la bande passante n'est pas optimale pour les solutions PCN. Ils montrent que le gain possible de débit peut-être de près de 10 % avant que l'alerte ne se déclenche. De plus l'utilisation de RED qui se base sur le nombre de paquets dans la file d'attente montre une réaction tardive. Malgré une limitation du nombre de paquets perdus elle ne permet pas de garantir la qualité de service.

Les résultats de l'ensemble des expériences menées dans ce chapitre valident le fait que la solution polynomiale est la solution la plus sécuritaire pour effectuer l'extrapolation des

données. Elle permet de prévenir les congestions en amont et d'éviter les pertes de paquets. De plus avec des réactions semblables entre le simulateur et le banc d'essai, la validation de l'outil de simulation NS-2 pour développer notre algorithme est une nouvelle fois prouvée.

La méthode développée dans ce mémoire permet une définition dynamique des niveaux basée sur l'évolution du trafic. En effet les niveaux varient en fonction de la variation du pourcentage de bande passante utilisée. Cela permet une réaction plus rapide lorsque le débit demandé s'approche de la capacité maximale du lien. Les tests ont montré qu'à travers les paramètres utilisés l'algorithme peut être utilisé avec différents niveaux de sévérité. En effet, selon le temps de réaction nécessaire à un agissement de l'ingénierie de trafic, il est possible de choisir une méthode dans un ordre de temps approprié. Cet ajustement est effectué en choisissant le mode de lissage. Les résultats donnent des temps de réaction possibles entre 1 à 4 secondes. Il est donc possible de choisir la méthode d'alerte qui convient le mieux aux mécanismes de réaction choisis.

CONCLUSION

Ce mémoire a présenté les problèmes que posent les congestions dans les réseaux informatiques actuels. Aujourd'hui, les services de nouvelle génération telle que la voix sur IP nécessitent d'avoir une très bonne qualité de service. Les congestions sont les évènements les plus néfastes à l'encontre de ce paramètre. Or les mécanismes d'ingénierie de trafic généralement utilisés pour lutter contre les congestions présentent certaines lacunes. Ils se basent sur des niveaux d'alertes fixes et se contentent d'éviter la congestion en limitant l'utilisation des liens. Certains mécanismes permettent d'intervenir une fois que la congestion s'est manifestée, mais cela entraîne une dégradation de la qualité de service. L'utilisation des notifications de pré-congestion qui a été présentée est une solution innovante dans les réseaux MPLS. Cependant, son approche concernant les niveaux d'alertes est restée sensiblement la même que pour les mécanismes traditionnels d'ingénierie de trafic.

Les études menées dans ce mémoire ont montré l'impact des congestions sur des équipements réels ainsi que sur le simulateur NS-2. Elles ont permis de valider le comportement global du simulateur lorsqu'une congestion est simulée. Elles ont également mis en évidence des éléments utiles à la détection de la congestion. Les informations de la file d'attente semblaient être les données idéales pour cette tâche. Mais des problèmes de stabilité de ces informations dans les équipements réels testés ont montré leurs limites. Le choix a donc été fait d'utiliser les informations de bande passante et de paquets supprimés qui peuvent être analysés plus facilement et avec moins d'erreurs.

Une nouvelle approche a été mise en place pour améliorer la détection des congestions dans les réseaux MPLS. Cette méthode a été modélisée via un algorithme de prédiction utilisant des formules d'extrapolation de données. Les techniques d'extrapolation utilisées sont des calculs mathématiques qui n'ont, à notre connaissance, jamais été utilisés dans la détection des congestions sur MPLS. L'algorithme a été conçu de façon modulaire. Une séparation a été effectuée en plusieurs parties permettant ainsi des modifications des caractéristiques plus

rapides. Il a été testé sur le simulateur de réseau NS-2 ainsi que de manière théorique en se basant sur les résultats obtenus avec le banc d'essai. Par le biais de ces expériences, l'algorithme a également été comparé aux solutions PCN et au mécanisme de traitement des files d'attente RED.

L'extrapolation polynomiale s'est montrée la plus à même de prévenir les congestions. Malgré une détection parfois très en amont de la congestion, elle permet de gagner de la bande passante utile sans causer de suppression de paquets. Notre algorithme permet donc une détection plus optimale que les mécanismes se basant sur des limites de déclenchements fixes. L'algorithme permet d'utiliser les liens de manière plus importante sans pour autant engendrer une dégradation de la qualité de service. Le temps auquel l'alerte se déclenche peut également être réglé en fonction des besoins des administrateurs réseau. Il est pour cela nécessaire de choisir le comportement des modules de lissage parmi ceux élaborés dans cette maîtrise. Ce lissage permet de détecter la congestion à un intervalle très court si cela est nécessaire. Mais il est également possible de choisir un lissage permettant une détection avec plus d'avance pour laisser plus de temps à un mécanisme d'ingénierie de trafic.

L'ensemble du travail effectué dans ce mémoire a permis de mettre en évidence la possibilité d'effectuer un contrôle de congestions différent des mécanismes existants à ce jour. Les résultats ont en effet montré que la solution développée donne des résultats positifs pour la détection de la congestion. Il est donc possible d'utiliser des mécanismes de prévisions simples pouvant être déployés sur des équipements réels et donnants des résultats satisfaisants.

RECOMMANDATIONS

Le travail effectué dans ce mémoire a présenté une étude théorique de l'algorithme sur les équipements réels. Sur le simulateur, seul le déclenchement d'une alerte a été testé. Dans le cadre de cette recherche, aucun mécanisme d'ingénierie de trafic n'a été couplé à l'algorithme de détection. Il reste donc des axes à développer pour compléter cette recherche.

La première étape serait d'intégrer ce mécanisme de détection à un mécanisme d'ingénierie de trafic sur simulateur. Cela permettrait de le tester sur plusieurs équipements de manière simultanée avec plusieurs apparitions de congestions. Les résultats pourraient dire si cet algorithme peut être développé sur un réseau plus grand. Le plus simple serait de le coupler sur un mécanisme de redirection du trafic.

Il serait ensuite intéressant de développer l'algorithme de manière interne dans des routeurs MPLS. Cela permettrait de pouvoir améliorer son développement et parfaire ses réactions. Il serait également possible de voir l'impact des calculs sur le comportement du routeur. L'étape la plus aisée serait d'utiliser des routeurs Linux avec fonctionnalités MPLS. Il est cependant nécessaire de disposer d'un grand nombre de serveurs pour effectuer l'ensemble des tests nécessaires.

Dans le but d'intégrer l'algorithme à d'autres solutions, il serait plausible de l'intégrer dans une solution PCN. En effet, disposant de plusieurs niveaux d'alertes il pourrait servir de niveaux pour le contrôle d'admission et la terminaison des flux.

Le dernier axe à voir serait d'effectuer un ensemble de tests en se basant sur des informations de trafic réel passant sur Internet. Ces informations permettraient de voir les réactions de l'algorithme sur du trafic à plus longue durée. Ceci ne peut être fait qu'avec une implémentation totale dans un système d'ingénierie de trafic, sur simulateur ou équipements réels.

ANNEXE I

EXEMPLE DE CONFIGURATION DES ROUTEURS CISCO

```
version 12.4
!
hostname 2821_5
!
mpls traffic-eng tunnels
mpls traffic-eng reoptimize timers frequency
15
mpls traffic-eng path-selection metric igp
!
interface Tunnel24
 no ip address
!
interface Loopback0
 ip address 100.10.5.5 255.255.255.255
!
interface GigabitEthernet0/0
 description lien vers 2821_2
 ip address 172.16.123.2 255.255.255.252
 duplex auto
 speed 10
 mpls traffic-eng tunnels
 mpls traffic-eng administrative-weight 10
 ip rsvp bandwidth
!
interface GigabitEthernet0/1
 description lien vers 2821_3
 ip address 172.16.135.2 255.255.255.252
 duplex auto
 speed 10
 mpls traffic-eng tunnels
 mpls traffic-eng administrative-weight 10
 ip rsvp bandwidth
!
interface FastEthernet0/1/0
 description lien de management
 switchport access vlan 5
 no mop enabled
!
interface GigabitEthernet0/3/0
 description lien vers 2821_4
 ip address 172.16.145.2 255.255.255.252
 negotiation auto
 mpls traffic-eng tunnels
 ip rsvp bandwidth
!
interface Vlan5
 ip address 192.168.1.15 255.255.255.0
!
router ospf 5
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
 log-adjacency-changes
 network 172.16.123.2 0.0.0.0 area 0
 network 172.16.135.2 0.0.0.0 area 0
 network 172.16.145.2 0.0.0.0 area 0
!
snmp-server community synchro RW
snmp-server host 192.168.1.104 version 2c
synchro
```

ANNEXE II

CONFIGURATION D'UN SERVEUR AVEC QUAGGA

Pour la configuration des équipements utilisant Quagga, les informations d'adresses IP sont celles définies sur le serveur. Les routes pour le routage statique sont également celles définies sur le serveur. C'est ce qui rend les fichiers de configurations presque vides. Dans la configuration d'un serveur en tant que routeur, il est également nécessaire de configurer les serveurs pour qu'ils acceptent de router des paquets ne leur étant pas destinés. De façon native les serveurs rejettent les paquets ne leur étant pas destinés.

Les fichiers de configurations sont séparés en fonctions des modules utilisés par Quagga. Ici en l'occurrence figurent des extraits des fichiers pour le routage simple et le routage OSPF. Comme sur la plateforme

Fichier zebra.conf

```
hostname zebraP02
password zebra
!
interface eth0
 ipv6 nd suppress-ra
!
interface eth1
 ipv6 nd suppress-ra
line vty
!
```

Fichier ospfd.conf

```
hostname ospfdP02
password zebra
log stdout
!
interface eth0
interface eth1
interface eth2
interface eth3
interface lo
!
router ospf
 network 172.16.212.2/30 area 0.0.0.0
 network 192.168.11.1/24 area 0.0.0.11
 network 192.168.44.1/24 area 0.0.0.44
!
line vty
!
```

ANNEXE III

APPLICATION LAGRIT POLLER

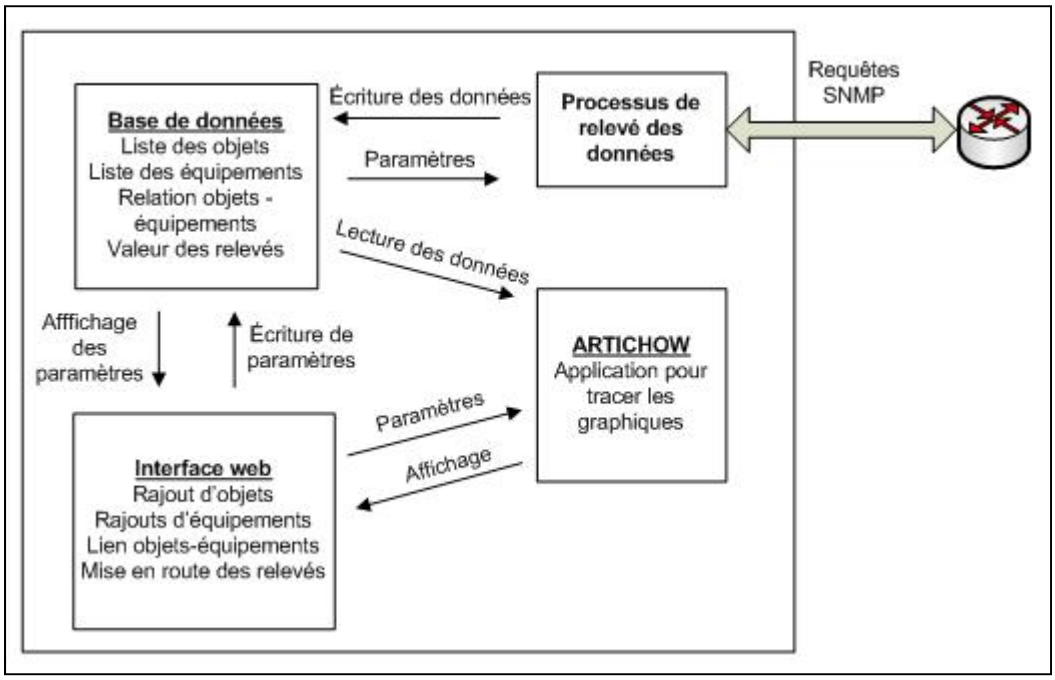

Figure-A III-1 Schéma récapitulatif de l'application Lagrit Poller.

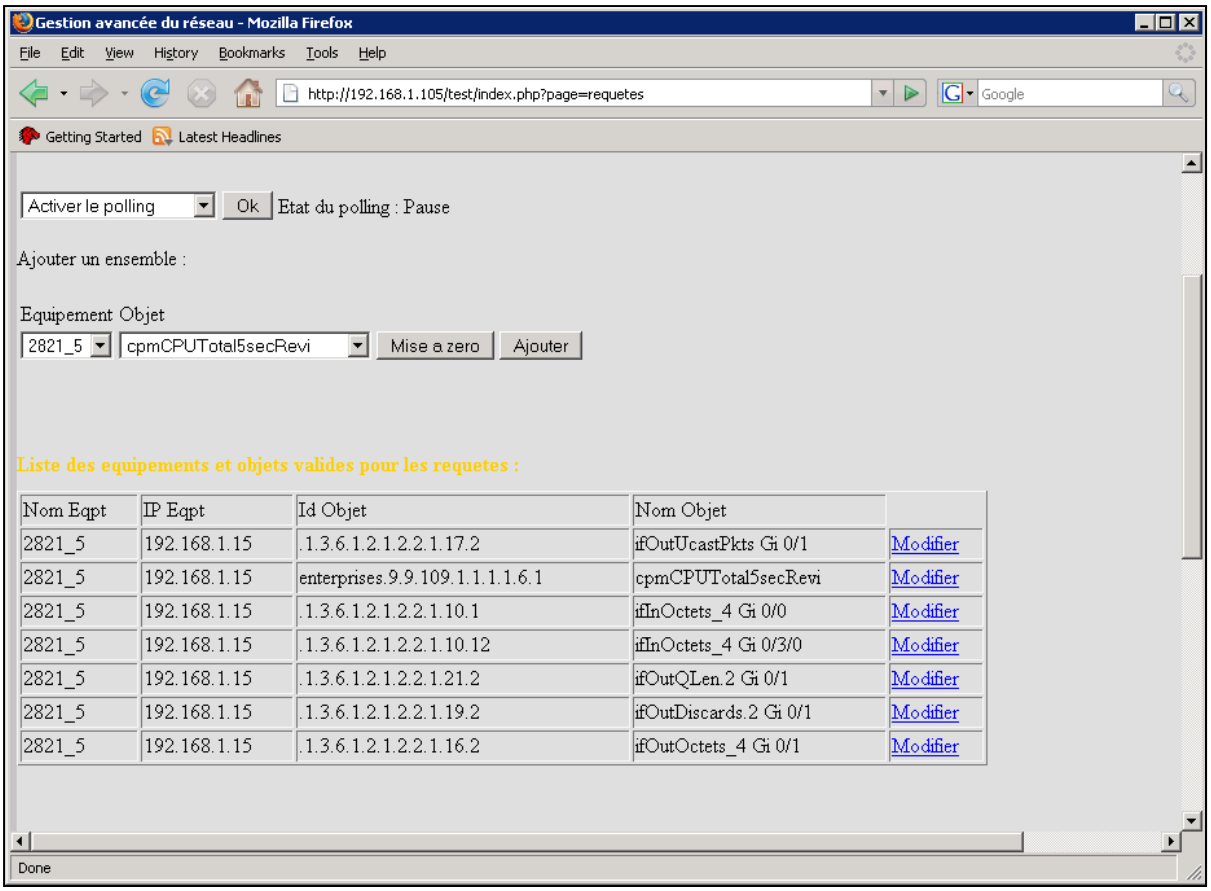

Figure-A III-2 Impression écran des listes de requêtes.

ANNEXE IV

CONFIGURATION DE L'ARCHITECTURE SUR NS-2

```
$ns rtproto DV

#Set the nodes of the network
set n0 [$ns node]
set n1 [$ns node]
$ns node-config -MPLS ON
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
$ns node-config -MPLS OFF
set n7 [$ns node]
set n8 [$ns node]

#Set the links in the network
$ns duplex-link $n0 $n2 10Mb  10ms DropTail
$ns duplex-link $n1 $n3 10Mb  10ms DropTail
$ns duplex-link $n2 $n3 10Mb  10ms DropTail
$ns duplex-link $n2 $n4 10Mb  10ms DropTail
$ns duplex-link $n3 $n5 10Mb  10ms DropTail
$ns duplex-link $n4 $n5 10Mb  10ms DropTail
$ns duplex-link $n4 $n6 10Mb  10ms DropTail
$ns duplex-link $n5 $n6 10Mb  10ms DropTail
$ns duplex-link $n6 $n7 10Mb  10ms DropTail
$ns duplex-link $n6 $n8 10Mb  10ms DropTail

$ns queue-limit $n5 $n6 40

# configure ldp agents on all mpls nodes
for (Qiu) {$i < 6} {incr i} {
            for {set j [expr $i+1]} {$j < 9} {incr j} {
                  set a n$i
                  set b n$j
                  eval $ns LDP-peer $$a $$b
            }
      }
```

ANNEXE V

SCRIPT TCL DE SIMULATION

```
#Schedule of the simulation
$ns at 0 "record"

for {set i 2} {$i < 7} {incr i} {
      set a n$i
      set m [eval $$a get-module "MPLS"]
      eval set n$i $m
      $m enable-reroute "new"
}
for {set i 100} {$i < 171} {incr i} {
      set a udp$i
      set b [expr 5 + [expr $i -100]* 0.2]
      set c [expr $b + 20]
      $ns at $b "$$a start"
      $ns at $c "$$a stop"
}
for {set i 1} {$i < 2} {incr i} {
      set a udp$i
      $ns at 0.0 "$$a start"
      $ns at 40.0 "$$a stop"
}
# This is for the removal of the mpls flag :
# "$NodeSource ldp-trigger-by-withdraw NodeDest -1"
      $ns at 0.2  "$n6 ldp-trigger-by-withdraw 7 -1"
      $ns at 0.2  "$n6 ldp-trigger-by-withdraw 8 -1"

# This is for the flow aggregation until the end :
# "$nodeSource flow-aggregation nodeDest -1  LastNodeRouter -1"
      $ns at 0.3  "$n2 flow-aggregation 7 -1  6 -1"

      $ns at 0.5  "$n6 ldp-trigger-by-withdraw 6 -1"

#Then the route :

      $ns at 0.7  "$n2 make-explicit-route 6 4_5_6 3000 -1"
      $ns at 0.9  "$n2 flow-erlsp-install   7 -1   3000"

$ns at 40.0 "finish"

#Line to begin the simulation
$ns run
```

ANNEXE VI

CODE DE L'ALGORITHME SUR NS-2

```
#Sélection de la file à observer
set qmon [$ns monitor-queue $n5 $n6  [open tracequeue.out w] 0.1];
[$ns link $n5 $n6] queue-sample-timeout;

#Processus créé pour l'algorithme
proc algorithme {} {

#Temps des cycles de mesure
set time 0.2

#Avoir le temps courant de simulation
set now [$ns now]

#Liaison des valeurs à utiliser pour l'agorithme
$qmon instvar pkts_ pdrops_ parrivals_ pdepartures_ barrivals_

#Il arrive que qdrops >0 même sans congestion (doit être un bogue) sit
el est le cas (alors pkts_ <0) on met pkts_ à 0
if {$pkts_ < 0} {
      $qmon set pkts_ 0
}

#Algorithme de mesure de la bande passante envoyée

# Refresh the 10 Bpvalues every record cycle
for {set i 11} {$i > 1} {incr i -1} {
      set a [expr $i-1]
      set b "bwarrivals_[subst $a]"
      if {[subst $$b]>0.0} {
      set bwarrivals_$i [subst $$b]
      } else {
      set bwarrivals_$i 0.0
      }
}
set bwarrivals_1 [expr $barrivals_/$time*8/1048576]


#Sélection du mode de lissage

# Mise à jour des valeurs passées
set BPValueUsed_3 $BPValueUsed_2
set BPValueUsed_2 $BPValueUsed_1
```

```
#Pas de lissage
#set BPValueUsed_1 $bwarrivals_1

#Moyenne une seconde sans pondération
#set BPValueUsed_1 [expr [expr $bwarrivals_5 + $bwarrivals_4 +
$bwarrivals_3 + $bwarrivals_2 + $bwarrivals_1 ]/ 5]

#Moyenne Pondérée sur deux secondes de relevés
set BPValueUsed_1 [expr [expr $bwarrivals_10 + 2*$bwarrivals_9 +
3*$bwarrivals_8 + 4*$bwarrivals_7 + 5*$bwarrivals_6 + 6*$bwarrivals_5 +
7*$bwarrivals_4 + 8*$bwarrivals_3 + 9*$bwarrivals_2 + 10*$bwarrivals_1
]/ 55]

#Variable de normalisation de la bande passante
set BandNorm [expr $BPValueUsed_1/10]
#
#
# Module d'extrapolation polynomiale
#
# Checks if the bandwith is under or above the bandwith limit

      if {$BPValueUsed_1 > 10.0} {
            set predictbw1_1 9999

      } else {

            #Translation des axes pour X0=0, Y0=0, X1, Y1 and X2, Y2
            set Y1 [expr $BPValueUsed_2 - $BPValueUsed_3]
            set Y2 [expr $BPValueUsed_1 - $BPValueUsed_3]

      # Calcul a et b de P(x)=ax2+bx

            set A [expr [expr -2 * $Y1 + $Y2]/0.08]
            set B [expr [expr 0.8 * $Y1 - 0.2 * $Y2]/0.08]

      #Calcul du discriminant

            set D [expr $B * $B - 4 * $A * [expr - 10 + $bwarrivals_1]]
            if {$A>0} {
                  if {$D<0} {
                        set predictbw1_1 -999

                  } elseif {$D>0} {
                        set X1 [expr [expr $B - sqrt($D)]/[expr -2 *
$A]]
                        set X2 [expr [expr $B + sqrt($D)]/[expr -2 *
$A]]
                        if {$X1>0} {
                              set predictbw1_1 [expr $X1 - 0.4]
                        } else {
                              set predictbw1_1 [expr $X2 - 0.4]
```

```
                        }
                  } else {
                        set X [expr $B / [expr -2 * A]]
                        if {$X>0} {
                              set predictbw1_1 [expr $X - 0.4]
                        } else {
                              set predictbw1_1 -999
                        }
                  }
            } else {
                  set predictbw1_1 -999
            }
      }
}
#
#
# Interpretation des données

      #Detection de la congestion
      if {$bwarrivals_1 > 9.99 && $pdrops_ > 0} {
            puts $alert "$now Congestion en cours, $pdrops_ paquets
supprimés, émission d'alerte"
      } else {
            #Calcul des seuils d'alerte
            set Ta3 [expr exp(8 * $BandNorm)/2980 + 0.2]
            set Ta2 [expr $Ta3 * 2]
            set Ta1 [expr $Ta3 * 3]

            if {$predictbw1_1 > 0.0} {
             # Traitement niveau 3
            # Threshold BP > 80% BPmax
                  if {$BandNorm > 0.8} {
                        if { $alert_1 != 3} {
                        set alert_3 $alert_2
                        set time_alert_3 $time_alert_2
                        set alert_2 $alert_1
                        set time_alert_2 $time_alert_1
                        set alert_1 3
                        set time_alert_1 $now
                        }
                        if {$predictbw1_1 <= $Ta3 } {
                              puts $alert "$nowLe temps avant la
congestion est critique, alerte émise !"
                        }
                  } elseif {$predictbw1_1 <= $Ta3} {
            # Si dernier niveau est 2, monter à 3
                        if {$alert_1 != 3 && $alert_1 >= 2} {
                              set alert_3 $alert_2
                              set time_alert_3 $time_alert_2
                              set alert_2 $alert_1
                              set time_alert_2 $time_alert_1
```

```
                        set alert_1 3
                        set time_alert_1 $now
            # Si le dernier niveau est 1, monter à 2
                  } elseif {$alert_1 == 1} {
                        set alert_3 $alert_2
                        set time_alert_3 $time_alert_2
                        set alert_2 $alert_1
                        set time_alert_2 $time_alert_1
                        set alert_1 2
                        set time_alert_1 $now
            # Si le dernier niveau est 0, monter à 1
                  } elseif {$alert_1 == 0} {
                        set alert_3 $alert_2
                        set time_alert_3 $time_alert_2
                        set alert_2 $alert_1
                        set time_alert_2 $time_alert_1
                        set alert_1 1
                        set time_alert_1 $now
                  }
                  set moy_time_alert_ [expr [expr $now -
$time_alert_3] /3]

            # Si la prediction < Ta3 et ce n'est pas une rafale,
emission d'alerte
                if {$predictbw1_1 <= $Ta3 && $moy_time_alert_ > 0.3 &&
$alert_1 == 3} {
                        puts $alert "$nowLe temps avant la
congestion est critique, alerte émise !"
                }
```

BIBLIOGRAPHIE

A. Capone, L. Fratta et F. Martignon. 2003. « Dynamic routing of bandwidth guaranteed connections in MPLS networks ». *International J. Wireless and Optical Commun.*, vol. 1, n° 1, p. 75–86.

Adami, D., D. C. Callegari, S. Giordano et M. Pagano. 2007. « A New Path Computation Algorithm and Its Implementation in NS2 ». In *Communications, 2007. ICC '07. IEEE International Conference on* (24-28 June 2007). p. 536-541. <10.1109/ICC.2007.94>.

Artichow. 2006. « Artichow, librairie graphique pour PHP ».

Arumaithurai, M., R. Geib, R. Rex et Fu Xiaoming. 2009. « Pre-congestion notification-based flow management in MPLS-based DiffServ networks ». In *Performance Computing and Communications Conference (IPCCC), 2009 IEEE 28th International* (14-16 Dec. 2009). p. 57-64. <10.1109/PCCC.2009.5403827>.

B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang. 1998. « RFC 2309 : Recommendations on Queue Management and Congestion Avoidance in the Internet ». <http://www.ietf.org/rfc/rfc2309.txt>.

Beverly, Robert. 2002. « RTG: A Scalable SNMP Statistics Architecture for Service Providers ». In *Proceedings of LISA '02: Sixteenth Systems Administration Conference*. p. 167-174. Berkeley, CA.

Cisco. 2010. « Visual Networking Index ». <http://www.cisco.com/en/US/netsol/ns827/networking_solutions_sub_solution.html>.

Dekeris, B., et L. Narbutaite. 2004. « Traffic control mechanism within MPLS networks ». In *Information Technology Interfaces, 2004. 26th International Conference on* (10-10 June 2004). p. 603-608 Vol.1. <10.1109/ITI.2004.242472>.

Eardley, P. 2009. « RFC 5559 : Pre-Congestion Notification (PCN) Architecture ». <http://tools.ietf.org/html/rfc5559>.

Hersent, Olivier, David Gurle et Jean-Pierre Petit. 2006. *La voix sur IP : déploiement des architectures VoIP, IMS et TISPAN, Protocoles SIP 3GPP et IETF, H.323, MGCP*, 2e éd. Paris: Dunod, xviii, 749 p. p.

Ishiguro, Kunihiro. 2010. « Quagga Routing Suite ». <http://www.quagga.net/>.

Kundu, Dinangkur, S. M. Ibrahim Lavlu, Andrei-Silviu Marinache, J. P. Pasnak et Books24x7 Inc. 2009. *Cacti 0.8 Network Monitoring [ressource électronique] : monitor your network with ease!* Birmingham, UK: Packt Publishing.

NLANR. 2010. « Iperf ». <http://iperf.sourceforge.net/>.

Oulai, D., S. Chamberland et S. Pierre. 2007. « A New Routing-Based Admission Control for MPLS Networks ». *Communications Letters, IEEE,* vol. 11, n° 2, p. 216-218.

Owezarski, Philippe. 2006. « Contribution de la métrologie Internet à l'ingénierie des réseaux ». LAAS-CNRS, 133 p.

Qiu, B. 1997. « A predictive fuzzy logic congestion avoidance scheme ». In *Global Telecommunications Conference, 1997. GLOBECOM '97., IEEE*. Vol. 2, p. 967-971 vol.2.

Reeves, D. S., et H. F. Salama. 2000. « A distributed algorithm for delay-constrained unicast routing ». *Networking, IEEE/ACM Transactions on,* vol. 8, n° 2, p. 239-250.

Salvadori, E., et R. Battiti. 2003. « A load balancing scheme for congestion control in MPLS networks ». In *Computers and Communication, 2003. (ISCC 2003). Proceedings. Eighth IEEE International Symposium on* (30 June-3 July 2003). p. 951-956 vol.2. <10.1109/ISCC.2003.1214239>.

Stallings, William. 1999. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*, 3rd. Reading, Mass.: Addison-Wesley, xiv, 619 p. p.

Zhiqun, Zhang, Shao Xu et Ding Wei. 2001. « MPLS ATCC: an active traffic and congestion control mechanism in MPLS ». In *Info-tech and Info-net, 2001. Proceedings. ICII 2001 - Beijing. 2001 International Conferences on* (2001). Vol. 2, p. 222-227 vol.2. <10.1109/ICII.2001.983581>.