THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Réseaux, Télécommunications, Systèmes et Architecture

Présentée et soutenue par :

M. ALI SANHAJI

le mardi 29 novembre 2016

Titre :

Nouveaux paradigmes de contrôle de congestion dans un réseau d'opérateur

---

Ecole doctorale :

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche :

Institut de Recherche en Informatique de Toulouse (I.R.I.T.)

Directeur(s) de Thèse :

M. ANDRE LUC BEYLOT

Rapporteurs :

M. GUILLAUME URVOY-KELLER, UNIVERSITE DE NICE SOPHIA ANTIPOLIS

M. STEFANO SECCI, UNIVERSITE PIERRE ET MARIE CURIE

Membre(s) du jury :

Mme ANNIE GRAVEY, TELECOM BRETAGNE CAMPUS DE BREST, Président

M. ANDRE LUC BEYLOT, INP TOULOUSE, Membre

M. PHILIPPE CADRO, ORANGE LABS LANNION, Membre

M. PHILIPPE NIGER, ORANGE LABS LANNION, Membre

« À quoi la musique fait appel en nous, il est difficile de le savoir. Ce qui est certain, c'est qu'elle touche une zone si profonde que la folie elle-même n'y saurait pénétrer. »

Emil Michel Cioran

« Faire des enfants, rien de mieux ; en avoir, quelle iniquité ! »

Jean-Paul Sartre

« Il faut être toujours ivre. Tout est là : c'est l'unique question. Pour ne pas sentir l'horrible fardeau du Temps qui brise vos épaules et vous penche vers la terre, il faut vous enivrer sans trêve.
Mais de quoi ? De vin, de poésie ou de vertu, à votre guise. Mais enivrez-vous. »

Charles Baudelaire

« Sans la musique, la vie serait une erreur, une besogne éreintante, un exil. »

Friedrich Wilhelm Nietzsche

REMERCIEMENTS

Je voudrais tout d'abord remercier mon directeur de thèse, André-Luc Beylot, de m'avoir soutenu depuis le moment où je lui ai avoué vouloir mener un projet de thèse, bien avant d'avoir obtenu mon diplôme d'ingénieur. Je le remercie d'avoir suivi attentivement le déroulement de la thèse, malgré la distance géographique entre nous, de m'avoir dirigé tout au long de ma thèse et poussé pour la mener à bien jusqu'au bout.

Ensuite, je voudrais remercier mes encadrants à Orange, Philippe Niger et Philippe Cadro. Je les remercie de m'avoir donné l'opportunité de mener une thèse à Orange. J'ai beaucoup appris ici, avec eux, grâce à eux, techniquement comme humainement, ils m'ont apporté critiques et encouragements qui m'ont formé, et leur participation à l'élaboration de nos articles et de ma thèse n'était pas des moindres. Je leur dois beaucoup, et je les remercie pour tout.

Je remercie Marc Le Guigner, responsable de mon ancienne équipe MAA, qui m'a accueilli à Orange, et donné la possibilité de participer à plusieurs événements très enrichissants durant mes deux premières années ici. Je remercie bien sûr tous mes anciens collègues de l'équipe MAA, ceux qui ont toute leur tête et surtout ceux qui n'en ont pas. Les moments passés avec eux étaient exceptionnels et je m'en souviendrai toujours : les cafés, les thés, les rires, les mauvais coups, et les questions de doigté.

Je remercie Maria Luisa Guerra Feliz De Vargas, responsable de ma nouvelle équipe NAVI, de m'avoir soutenu durant ma dernière année de thèse, de m'avoir donné la liberté de me concentrer complètement sur la conclusion de mes travaux de recherche, et de m'avoir donné l'opportunité d'intégrer son équipe de manière plus pérenne. J'aimerais remercier mes nouveaux collègues de la « marine » de m'avoir bien accueilli parmi eux, j'ai appris à

les connaître et j'espère passer encore de bons moments avec eux et naviguer ensemble sur les nouvelles mers.

Je ne voudrais pas oublier l'équipe IRT qui m'a si bien accueilli quand je suis passé les voir à Toulouse. Je remercie tous les professeurs et les doctorants avec qui j'ai pu échanger pour enrichir ma réflexion, et que j'ai pu revoir avec joie à Lyon à l'occasion de l'école d'été RESCOM. Les quelques jours passés avec eux sont des souvenirs précieusement gardés.

Beaucoup de personnes m'ont aidé, assisté, éclairé durant mes années de thèse, et je leur suis très reconnaissant, mais je ne voudrais pas oublier de mentionner ceux qui m'ont le plus conseillé techniquement et auprès desquels j'ai beaucoup appris. D'abord, Cédric Ollivier, notre grand expert des distributions manchots, vers qui je me suis toujours tourné quand je butais sur un problème technique ou un souci sur une plate-forme ; l'homme qui a le plus contribué à la debianisation des esprits a toujours été là pour nous sortir des impasses, et je l'en remercie beaucoup. Je voudrais remercier également Patrick Truong, François Eleouet, Mathieu Rohon et Thomas Morin de m'avoir aidé dans l'élaboration de mes plates-formes expérimentales, de m'avoir donné des idées pour continuer mes recherches et de m'avoir guidé sur les bonnes routes grâce à leur grande expertise technique.

Enfin, et je ne sais pas s'il est possible de remercier une région, un pays, une terre, mais je voudrais remercier la Bretagne. Malgré les journées grises d'hiver, mélancoliques et sourdes à la musique du cœur ; malgré la pluie qui s'invite jour et nuit sur nous, en toutes saisons, partout (car en vérité, elle est chez elle et nous sommes les fugitifs invités) ; malgré les journées où la lumière garde la même intensité, basse, sombre et filtrée par une couche de nuages qui est un autre ciel, toute la journée avec la même lumière comme une unique lampe pour toute cette terre celtique, comme si le soleil s'était perdu ou, aveugle, regardait ailleurs, ou comme si le temps était suspendu sur une seule et même heure jusqu'à la soudaine tombée de la nuit ; malgré tout ce que l'on fuit ici, la Bretagne est une terre que la beauté et le sublime ont enfantée. Plus que la remercier, je voudrais dire combien j'aime cette région, la mer dont le son berce et calme le sang agité, l'horizon riche d'îles, de rochers, de paysages insensés de splendeurs et de grâce, le granit rose qui vous porte haut au-dessus des vagues qui se fracassent sur terre, et le vent d'ouest qui emporte vers d'autres espaces vos soucis et vos peines. Et voici que vous revenez de la mer l'âme pleine de splendeurs et de gratitude d'être proche de tant de beauté et d'avoir saisi l'essence des choses, alors ce que je peux dire, si l'on peut m'entendre : cette Bretagne est véritablement grande.

RÉSUMÉ

La congestion dans les réseaux est un phénomène qui peut influer sur la qualité de service ressentie par les utilisateurs. L'augmentation continue du trafic sur l'internet impose aux opérateurs de répondre efficacement aux phénomènes de congestion qui apparaissent pour satisfaire leurs clients. Ils sont alors confrontés à une double exigence vis-à-vis de ces utilisateurs : leur offrir de la qualité de service et assurer une équité. Les solutions historiques de traitement de la congestion pour un opérateur, comme le surdimensionnement des liens de son infrastructure, ne sont plus aujourd'hui viables. Avec l'évolution de l'architecture des réseaux et l'arrivée de nouvelles applications sur l'internet, de nouveaux paradigmes de contrôle de congestion doivent être envisagés.

Dans cette thèse, nous étudions les nouvelles approches proposées pour le contrôle de congestion dans le réseau d'un opérateur, notamment Congestion Exposure (ConEx), un mécanisme qui permet aux utilisateurs d'exposer le niveau de congestion qu'ils ont rencontré dans le réseau. Cela permet à l'opérateur de limiter les utilisateurs du réseau proportionnellement à la congestion exposée. Nous proposons une évaluation de ConEx à travers des simulations, et nous montrons son efficacité pour établir l'équité entre les utilisateurs du réseau et la possibilité de le déployer par étapes incrémentales dans le réseau de l'opérateur.

Nous proposons également des solutions de traitement de la congestion appliquées à l'environnement cloud. Le cloud, déployé sur un ou plusieurs data centers, est de plus en plus utilisé par les entreprises pour offrir des capacités de traitement de données et de stockage pour leurs services internet. Cela engendre une grande quantité de trafic qui provoque un risque de congestion menant à une dégradation de la qualité des services cloud proposés. Nous montrons comment nos solutions s'insèrent dans les environnements intra et inter-data centers, et permettent le traitement de la congestion tout en prenant en compte la spécificité de chaque environnement. Pour appuyer nos propositions d'architectures de

traitement de la congestion, nous avons mis en place des plates-formes expérimentales qui montrent le fonctionnement et le potentiel de nos solutions.

ABSTRACT

Network congestion is a phenomenon that can influence the quality of service experienced by the users. The continuous increase of internet traffic makes this phenomenon an issue that should be addressed by the network operator to satisfy its clients. They have to respond to two requirements towards their users : offer quality of service and ensure fairness. The usual solutions to congestion for an operator, such as overdimensioning the infrastructure, are not viable anymore. With the evolution of the network architecture and the emergence of new internet applications, new paradigms for congestion control have to be considered.

In this thesis, we examine new approaches to congestion control in an operator's network, in particular ConEx, a mechanism which allows the users to expose the level of congestion they encounter in the network. It allows the network operator to limit the network users proportionally to the exposed congestion. We propose an evaluation of ConEx through simulations, and we show its efficiency in establish fairness between network users and its gradual deployability in the network.

We also propose solution for congestion control applied to the cloud environment. The cloud, deployed over one or many data centers, is more and more used by companies to offer the compute and storage capacities for their internet services. This generates a great amount of traffic which causes a congestion risk, which could lead to a degradation of cloud services quality. We show how our solutions can be used in the intra and inter-data centers environments, and allow congestion control while taking into consideration the specificity of each environment. To support our architecture propositions for congestion control, we present experimental platforms which show the operation and potential of our solutions.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

ACRONYMES

AC Attachment Circuit. 95, 102

ACE Accurate ECN. 35

ADSL Asymmetric Digital Subscriber Line. 10, 61

AIMD Additive Increase/Multiplicative Decrease. 13, 18

API Application Programming Interface. 69, 73, 79–82, 94–97, 99–104, 106, 107, 112, 118

AQM Active Queue Management. 16, 18

ARED Adaptive Random Early Detection. 18

BCN Backward Congestion Notification. 20, 29

BDP Bandwidth Delay Product. 40

BGP Border Gateway Protocol. 71

BIC Binary Increase Congestion control. 13

CAPI Control Application Programming Interface. 94, 100–104, 106–108, 112, 113, 118–121

CBR Constant Bit Rate. 109

CE Congestion Experienced. 17

CMTS Cable Modem Termination System. 8

CoDel Controlled Delay. 18

ConEx Congestion Exposure. 2–4, 24, 25, 29, 31–39, 41, 42, 44–52, 54, 55, 58–63, 65–67, 78, 116–119, 121

CRUD Create, Retrieve, Update and Delete. 102

CWR Congestion Window Reduced. 35

DCTCP Data Center TCP. 34

DHCP Dynamic Host Configuration Protocol. 80

DiffServ Differentiated Services. 2, 19

DSCP Differentiated Services Code Point. 26, 60, 82

ECE ECN-Echo. 17, 20, 34, 35

ECMP Equal-Cost Multi-Path. 71

ECN Explicit Congestion Notification. 17–20, 23, 24, 29, 33–36, 38, 40, 46, 49–56, 58, 76, 82–84, 91, 97, 117

EWMA Exponentially Weighted Moving Average. 76, 77, 82, 99

FCC Federal Communications Commission. 24

FIFO First In, First Out. 16

FTP File Transfer Protocol. 39, 62, 65

FTTH Fiber To The Home. 31, 61

GRED Gentle Random Early Detection. 18

HTTP HyperText Transfer Protocol. 65, 101

IETF Internet Engineering Task Force. 2, 5, 17–19, 21, 24, 29, 31, 32, 34, 35, 76, 100, 115, 116

IntServ Integrated Services. 2

IP Internet Protocol. 7, 15, 17, 19, 26, 33, 34, 54, 76, 80, 97

IPv4 Internet Protocol version 4. 18, 34, 39

IPv6 Internet Protocol version 6. 34

IS-IS Intermediate System to Intermediate System. 27

JSON JavaScript Object Notation. 102, 104

JUNG Java Universal Network/Graph. 103

KVM Kernel-based Virtual Machine. 72, 80

LEDBAT Low Extra Delay Background Transport. 62, 64, 65, 67, 116

LSP Label Switched Path. 27, 95

LXC Linux Containers. 80

MIB Management Information Base. 20

MPLS MultiProtocol Label Switching. 27, 95

MPLS-TE MultiProtocol Label Switching-Traffic Engineering. 27

NETCONF Network Configuration Protocol. 21, 29, 101

NFV Network Function Virtualization. 3, 101, 120

NS Nonce Sum. 35

NS2 Network Simulator 2. 39, 61, 65

ODL OpenDaylight. 95, 98, 101–103, 106, 118

OSGi Open Services Gateway initiative. 101

OSPF Open Shortest Path First. 27, 71

OTT Over-The-Top. 9, 10

OVS Open vSwitch. 80, 84

PCN Pre-Congestion Notification. 2, 5, 19, 21, 22, 94, 97–100, 104, 106, 107, 109, 110, 112, 118

PE Provider Edge. 4, 95, 96, 102, 104

PIE Proportional Integral controller Enhanced. 18

PW Pseudo Wire. 95

QCN Quantized Congestion Notification. 20, 29

QoE Quality of Experience. 61–65, 67

QoS Quality of Service. 1, 10, 81–84, 91, 94, 112, 113

RED Random Early Detection. 16–18, 40, 46, 48, 49, 51, 54, 55, 57, 58, 66, 84, 85

REST REpresentational State Transfer. 102, 112

RFC Requests For Comments. 39

RPC Remote Procedure Call. 80

RSVP Resource Reservation Protocol. 2, 27

RSVP-TE Resource Reservation Protocol-Traffic Engineering. 27

RTO Retransmit TimeOut. 60

RTT Round-Trip Time. 12, 13, 20, 34, 38, 40, 42, 45–48, 50, 52, 55, 58, 66, 118

SACK Selective Acknowledgment. 13, 35, 40

SDN Software Defined Networking. 3–5, 22, 25, 29, 30, 73, 93–95, 97–101, 108, 112, 118–120

SLA Service Level Agreement. 23, 29, 95

SNMP Simple Network Management Protocol. 20, 29

ssthresh Slow-Start Threshold. 12, 13

TCP Transmission Control Protocol. 11–15, 17, 19, 20, 23, 25, 28, 29, 32–35, 39–42, 46, 48, 54, 55, 60–62, 65, 66

TE Traffic Engineering. 27

URI Uniform Resource Identifier. 102, 104

VLAN Virtual Local Area Network. 71, 80, 95, 102

VM Virtual Machine. 72, 74, 76–91, 95, 107, 108

VoIP Voice over IP. 26

VPN Virtual Private Network. 4, 5, 93–104, 106–113, 118–120

VxLAN Virtual Extensible Local Area Network. 73, 80, 82

WAN Wide Area Network. 4, 93, 96, 97, 112, 113, 118

INTRODUCTION

L'internet est aujourd'hui en tête dans les infrastructures d'échange d'informations. C'est un grand réseau sur lequel se trouve la majorité des données circulant à l'échelle mondiale : des transactions financières, des photos personnelles, des vidéos de manifestations, de la musique en *streaming*, des informations sur les utilisateurs... Jamais un réseau n'a été aussi imposant que l'internet, servant autant à la plus humble recherche d'informations qu'aux échanges les plus secrets, et la tendance n'est guère à la baisse, l'usage est de plus en plus naturel. Ainsi le trafic global augmente, les demandes en débit sont importantes, les données échangées sont de plus en plus gigantesques, le succès de certains acteurs motive l'entrée d'autres, et les liens de l'internet s'encombrent, alors que tout le monde est en constante demande de débit.

Les opérateurs de réseaux sont les premiers affectés par cette évolution effrénée. Étant en première ligne devant leurs clients, ils ont le rôle de transporter leurs requêtes dans le maillage complexe de l'internet en leur délivrant la qualité promise lors de leur souscription. Il y a donc une exigence de qualité à maintenir dans les efforts de l'opérateur, mais aussi d'équité qu'il doit établir entre les utilisateurs d'un même service. Ce besoin de qualité de service (Quality of Service (QoS)) est aussi renforcé par le fait que l'internet devient un réseau fédérateur qui doit pouvoir prendre en charge toutes les applications, y compris les plus exigeantes.

L'augmentation des débits a toujours appelé une réponse évidente : le surdimensionnement des liens, un automatisme qui a longtemps fonctionné, simple en raisonnement, simple en mise en place, simple en justification, aux conséquences immédiatement perceptibles, sauf que les automatismes cèdent devant les obstacles grandissants. Le surdimensionnement

des liens n'est pas une solution indéfiniment viable, elle a un certain coût qui fait face un obstacle financier qui est celui de la stagnation des revenus des opérateurs. De plus, son éventuelle mise en place s'opère sur une longue échelle de temps.

Sur une échelle de temps plus courte, plusieurs propositions ont été apportées par les opérateurs, par la communauté de la recherche et plus globalement par la communauté de l'internet pour l'amélioration et le maintien de la qualité de service. Sans agir en temps réel, le modèle Integrated Services (IntServ) [4] se voulait capable de garantir un certain niveau de service en réservant, à travers la signalisation Resource Reservation Protocol (RSVP), les ressources nécessaires sur tout le chemin emprunté par le flux à garantir, et ceci individuellement pour chaque flux, ce qui pose des problèmes de mise à l'échelle.

Enfin, sur une échelle de temps quasi temps réel, avec une précision de niveau paquet, une solution comme Differentiated Services (DiffServ) [5] permet de prioriser un certain trafic par rapport à un autre en établissant plusieurs classes de service, privilégier la voix à la vidéo par exemple, la vidéo à la consultation de sites web également. En revanche, cette approche ne permet pas de prendre en compte l'état réel du réseau et la charge de trafic à écouler, donc les risques de congestion. Ainsi, de nouvelles solutions, ont été proposées par la communauté de l'internet, notamment par l'Internet Engineering Task Force (IETF) qui, ces dernières années, a tenté d'approcher d'une nouvelle manière le problème de la congestion ou du risque de congestion dans le réseau, de modifier sa perception du phénomène en cessant de le considérer uniquement comme problématique et hasardeux pour les utilisateurs, mais en l'exploitant également en tant que ressource reflétant l'état du réseau et la qualité potentielle du trafic transporté. L'idée serait donc de prendre en compte les informations sur la charge de trafic et sur la congestion disponibles dans le réseau, et de les utiliser comme métriques pour répartir la bande passante entre les utilisateurs. C'est dans cette logique que l'IETF a proposé des mécanismes tels que Pre-Congestion Notification (PCN) [6] et ConEx [7].

Nous nous fonderons donc sur de telles approches pour proposer, dans les différents environnements où un opérateur comme Orange est en activité, des solutions à notre problématique qui est celle de la bonne exploitation des informations provenant du réseau afin d'avoir l'action la plus précise sur les causes, ou le risque, d'une dégradation de la qualité de service. Les informations sur la charge du réseau et le risque de congestion ou même l'occurrence de la congestion seront les éléments à exploiter pour atteindre nos objectifs.

Objectifs et plan

Orange opère principalement comme opérateur de réseau étendu et de fournisseur d'accès à l'internet. Il est naturel que notre attention se focalise d'abord sur cet environnement où les contraintes sont nombreuses : bande passante limitée, latence importante et variable, hétérogénéiété des équipements d'extrémité, etc. En outre, aux extrémités de ce réseau peuvent se trouver des utilisateurs aux comportements très distincts. L'impact sur l'état de congestion du réseau sera potentiellement très différent d'un type d'utilisateur à un autre. En effet, les utilisateurs n'ont pas le même impact sur le réseau ; certains en ont une utilisation bien plus intensive que d'autres. Leur trafic excessif nuit à la qualité de service des autres utilisateurs moins gourmands. Un déséquilibre se crée et les conséquences de cette iniquité se retournent contre l'opérateur qui doit répondre à la dégradation de la qualité de service de tous ses utilisateurs. C'est au travers du mécanisme ConEx, qui rend compte de l'influence de chaque utilisateur sur la congestion du réseau, que nous tenterons de remédier à ce problème, tout en prenant en considération les contraintes sus-citées. Nous présenterons une évaluation des performances du mécanisme, ainsi qu'une proposition pour un déploiement graduel dans le réseau. Nous aurons alors à discuter les difficultés qui peuvent entraver son implantation et son adoption.

Ensuite, nous nous focaliserons sur un autre environnement où Orange est de plus en plus présent et qui, dans le contexte des réseaux d'aujourd'hui, occupe une place prépondérante : le cloud et les data centers. C'est effectivement dans ces derniers que résident beaucoup de contenus réclamés par les utilisateurs, et où s'échange en permanence en interne une grande quantité de trafic. Il peut y apparaître donc une surcharge du réseau qu'il faut prendre en compte. Nous aurons donc comme objectif de permettre la récupération d'informations sur la charge de trafic et sur la congestion et de l'utiliser comme reflet de l'état du réseau et un indicateur du moment où notre réaction est nécessaire pour permettre une bonne qualité des services cloud. En revanche, en liant le déploiement des architectures Software Defined Networking (SDN) et Network Function Virtualization (NFV), l'introduction de la qualité de service présente de nouveaux défis, notamment des problèmes liés à son implantation au travers d'un contrôleur centralisé comme nous allons le voir.

Nous allons envisager l'environnement cloud en deux parties. Nous nous focaliserons d'abord sur la partie intra-data center, puis sur la partie inter-data centers ; elles présentent des caractéristiques différentes et ont donc besoin d'études distinctes :

Intra-data center

L'environnement intra-data center se caractérise par une utilisation massive de la virtualisation pour offrir aux *tenants* (les utilisateurs du cloud) une isolation de leurs réseaux L2. Cette encapsulation des données dans un réseau virtualisé ne pose pas de problèmes en terme de débits si le réseau physique sur lequel il repose est surdimensionné, ce qui a été le cas jusqu'à présent. Mais l'augmentation très importante des volumes de trafic, particulièrement dans les data centers, rend cette approche moins fiable pour assurer une bonne qualité de service pour le cloud. De plus, la séparation logique entre le réseau physique et le réseau virtuel rend l'état du trafic difficile à évaluer et la qualité de service difficile à maintenir à cause de ce manque de visibilité et d'échanges d'informations entre les deux couches. Notre objectif sera donc de permettre une meilleure coordination entre le réseau physique et le réseau virtuel pour donner la possibilité aux contrôleurs de ce dernier de réagir aux informations provenant du premier. Nous proposerons ainsi une solution pour l'introduction de la qualité de service, reposant sur la congestion comme indicateur de performance du réseau, et exploitant nos connaissances venant de notre étude sur ConEx. Nous proposerons également une implantation de la solution sur une plate-forme open source de déploiement de clouds sur laquelle nous verrons qu'il est possible, au travers de l'utilisation des informations de congestion, de limiter l'impact négatif de certains *tenants* sur le réseau intra-data center.

Inter-data centers

La problématique dans l'environnement inter-data centers est autre. Les clients cherchent alors la possibilité de raccorder, au travers de Virtual Private Networks (VPNs), leurs sites distants de data centers qui sont séparés par un Wide Area Network (WAN) aux contraintes en bande passante, tout en ayant une qualité de service adéquate pour le bon fonctionnement de leurs applications. La difficulté est de pouvoir introduire la qualité de service dans un environnement où il n'existe pas de coordination entre le réseau WAN et les data centers. En effet, le réseau WAN et les data centers sont actuellement séparés par une interface stricte qui se trouve au niveau des Provider Edges (PEs) de l'opérateur, et qui ne permet que de remettre le trafic des data centers au WAN pour acheminement sans permettre une connaissance de son état de charge afin d'avoir une meilleure exploitation du cloud. Il nous faut d'abord un moyen d'y parvenir avant d'envisager l'introduction de la qualité de service. Par conséquent, nous proposerons d'abord une architecture d'orchestration qui repose sur un contrôleur SDN pour le contrôle du réseau WAN et l'établissement des VPNs, et sur

un orchestrateur qui coordonne les opérations entre ce contrôleur SDN et les contrôleurs des plates-formes cloud localisés dans les data centers distants. Ensuite, nous pourrons exploiter et adapter les mécanismes de PCN, définis par l'IETF, pour le monitoring de l'état de charge du réseau et des VPNs reliant les data centers, et permettre l'utilisation de cette information de monitoring par le contrôleur SDN et par l'orchestrateur dans le but d'améliorer la qualité de service du client. Nous présenterons enfin une plate-forme expérimentale avec laquelle nous montrerons le fonctionnement de l'architecture proposée et des possibilités qu'elle offre.

CHAPITRE 1

TRAITEMENT DE LA CONGESTION DANS LE RÉSEAU D'UN OPÉRATEUR

Tout au long de ce chapitre et plus globalement dans cette thèse, nous traiterons du problème ou du moins du phénomène de congestion, plus particulièrement celui qui peut atteindre les applications des réseaux Internet Protocol (IP). Commençons d'abord par définir précisément ce terme. Le Larousse définit la congestion ainsi : « État d'un lieu de passage, d'un réseau qui est fortement encombré. » [8] C'est très simple donc pas suffisamment précis, mais c'est un début de réponse. Dans le monde des réseaux de communication, on parle de congestion lorsque la quantité de trafic qui doit transiter par le réseau dépasse les capacités de celui-ci à la traiter et la transporter. On se rapproche d'une définition qui pourrait nous permettre de savoir comment l'appréhender avec précision, mais arrivé à ce point, la congestion devient plus difficile à identifier, car sa définition exacte n'est ni unanime ni normalisée. On en connaît les conséquences (perte de paquets, effondrement des débits, augmentation des délais) mais on ne saurait être d'accord sur quand le phénomène commence et quand il finit.

1.1 Qu'est-ce que la congestion ?

Comme le rappelle très bien [9], plusieurs experts peuvent donner une définition très précise de la congestion, différentes selon leur point de vue et leur domaine d'expertise. Ainsi, d'après la théorie des files d'attente, la congestion peut être considérée quand le taux d'arrivée dans un système dépasse le taux de service de ce système pendant une certaine

période de temps. Donc dès qu'une file d'attente dans un équipement réseau commence à accumuler des paquets, on peut dire que l'on observe de la congestion sur cet équipement.

Le point de vue de certains experts réseau serait plutôt de définir la congestion comme le moment où le buffer sur une interface d'un équipement arrive à saturation, qu'il ne peut plus stocker de nouveaux paquets et qu'il y a par conséquent une perte de paquets. La congestion se mesurerait donc par le taux de paquets perdus sur cette interface.

Le point de vue d'autres experts réseau, notamment chez un opérateur, est différent. L'opérateur cherche à éviter la congestion, et en même temps à garantir une équité entre ses utilisateurs. Il peut donc parler de congestion et de traitement de la congestion avant même la saturation des liens. Dans [10], Comcast propose un système de traitement de congestion qui agit lorsque l'utilisation d'un port Cable Modem Termination System (CMTS) dépasse un seuil configuré pendant une certaine période de temps. Un exemple de configuration donné dans leur proposition est d'un seuil de 80% de la bande passante du port sur une période de 15 minutes en ce qui concerne les ports descendants. Dans ce cas-là, le réseau est encore capable de transporter le trafic des utilisateurs, donc il n'y a pas réellement de congestion, mais il se peut qu'un excès de trafic fasse entrer le réseau dans une phase de congestion, c'est pour cela qu'on devrait plutôt parler de risque imminent de congestion ou même de pré-congestion [6] plutôt que de congestion.

Par souci de cohérence dans le reste de cet écrit, nous allons préciser les notions autour du concept de congestion et donner des définitions qui bien évidemment s'approchent des points de vue précédents. Nous dirons donc qu'il y a congestion lorsque les capacités du réseau (bande passante, files d'attente) sont entièrement utilisées, et en conséquence, des pertes de paquets ou une augmentation importante des délais de transit contribuent à la dégradation de la qualité de service de l'utilisateur dans le réseau de l'opérateur. Dans le cas où les liens ne sont pas saturés mais que des mécanismes sont mis en place afin de lutter contre un risque de congestion dans le réseau, nous dirons que ces mécanismes traitent la pré-congestion.

Dans cette thèse, nous nous focaliserons sur la prévention de perte de paquets dans le réseau, mais bien évidemment, le délai subira également l'influence des techniques de traitement de la congestion ou de la pré-congestion que nous proposerons.

1.2 Où se trouve la congestion et qui en est responsable ?

Dans l'offre d'un service de distribution de contenus ou d'un service d'entreprise, le trafic du serveur qui fournit le contenu transite par plusieurs sections de réseaux pour

arriver enfin à l'utilisateur (cf. figure 1.1). Plusieurs points de congestion peuvent altérer la qualité de service, et tous ne sont pas sous le contrôle de l'opérateur : le data center où est hébergé le serveur peut appartenir à un Over-The-Top (OTT) tel que Google, ou à un fournisseur de services cloud comme Amazon et son offre Amazon Web Services. Avec son architecture particulière, le data center est éventuellement sujet à de la congestion [11] et la dégradation de la qualité de service peut démarrer de là.

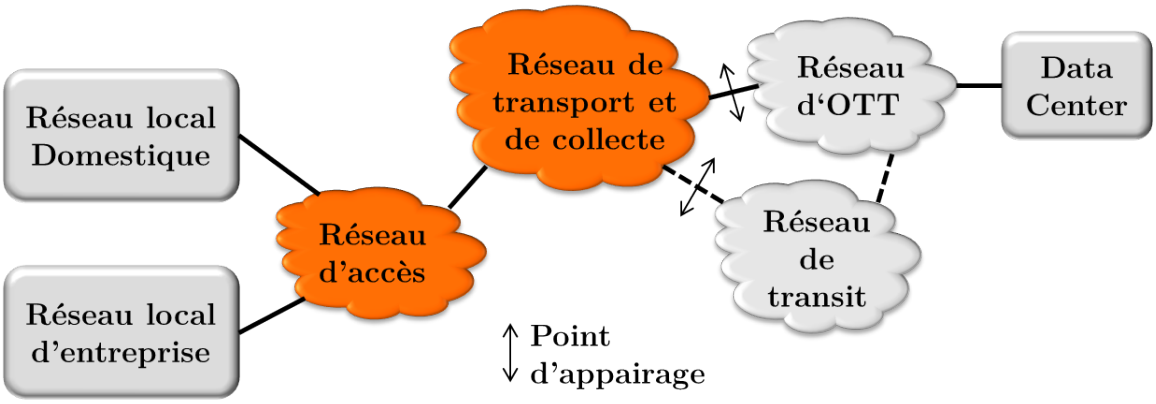

FIGURE 1.1 – Parcours du trafic jusqu'à l'utilisateur

Ensuite, le trafic du serveur qui sort du data center passe par le réseau du fournisseur de service et éventuellement par le réseau d'un transitaire avant d'atteindre le réseau de l'opérateur via une interconnexion dans un point d'échange. La congestion dans ces réseaux est à traiter par celui qui en a la charge. Les points d'échange sont des lieux très importants où l'apparition de la congestion est particulièrement envisageable. Ceci est dû au dimensionnement des interfaces d'échange entre les réseaux pairs qui sont souvent limitées, non seulement pour des raisons technologiques mais également pour des raisons politiques et financières. Ici les pairs ont une responsabilité commune face au risque de congestion et à la dégradation de la qualité de service.

Après le point d'appairage, le trafic entre dans le domaine de l'opérateur. Le réseau de transport et le réseau de collecte se caractérisent le plus souvent par des liens surdimensionnés qui peuvent transporter de très hauts débits. L'opérateur est préoccupé par la perte de paquets dans ces réseaux, c'est ce qui le pousse à surdimensionner et les liens n'y sont en général pas saturés. La pré-congestion y sera donc plus surveillée que la congestion.

Enfin, avant d'arriver aux réseaux des clients de l'opérateur, le trafic passe par le réseau d'accès. Selon la technologie utilisée, la congestion peut survenir avec plus ou moins d'intensité, dans le sens montant comme dans le sens descendant. Dans les cas des clients

résidentiels, la congestion peut survenir principalement sur la voie descendante à cause de leurs abonnements asymétriques. Plus particulièrement dans les réseaux Asymmetric Digital Subscriber Line (ADSL) et mobiles qui offrent une bande passante limitée, l'opérateur doit mettre en place des mécanismes qui vont traiter la congestion de manière à ce que la qualité de service ne se dégrade pas à cet endroit pour le trafic du client.

Quand le trafic arrive finalement au réseau du client, que ce soit dans le réseau local domestique d'un client résidentiel qui sollicite le service d'un OTT, ou que ce soit le réseau local d'un client d'entreprise qui accède à un service cloud, l'occurrence de la congestion dans ces réseaux est hors de portée de l'opérateur. Ce sont les administrateurs de ces réseaux qui y prennent en charge le risque de congestion.

Globalement, la qualité d'un service attendue par le client de l'opérateur est la résultante de la qualité de service dans chaque section de réseau traversée par le trafic. L'absence de congestion dans le réseau de l'opérateur n'implique pas forcément une bonne qualité de service globale. La chaîne de bout-en-bout est contrôlée par plusieurs acteurs, et chacun a la responsabilité du traitement de la congestion et de la qualité de service dans son réseau.

Dans un environnement aussi étendu, différents mécanismes de traitement de la congestion peuvent coexister et interagir de façon plus ou moins concertée. Typiquement, les mécanismes de bout-en-bout, où chacun des acteurs de la chaîne contribue de manière collaborative au traitement de la congestion, peuvent s'additionner aux mécanismes propres à l'opérateur qui implante dans son réseau des solutions pour agir localement au niveau des équipements du réseau en cas de congestion, ainsi qu'au niveau des points importants comme à l'entrée de son réseau au niveau des interfaces avec les fournisseurs de service, ou des interfaces avec les réseaux des clients. Les mécanismes agissant de bout-en-bout ont la capacité de traiter l'ensemble des problèmes rencontrés, avec parfois cependant une efficacité limitée. À l'inverse, les mécanismes locaux sont de portée limitée mais sont potentiellement plus efficaces et ont pour objectif de permettre à chacun des acteurs de la chaîne de garantir la qualité de service sur le segment qu'il contrôle, contribuant à la QoS globale.

Nous nous intéressons particulièrement dans cette thèse à ce que l'opérateur peut apporter dans le traitement de la congestion que ce soit dans un cadre collaboratif pour bâtir une architecture opérant de bout-en-bout ou opérant de façon autonome sur l'environnement qu'il maîtrise.

1.3 Comment traiter la congestion ?

Comme représenté sur la figure 1.2, entre deux utilisateurs du réseau d'un opérateur, un protocole de transport se charge d'établir la communication de bout-en-bout pour les applications de ces utilisateurs. Le réseau de l'opérateur est quasiment une boîte noire, ou au moins grise, mais sûrement opaque dont on peut estimer quelques caractérisques des chemins empruntés par les flux comme le débit, le délai de transit de bout-en-bout, et possiblement le taux de perte, mais où les machines d'extrémité ont peu de visibilité sur les actions de l'opérateur dans le cas d'un risque de congestion.

Le « end-to-end argument » [12] suggérait que les fonctions de bas niveau étaient inutiles face au traitement de bout-en-bout, et il a longtemps prévalu, en particulier comme principe d'architecture de l'internet. En conséquence, c'est le protocole de transport, à l'instar de TCP, qui s'est chargé d'implanter des algorithmes de contrôle de congestion (qui agiront donc de bout-en-bout) et qui tente de s'adapter aux caractéristiques du réseau de l'opérateur.

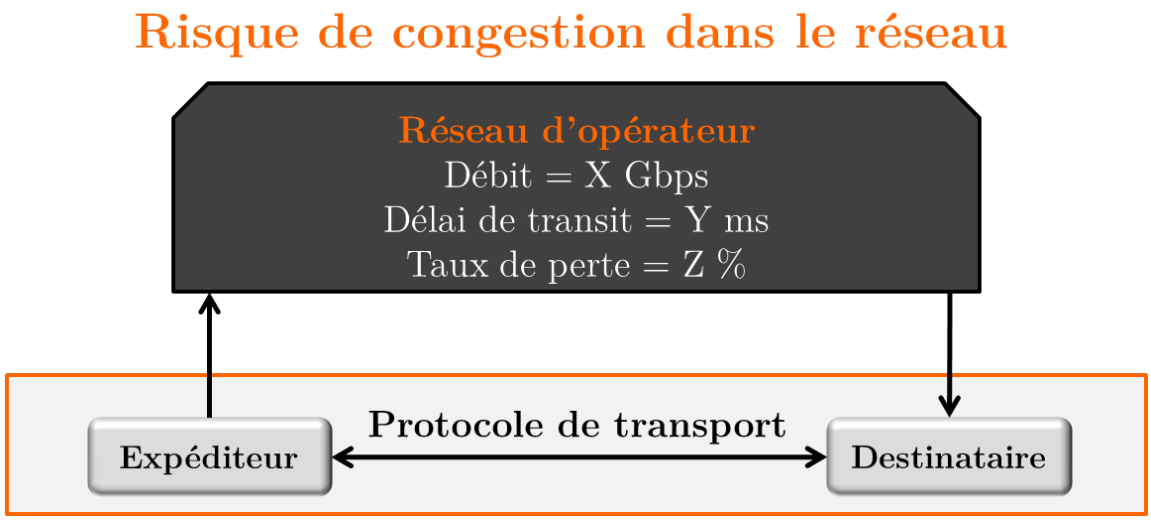

FIGURE 1.2 – Risque de congestion dans le réseau

1.3.1 Une approche de bout-en-bout : l'approche TCP

La première réponse apportée à la congestion dans le réseau est venue de [13], qui propose un algorithme de congestion avoidance pour TCP reposant sur plusieurs stratégies pour éviter de continuer d'encombrer le réseau en cas de congestion. Lorsqu'une connexion

TCP est établie entre deux utilisateurs, l'expéditeur initialise une fenêtre de congestion (cwnd) à un petit nombre de segments (p.ex. 2 ou 4) à envoyer (cf. figure 1.3). À chaque réception de segment, le destinateur informe l'expéditeur de sa bonne réception au travers d'accusés de réception. Pour chaque accusé de réception, l'expéditeur augmente sa fenêtre congestion d'un segment, ce qui revient à doubler la fenêtre de congestion au bout d'un RTT. La fenêtre de congestion peut continuer de s'agrandir jusqu'à ce qu'elle atteigne la fenêtre de réception annoncée par le destinataire, ou qu'elle atteigne une valeur maximale configurée. C'est le mécanisme de slow-start.

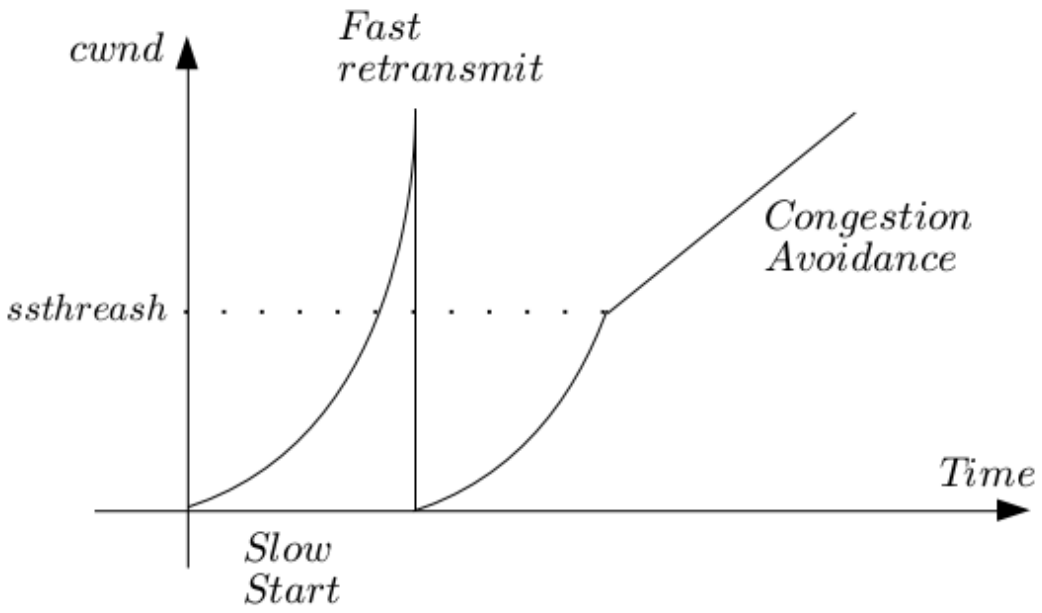

FIGURE 1.3 – Fenêtre de congestion de TCP

Un autre événement peut interrompre l'évolution de la fenêtre de congestion, c'est la perte d'un segment. Le destinataire envoie toujours l'accusé de réception pour le dernier segment en séquence reçu. Donc s'il y a perte dans le réseau, l'expéditeur reçoit plusieurs accusés de réception dupliqués. Le mécanisme de fast-retransmit consiste à décréter qu'il y a eu perte de segment, signe d'une congestion dans le réseau, au bout d'un certain nombre d'accusés de réception dupliqués (typiquement trois). Le segment considéré comme perdu doit alors être retransmis, la moitié de la fenêtre de congestion courante est enregistrée en tant que Slow-Start Threshold (ssthresh), et l'expéditeur redémarre en slow-start (pour la version Tahoe de TCP). Lorsque la fenêtre de congestion augmente au rythme des nouveaux

acks et atteint cette fois-ci ssthresh, c'est là que l'expéditeur entre en phase de congestion avoidance. Cela consiste à n'augmenter la fenêtre de congestion que d'un seul segment à chaque RTT, mais toujours à réduire de moitié la fenêtre de congestion en cas de perte. C'est le principe d'Additive Increase/Multiplicative Decrease (AIMD).

Beaucoup d'améliorations ont été proposées à l'algorithme de congestion de TCP : le fast recovery pour ne pas retomber en slow-start en cas de congestion [14], le Selective Acknowledgment (SACK) pour informer l'expéditeur avec plus de précision des segments reçus [15]. Des variantes plus ou moins aggessives de TCP ont été proposées (Binary Increase Congestion control (BIC), CUBIC) pour contrôler la fenêtre de congestion. D'autres variantes se focalisent surtout sur le délai pour contrôler celle-ci (Vegas), et encore d'autres sur une combinaison de perte et de délai (Compound). Beaucoup de contributions ont été faites dans ce sens car, longtemps, TCP a été l'élément le plus important pour le contrôle de congestion.

Envoyer des segments avec de plus en plus d'intensité jusqu'à la détection d'une perte, puis continuer avec beaucoup plus de prudence à l'étape qui suit est une manière de sonder le réseau pour en connaître l'état, et en cas de congestion, d'éviter de provoquer plus de pertes. Mais c'est surtout une manière de procéder qui prouve combien le réseau est un environnement opaque qui agit très peu pour traiter ou aider à traiter la congestion qui peut survenir en son sein.

1.3.2 Le traitement de la congestion par l'opérateur

Jusqu'ici, les algorithmes de contrôle de congestion de TCP ont tenté de s'adapter aux différentes variations de l'environnement de l'internet, que ce soient les variations de la nature du trafic délivré par les applications, ou les variations de l'infrastructure qui achemine les données de l'expéditeur vers le destinataire. Mais ne pouvant agir que de bout-en-bout, les capacités de TCP à contrôler la congestion restent limitées. De plus, elles restent sous l'influence des machines d'extrémité, ce qui peut provoquer une iniquité dans l'utilisation du réseau entre les utilisateurs, puisque TCP établit une équité entre sessions et non entre utilisateurs. C'est à l'opérateur de mettre en place une infrastructure de traitement de la congestion dans son réseau et, grâce à une connaissance plus fine du trafic et de l'état du réseau, d'implanter des mécanismes plus efficaces et plus équitables que ceux utilisés par les machines d'extrémité qui opèrent de bout-en-bout. Ces nouveaux mécanismes peuvent être strictement sous le contrôle de l'opérateur comme ils peuvent agir conjointement avec les utilisateurs dans un système plus collaboratif.

Un réseau d'opérateur à même de traiter la congestion pourrait être représenté comme

sur la figure 1.4. Tout système de traitement de la congestion peut être décomposé en quatre opérations fondamentales : la détection, la notification, la décision, et la réaction. La détection consiste à mettre en place un mécanisme pour collecter des informations pertinentes sur l'état du réseau permettant d'en déduire d'une façon plus ou moins fiable l'état de la congestion. La notification consiste à informer l'organe de décision des indicateurs de congestion collectés. La décision consiste à analyser ces informations, et à décider s'il y a un risque de congestion. Enfin, si le risque de congestion est détecté, la réaction agit sur le trafic pour éviter la congestion. Les quatre opérations peuvent être mises en œuvre en différents endroits du réseau et peuvent aussi faire intervenir les machines d'extrémité.

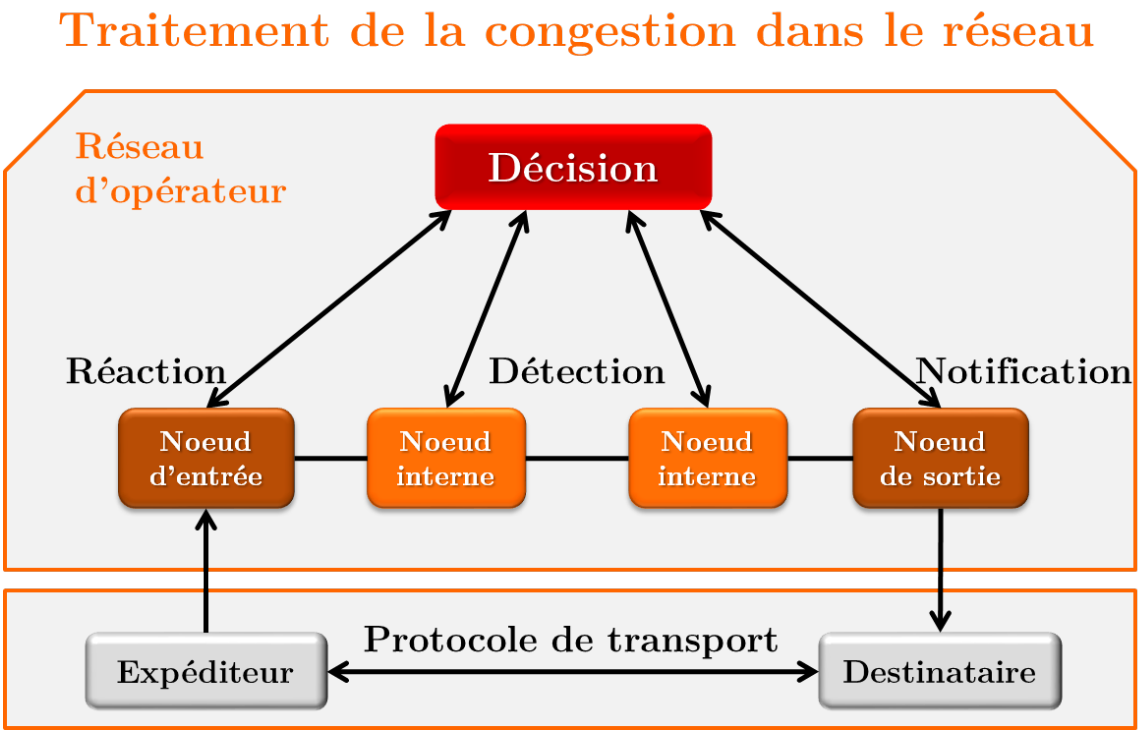

FIGURE 1.4 – Traitement de la congestion dans le réseau

Prenons l'exemple du contrôle de congestion de TCP (fondé sur les pertes). Celui-ci est uniquement de bout-en-bout, donc nécessairement, les quatre opérations se trouvent toutes du côté des utilisateurs. Lorsqu'il y a congestion, on perd des segments, alors :

— La fonction de détection de la congestion de TCP consiste à suivre les numéros de séquence des segments reçus au niveau du destinataire.

— La fonction de notification, qui est également au niveau du destinataire, se charge

d'envoyer à l'expéditeur des accusés de réception qui ne rendent compte que du dernier segment reçu en séquence.

— La fonction de décision se trouve donc au niveau de l'expéditeur. Elle est implantée grâce à l'algorithme de fast retransmit : la réception de trois accusés de réception dupliqués est le signal qu'il y a congestion dans le réseau et qu'il faut agir pour contrôler cette congestion.

— La fonction de réaction est aussi au niveau de l'expéditeur, et elle agit par la réduction de la taille de la fenêtre de congestion et par l'algorithme de congestion avoidance.

Le réseau n'intervient dans aucune de ces fonctions car rien n'est mis en place pour informer du risque de congestion, et TCP ne fait que s'adapter aux variations du trafic grâce aux quelques informations qu'il récupère au travers de ses mécanismes. Le rôle de l'opérateur, comment on l'a dit, devrait être de se charger d'implanter au sein du réseau ces fonctions. Parmi celles-ci, la fonction de décision est particulièrement importante car c'est elle qui porte toute l'intelligence. Celui qui contrôle cette fonction contrôle l'ensemble de la solution de gestion du trafic. Pour un opérateur de réseau, il est important de localiser cette fonction dans son réseau afin de maîtriser l'utilisation de ses ressources.

Finalement, qu'elles soient entièrement ou partiellement contrôlées par lui, qu'elles soient centralisées ou décentralisées, il y a plusieurs manières d'implanter chacune de ces fonctions. Nous allons nous intéresser à chacune d'elles en détail et montrer comment il est possible de les mettre en place et de les associer pour construire une architecture globale de traitement de la congestion.

1.3.3 La détection

Statistiques réseau

Les informations les plus triviales sur l'état du réseau et sur une éventuelle congestion nous viennent directement des statistiques que l'on peut collecter sur le trafic transitant par les équipements du réseau. En entrée de réseau, on peut par exemple compter les paquets ou les octets qui nous viennent de l'interface d'échange avec un pair, ou ceux qui viennent d'un client de l'opérateur identifié par son adresse IP et d'autres informations relatives à sa souscription. Dans le cœur du réseau, on peut enregistrer les statistiques du trafic qui transite sur chaque interface du réseau et sur les files d'attente qui sont implantées sur ces interfaces pour en connaître le degré d'utilisation (octets transmits, octets reçus, octets perdus). Dans la pratique, les données sont généralement collectées sur des intervalles de

temps de plusieurs minutes ce qui ne permet pas une réaction rapide à l'apparition de la congestion.

Mesures de bout-en-bout

Les statistiques permettent certes d'avoir une idée sur les caractéristiques du trafic en un point du réseau, mais elles ne rendent pas compte de la qualité de service ressentie par un flux traversant tout le réseau ou une portion de celui-ci. Pour cela, il faudrait faire des mesures de bout-en-bout, effectuées entre deux points du réseau. Il y a deux méthodes pour avoir ces mesures, la méthode passive et la méthode active [16]. Les mesures passives consistent à analyser le trafic réel entrant par un point du réseau et de l'analyser de nouveau en un autre point du réseau pour récupérer les statistiques concernant le débit, les pertes, et le délai. Cette méthode permet d'avoir des informations sur les conditions réelles d'un flux, mais elle peut être lourde à mettre en œuvre car elle nécessite des équipements dédiés qui identifient et analysent à la volée le trafic en corrélant les informations en entrée avec les informations en sortie. Elles doivent en conséquence être réservées à des trafics spécifiques : services entreprise, signalisation, etc. Les mesures actives sont moins lourdes car elles consistent en l'analyse d'un flux introduit dans le réseau spécifiquement pour sonder celui-ci. En revanche, si notre but est de détecter la congestion dans le réseau, l'introduction d'un trafic supplémentaire risque d'aggraver la situation et de fausser les mesures.

Seuil et marquage

Un mécanisme plus adéquat pour rendre compte immédiatement de la congestion consiste à implanter un seuil sur les files d'attente à partir duquel une alerte est donnée sur le risque de congestion. Les mécanismes à seuil seront très largement utilisés dans nos travaux, nous allons les décrire avec un certain niveau de détail. Voici leur principe : au lieu d'implanter une simple file First In, First Out (FIFO) qui perd des paquets lorsqu'elle déborde, il serait plus judicieux d'anticiper le débordement de cette file et de commencer à détruire les paquets avec de plus en plus d'intensité selon l'occupation de la file d'attente, comme le fait une file Random Early Detection (RED). Cette dernière est une technique d'Active Queue Management (AQM), proposée par [17], qui permet de détruire aléatoirement des paquets selon une probabilité qui augmente de 0 à une probabilité maximale p_{max} quand la taille moyenne de la file augmente d'un seuil minimal mTh à un seuil maximal MTh (cf. figure 1.5). Au-delà du seuil maximal, tout le trafic est détruit. Cette méthode

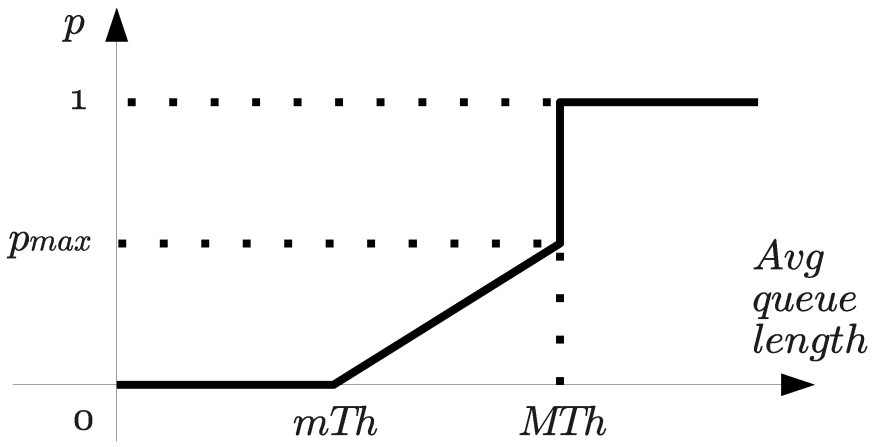

FIGURE 1.5 – Fonction RED

permet d'alerter les machines d'extrémité sur le risque de congestion en envoyant un signal préalable au débordement de la file qui pourrait mener à la congestion et dégrader la qualité de service de l'utilisateur. Elle est destinée aux sources dites réactives capables de réagir à la perte de paquets, typiquement TCP. La perte de paquets entraînée par ce genre de techniques a pour but d'éviter une dégradation supplémentaire de la qualité de service.

La file RED apporte une manière implicite de détecter le risque de congestion et de le signaler aux machines d'extrémité puisqu'elle repose sur la génération d'une information (perte dans les numéros de séquence) attendue par les mécanismes de TCP pour traiter la congestion. Une manière plus explicite d'alerter du risque de congestion lors du dépassement d'un seuil dans la file d'attente serait de marquer les paquets plutôt que de les détruire. ECN est un mécanisme permettant un tel marquage. Standardisé par l'IETF dans [18], il utilise les deux bits ECN dans l'en-tête IP (cf. figure 1.6) pour signaler explicitement au destinataire un risque de congestion dans le réseau, à charge pour celui-ci d'informer la source. Dans le cas de TCP par exemple, le destinataire reçoit du réseau une marque Congestion Experienced (CE) et à son tour informe explicitement l'expéditeur en envoyant un signal ECN-Echo (ECE) sur l'en-tête TCP des accusés de réception. À la réception d'un accusé portant cette mention, l'expéditeur réagit comme lors de la détection d'une perte, c'est-à-dire en réduisant la fenêtre de congestion, sans avoir à réémettre de paquet.

Pour faire face aux problèmes rencontrés par la première version de la file RED, notamment la discontinuité dans la destruction ou marquage des paquets après le dépassement du seuil maximal, et des problèmes de configuration de la file pour plusieurs types de trafic,

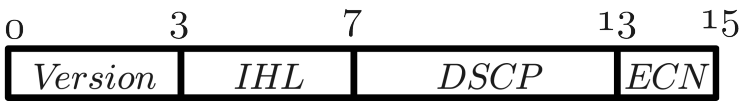

FIGURE 1.6 – Octets 1 et 2 de l'en-tête IPv4

plusieurs variantes de RED ont été proposées. La variante Gentle Random Early Detection (GRED) [19] permet une augmentation plus lisse de la probabilité de destruction ou de marquage quand la taille moyenne de la file dépasse le seuil maximal MTh en augmentant la probabilité linéairement jusqu'à $2 \times MTh$. La variante Adaptive Random Early Detection (ARED) [20] quant à elle permet d'adapter dynamiquement la configuration de la file selon la charge de trafic. Ici, la probabilité de marquage maximale p_{max} (donc l'agressivité de la file ARED) s'adapte en adoptant une lente variation en AIMD qui a pour objectif d'obtenir une stabilité dans l'oscillation de la taille moyenne de la file.

Mais RED et ses variantes ne sont pas les seuls mécanismes d'AQM qui ont été proposés. Alors que RED est utilisé entre autres pour notifier les sources de la proche saturation des files qui risquent de déborder et de dégrader leurs performances, un autre phénomène a été de plus en plus rencontré sur les files d'attente des routeurs dans l'internet qui mène également à une dégradation des performances. Le phénomène de bufferbloat survient lorsque des files d'attente à grande capacité dans les équipements du réseau se remplissent et introduisent un très grand délai pour le trafic qui passe par ces équipements, très préjudiciable surtout pour les applications très sensibles au délai. Parmi les solutions au bufferbloat, Controlled Delay (CoDel) [21] et Proportional Integral controller Enhanced (PIE) [22] ont été proposées à l'IETF. Le premier suggère de contrôler le délai dans une file d'attente en détruisant les paquets lorsque le délai d'attente minimum des paquets dans la file dépasse un délai cible, alors que le second s'inspire des systèmes de contrôle à action proportionnelle et intégrale pour arriver au même but. Le bufferbloat est un problème que peut rencontrer le trafic des utilisateurs dans le réseau et auquel l'opérateur devrait répondre, néanmoins, comme les variations de délai ne sont pas dans le cœur de notre problématique, nous ne nous attarderons pas sur cette famille de solutions.

Dans RED comme dans ECN, les machines d'extrémité restent responsables de la majorité des opérations de traitement de la congestion. Ici néanmoins, l'opérateur contribue à la détection de la congestion dans son réseau en implantant des mécanismes de seuil qui permettent d'anticiper la saturation des capacités de ses équipements. Plus globalement, l'utilisation d'un seuil sur les files d'attente, et encore plus avec le marquage du trafic excédantaire, est une solution simple et peu onéreuse pour rendre compte directement du

risque de congestion dans le réseau.

Le même système de seuil peut être utilisé cette fois-ci sur les liens au lieu des files d'attente, c'est-à-dire en fixant un seuil correspondant à un pourcentage de la bande passante d'un lien, associé par exemple à une classe de service DiffServ, et en marquant le trafic excédant ce seuil. Le trafic marqué mesurera de ce fait la pré-congestion dans le réseau. C'est ce principe de détection qui a été standardisé par l'IETF sous le nom de PCN [6]. Comme pour le mécanisme de contrôle de congestion de Comcast [10], PCN définit deux seuils de marquage sur les liens appartenant à un domaine PCN. On réutilise le champ ECN de l'en-tête IP pour marquer les paquets en threshold-marked (ThM) lorsque le seuil PCN-threshold-rate est dépassé, et les marquer en excess-traffic-marked (ETM) lorsque le seuil PCN-excess-rate est dépassé [23], le second étant plus élevé que le premier. Ces marquages peuvent ensuite être remontés à l'organe de décision qui agira pour traiter la pré-congestion.

1.3.4 La notification

La manière de notifier de l'état du réseau et des informations sur la congestion dépend beaucoup de qui sera en charge de prendre la décision d'agir pour traiter cette congestion et où il se trouve au niveau du réseau. Sont-ce les machines d'extrémité ? Est-ce l'opérateur ? Comment informer l'un ou l'autre de la congestion ?

D'abord, quel que soit l'organe de décision, quelles informations pertinentes devraient être transmises à ce dernier ? Ce qui serait le plus intéressant, en plus d'avoir l'indication qu'il y a une congestion, ce serait de connaître également sa localisation dans le réseau, son intensité et la source ou les sources du trafic en congestion, ce qui permettrait à l'organe de décision d'agir en ciblant les points de congestion, d'agir proportionnellement au niveau de congestion et de cibler directement la cause de la congestion. Mais, avoir les quatre informations sur la congestion (présence, localisation, intensité, source) en même temps tout en implantant des mécanismes simples n'est pas toujours évident.

Du réseau vers la source

Avec ECN dans TCP par exemple, l'organe de décision est l'expéditeur du trafic. La présence de la congestion est notifiée d'une part par l'opérateur en marquant le flux qui transporte l'information au destinataire, et d'autre part par le destinataire à travers les accusés de réception. L'organe de décision connaît de ce fait la source du trafic impliquée dans la congestion (lui-même), mais il ne connaît pas la localisation de la congestion à part

globalement sur le chemin vers le destinataire, ce qui n'est pas pénalisant ici car il n'est pas capable d'agir directement sur le réseau. Il ne connaît pas non plus son intensité avec précision à cause du fonctionnement même d'ECN dans TCP. En effet, même si plusieurs paquets sont marqués CE sur la connexion TCP, il ne peut y avoir qu'une seule notification ECE par RTT, et l'expéditeur ne devra réduire sa fenêtre de congestion qu'une seule fois par RTT, et la réduire de moitié au lieu de le faire proportionnellement au nombre de paquets marqués. Des améliorations de TCP ont été proposées pour pallier ce problème, avec [24] qui permet au destinataire de rendre compte avec précision des paquets marqués dans le réseau, et avec [25] qui permet à l'expéditeur de réduire la fenêtre de congestion proportionnellement au nombre de paquets marqués.

La participation de l'opérateur à la notification de la congestion reste ici assez minime puisque c'est le destinataire qui est en contact direct avec l'expéditeur. Mais l'opérateur peut lui-même notifier le marquage dans le réseau à l'organe de décision. Au lieu de le signaler sur les flux, il peut implanter au niveau des équipements un mécanisme qui contacte directement la source du trafic lorsqu'il y a congestion. Ceci est réalisable par exemple grâce à des solutions comme Backward Congestion Notification (BCN) [26] ou Quantized Congestion Notification (QCN) [27] [28].

Du réseau vers l'opérateur

Jusqu'ici, ce sont les machines d'extrémité qui implantaient la fonction de décision, et l'opérateur avait soit la possibilité d'indiquer le risque de congestion en marquant les flux des machines, soit en contactant directement la source de la congestion. Toutefois, les machines d'extrémité ne sont pas toujours sous le contrôle de l'opérateur et celui-ci ne peut garantir qu'elles réagiront à la notification de congestion qu'il leur envoie. Dans un souci de protection de son réseau et de l'ensemble de ses utilisateurs, l'opérateur devrait donc implanter lui-même l'organe de décision. Mais dans ce cas-là, d'autres méthodes sont à envisager pour notifier des informations de congestion.

L'opérateur peut collecter à distance les statistiques de ses équipements pour en connaître l'utilisation en bande passante et en file d'attente, mais aussi pour récupérer lui-même les informations de marquage sur les interfaces où un mécanisme de détection a été implanté. Le protocole le plus connu pour rassembler les informations de gestion d'un équipement est Simple Network Management Protocol (SNMP) [29]. Il repose sur une base de données implantée sur les équipements et appelée Management Information Base (MIB) [30] pour collecter les statistiques sur les objets réseaux répertoriés. Plus récemment, l'IETF a standardisé Network Configuration Protocol (NETCONF) [31] ainsi qu'un langage de

modélisation de données YANG [32] pour gérer entre autres les statistiques récupérées au travers de ce protocole.

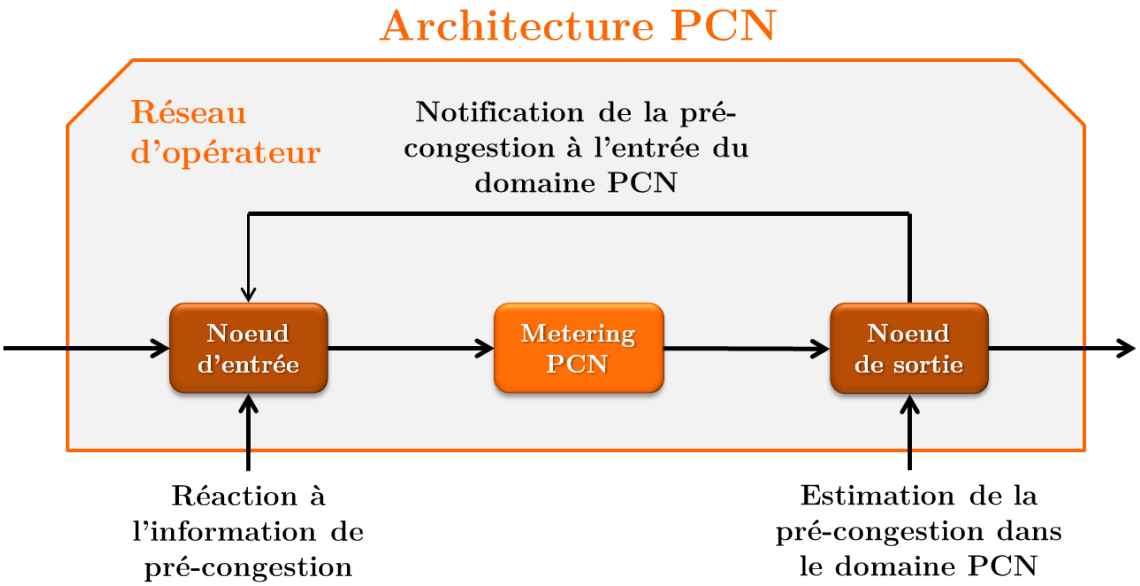

FIGURE 1.7 – Architecture PCN

Notifier des informations de marquage pour chaque interface permet de connaître sa présence, sa localisation, et son intensité locale, mais pas son intensité globale sur le chemin des flux dont la qualité de service est à maintenir. Ce serait possible à condition de garder sur chaque équipement l'état de ces flux et de récupérer sur le chemin de chaque flux les statistiques de marquage de celui-ci, mais cela serait fastidieux et pratiquement inenvisageable dans l'architecture actuellement décentralisée du réseau d'opérateur. Une solution pour avoir une vue globale de la congestion sur les flux surveillés est de récupérer les informations de marquage en sortie de réseau. Cette méthode permet de connaître l'intensité de la congestion très simplement car il suffit de compter la proportion de paquets marqués par rapport à tout le trafic du flux pour en avoir l'estimation. Notifier de cette information à l'organe de décision est donc beaucoup plus léger puisqu'elle ne provient que des seuls équipements en sortie de réseau. C'est le cas de PCN qui notifie des proportions de paquets marqués ThM et ETM à l'organe de décision de l'opérateur qui se trouve dans le cadre de l'architecture PCN au niveau des entrées du réseau (cf. figure 1.7).

L'architecture PCN reste une architecture décentralisée où les décisions sont prises indépendamment les unes des autres. Elles ne prennent donc pas en compte la globalité de

l'état du réseau et les différentes influences que peuvent avoir telle réaction dans un endroit donné sur les autres fonctions de décision distantes.

Avec une architecture SDN

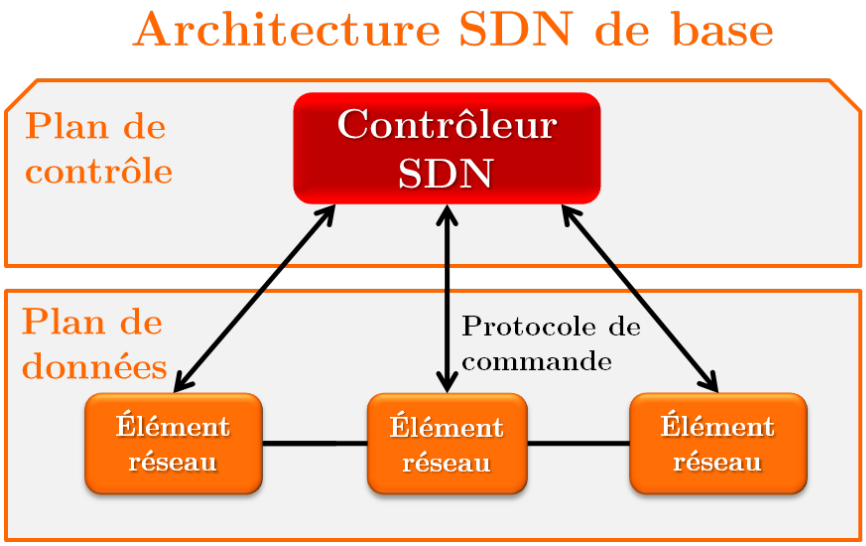

FIGURE 1.8 – Architecture SDN de base

Dans une architecture où les décisions sur le comportement du réseau sont centralisées au lieu d'être réparties en plusieurs points, la récupération des informations sur la congestion serait beaucoup plus directe. Les équipements seraient tous sous le contrôle d'une même entité qui aurait ainsi la vision globale sur le réseau et sur les flux y transitant. C'est ce qui est proposé dans les architectures SDN [2]. Ici, le plan de données qui transfert les paquets dans le réseau et le plan de contrôle où « l'intelligence » dans le traitement des paquets réside sont séparés. Le plan de contrôle peut être centralisé dans un contrôleur de réseau SDN, et la communication entre le contrôleur et les éléments du réseau se fait au travers d'un protocole de commande (cf. figure 1.8). Ce protocole peut collecter les statistiques de marquage des paquets au contrôleur, qui agit dans ce cas-là comme organe de décision dans l'architecture de traitement de la congestion, que ce soit via un polling régulier des équipements par le contrôleur, ou via des notifications déclenchées par alarme sur la détection de congestion par exemple.

1.3.5 La décision

Que l'organe de décision se trouve du côté des machines d'extrémité, donc hors du réseau, ou dans le réseau maîtrisé par l'opérateur, le déclenchement de la décision de traiter la congestion se fait à l'examen des informations notifiées ou inférées du réseau et une fois ces informations jugées assez alarmantes pour devoir agir afin de prévenir une dégradation de la qualité de service.

La décision peut être très rapide comme pour TCP qui décide de réduire son débit dès la réception de la première information de marquage ECN. Mais le risque avec un déclenchement trop rapide de la décision, après juste une information ponctuelle sur la congestion, est d'activer une réaction qui ne modifie pas la qualité de service de l'utilisateur voire qui le limite inutilement et mène à une sous-exploitation des ressources du réseau, ce qui n'est pas non plus dans l'intérêt de l'opérateur.

Une décision trop tardive est également superflue, puisque la réaction pourrait venir alors que la congestion est déjà passée. Ou alors, bien que la réaction tardive vînt tout de même résoudre la congestion encore présente, cette dernière aurait déjà dégradé suffisamment la qualité de service pour nuire à l'expérience du client de l'opérateur.

Plusieurs possibilités s'offrent à l'organe de décision pour se prononcer sur la congestion au moment opportun. Les informations qu'il reçoit de la part du réseau sur le marquage peuvent servir de base à un calcul de moyenne pondérée sur un intervalle de temps pour avoir une vision plus lissée de ce qui se passe dans le réseau. C'est sur la valeur de la moyenne que sera prise la décision de réagir ou non à la congestion.

La décision qui est prise par les machines d'extrémité est plutôt une décision de niveau flux et généralement rapide à effectuer, alors qu'une décision de niveau réseau, c.-à-d. par l'opérateur, est plutôt une décision par utilisateur ou par agrégat de flux, et souvent moins réactive sur les informations d'utilisation du réseau.

Service Level Agreement

Dans le cas où le client a signé un contrat avec son opérateur appelé Service Level Agreement (SLA), ses statistiques peuvent être surveillées pour réagir quand la qualité de service risque de ne plus satisfaire le contrat comme par exemple lorsque les pertes dépassent le pourcentage précisé dans celui-ci, ou si la latence devient trop importante. En pratique, cette approche complexe est réservée à des services spécifiques, typiquement aux services à destination des entreprises.

Volume de données

Une autre possibilité est que les données d'utilisation du réseau par le client puissent être comparées à ce qui a été précisé dans son contrat de souscription avec l'opérateur pour réagir au bout d'une certaine limite. C'est ce qui est fait actuellement par exemple dans les réseaux mobiles où le client souscrit à une certaine quantité de données à consommer au-delà de laquelle il peut payer le surplus, voire son débit limité, ou bien encore ne plus avoir accès au réseau [33]. Le but est de prévenir la congestion et de garantir l'équité entre clients puisqu'en incitant l'utilisateur à modifier son comportement, il peut lisser lui-même son utilisation du réseau pour ne pas consommer son volume de données limité. Mais prendre une décision uniquement sur la quantité de données ne préviendra pas forcément la congestion dans le réseau puisqu'elle ne prend pas en compte les informations sur l'état du réseau. Un utilisateur peut très bien consommer son volume de données alors que le réseau est soumis à une grosse charge de trafic, par exemple pendant les heures chargées du soir entre 19 heures et 22 heures [34], tandis que consommer son volume de données pendant les heures non chargées n'est ni nécessaire ni utile car il n'affecte en rien l'état du réseau. De plus, la limitation en volume de données est une pratique très impopulaire chez les clients de l'opérateur [35]. Elle est également combattue par les autorités de régulation comme la Federal Communications Commission (FCC) [36] aux États-Unis qui reçoit plusieurs plaintes des clients des opérateurs américains.

Volume de congestion

Une stratégie de décision similaire est possible mais cette fois-ci en limitant les utilisateurs seulement selon leur impact sur le réseau. Lorsque celui-ci est non chargé, aucune congestion n'est détectée, aucune information sur celle-ci n'est engendrée, les utilisateurs peuvent alors utiliser le réseau comme il leur sied, sans limitation sur un volume de données. Mais quand le réseau est encombré, des signaux sur la congestion (comme le marquage) sont générés pour le trafic des utilisateurs, idéalement proportionnellement au volume de congestion généré, et c'est selon la quantité de ces informations qu'on les limite. On ne déclenche alors une décision de traitement de la congestion que pour les utilisateurs ayant un impact important sur le réseau.

Pour cela, il faudrait obtenir les informations sur la contribution de chaque utilisateur à la congestion du réseau. Dans une approche collaborative, l'IETF a proposé un mécanisme, appelé ConEx [7], où l'utilisateur lui-même notifie à l'opérateur le marquage qu'il a subi sur le chemin des données. Par exemple, l'information sur le marquage ECN que

l'expéditeur TCP reçoit du destinataire à travers les accusés de réception est réinjecté dans le réseau par l'expéditeur à travers des messages appelés Re-Echo [37] (cf. figure 1.9). À chaque utilisateur est attribué un volume de congestion (et non un volume de données) qui est consommé par les messages Re-Echo et au-delà duquel la décision de réagir est prise par l'opérateur. Avec cette approche, l'utilisateur est incité à ne pas encombrer le réseau pendant les périodes chargées de la journée en reportant ses téléchargements les moins importants, comme les mises à jour des systèmes par exemple, aux heures où le réseau est le moins chargé. À l'inverse, quand le réseau n'est pas chargé, l'utilisateur peut envoyer sans limitation du trafic.

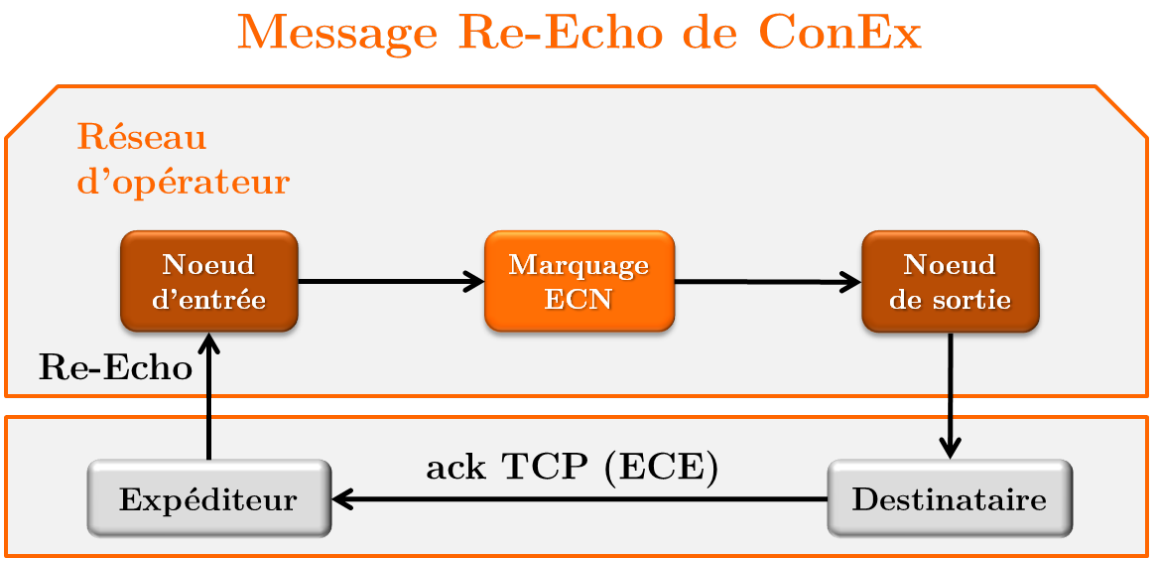

FIGURE 1.9 – Message Re-Echo de ConEx

Les mêmes effets peuvent être atteints au niveau du réseau de l'opérateur sans l'intervention de l'utilisateur avec une architecture centralisée telle que le SDN. Sans centralisation, ConEx notifie des informations de marquage (l'impact de l'utilisateur sur l'utilisateur) directement en entrée de réseau où se situe l'organe de décision (valeur du volume de congestion) de l'opérateur et où la réaction a lieu. Avec un contrôleur SDN centralisé, l'opérateur peut récupérer le marquage spécifique à un utilisateur directement en sortie de réseau à travers le protocole de commande et le comparer au volume de congestion qui lui est attribué, information située également au niveau du contrôleur, pour prendre la décision d'agir lorsque le volume de congestion est consommé.

1.3.6 La réaction

Quand la décision est prise de réagir vis-à-vis de la congestion détectée dans le réseau, la réaction peut être plus ou moins efficace selon les informations notifiées à l'organe de décision. La réaction la plus efficace serait de n'impacter que les flux de trafic qui transitent par le point de congestion et de réagir proportionnellement à la congestion détectée, mais cela supposerait que l'on ait la localisation de celle-ci, son intensité, et que l'on puisse identifier la source des flux subissant la congestion. Comme nous l'avons vu dans la section 1.3.4, selon les mécanismes déployés dans le réseau pour le traitement de la congestion, nous pouvons avoir une partie ou la totalité de ces informations.

Priorisation

Par ailleurs, les réactions peuvent être de plusieurs natures. Une réaction possible serait de prioriser le trafic qui a besoin de la plus grande qualité de service par rapport au trafic qui peut se suffire d'un traitement de moindre qualité par exemple. Le champ Differentiated Services Code Point (DSCP) de l'en-tête IP (cf. figure 1.6) permet de définir plusieurs classes de service qui pourraient avoir dans le réseau des traitements différentiés. Cette solution nécessite néanmoins que chaque équipement dans le réseau soit conscient du marquage et du traitement prioritaire à effectuer selon ce marquage. Il serait en revanche possible de le faire au préalable, et de manière statique sur des flux agrégés, et des flux identifiés comme prioritaires comme la voix sur IP (Voice over IP (VoIP)) par exemple.

Scheduling

Les systèmes de priorisation se conjuguent souvent avec des algorithmes d'ordonnancement au niveau des files d'attente pour servir les flux les plus importants en priorité et en gardant une forme d'équité entre tous les flux à servir. Le Class-Based Weighted Fair Queueing (CBWFQ) par exemple [38] s'organise en attribuant un poids à chaque classe de service configurée et en servant cette classe afin qu'elle ait une part de la bande passante proportionnelle à ce poids.

Traffic Engineering

L'autre solution pour l'opérateur afin de traiter la congestion dans le réseau est de pouvoir rerouter le trafic pour mieux répartir la charge autour du point de congestion. Cette opération est d'autant plus réalisable quand l'opérateur est en mesure d'établir des

tunnels maîtrisés pour les flux qui entrent dans son réseau, par exemple avec une solution telle que MultiProtocol Label Switching-Traffic Engineering (MPLS-TE) [39].

MPLS-TE propose d'utiliser les possibilités offertes par l'ingénierie de trafic, qui consiste à optimiser l'utilisation des ressources disponibles dans le réseau pour éviter la congestion, avec la flexibilité et la simplicité d'utilisation de la commutation de labels MultiProtocol Label Switching (MPLS) afin de transporter les flux de trafic à travers un réseau dans des Label Switched Paths (LSPs) tout en respectant des contraintes de qualité de service. Les ressources disponibles dans un réseau sont échangées à travers des extensions au protocole de routage interne, tel que Open Shortest Path First (OSPF) ou Intermediate System to Intermediate System (IS-IS), nécessaires pour annoncer les paramètres Traffic Engineering (TE) des liens (bande passante, ...). MPLS-TE utilise ensuite un algorithme de routage par contrainte afin de trouver, pour un flux donné, le chemin le plus court dans la topologie du réseau, et qui en même temps satisfait ses exigences en qualité de service (débit, délai). Le chemin calculé est finalement établi (distribution de labels et réservation de ressources), par Resource Reservation Protocol-Traffic Engineering (RSVP-TE), l'extension de RSVP qui permet le routage explicite avec réservation de bande passante et la distribution de labels.

Grâce à cette technique, il est aussi possible de réaliser un contrôle d'admission des flux qui empêche de nouveaux flux d'entrer dans le réseau si l'on estime que leur admission contribuerait à la congestion. Dans le cas où l'on aurait besoin d'admettre un tunnel avec une haute priorité dans un réseau déjà sous forte charge, la technique de préemption de tunnels pourrait écarter les tunnels avec une priorité plus basse pour établir les tunnels à haute priorité.

Mais la réservation de bande passante ici n'est malheureusement que logique et non physique, et il n'y pas de réelle garantie de bande passante pour le tunnel établi. De plus, les variations en débit d'un flux au cours du temps en rendent la réservation vraiment utile et efficace que lorsqu'il s'agit d'un tunnel comportant de nombreux flux dont l'agrégation mène à un trafic dont on connaît plus ou moins les caractéristiques [39]. Actuellement, dans les réseaux, cette solution est utilisée principalement pour établir un chemin principal pour une telle agrégation de trafic ainsi qu'un chemin de protection en cas de dysfonctionnement dans le réseau. Mais son utilisation pour un traitement dynamique de la congestion se heurterait surtout à une importante barrière posée par l'ingénierie de trafic : sa complexité.

Traffic shaping et traffic policing

Enfin, pour réagir à la congestion dans le réseau, une dernière solution consisterait à limiter le trafic en congestion dans le réseau. Soit ce sont les machines d'extrémité qui se chargent de réduire leur débit lors de la détection de congestion comme avec les algorithmes de contrôle de congestion de TCP, soit c'est l'opérateur qui réagit en limitant les utilisateurs ou les flux, idéalement ceux qui passent par les points de congestion si les informations pour faire cela sont disponibles. Deux approches de limitation de débit sont possibles : le traffic shaping ou le traffic policing. Les deux approches consistent à mesurer le trafic entrant pour voir s'il est conforme au débit imposé par l'opérateur. En revanche, leur réaction face au trafic non conforme est différente. Le traffic shaping retarde ce dernier en plaçant les paquets dans des files d'attente et en les libérant à un rythme qui respecte les caractéristiques de trafic imposées. Le traffic policing quant à lui détruit le trafic non conforme ou le marque avec une priorité plus basse si des mécanismes de priorisations ont été implantés dans le réseau. Les deux approches peuvent être mises en œuvre grâce à des algorithmes tels que le leaky bucket ou le token bucket [40].

La limitation de débit est une méthode utile, peut-être la plus radicale, pour réagir à la congestion car elle limite ce qui est finalement la cause de la congestion : un trafic trop important par rapport aux capacités du réseau. Mais elle ne serait réellement efficace que si elle est proportionnelle à la congestion occasionnée, et pour cela il faudrait connaître l'intensité de la congestion, ce qui dépend des mécanismes de notification déployés. Quand le policing ou le shaping sont appliqués proactivement avec une sévérité excessive, on se retrouve avec un réseau sous-utilisé.

L'implantation de ce type de réaction en entrée de réseau est meilleure que les réactions locales comme la priorisation ou le scheduling car contrairement à celles-ci le trafic n'est pas encore dans le réseau. En ciblant la réaction sur un nombre limité de flux, on réduit l'impact sur le trafic déjà présent dans le réseau et on augmente la probabilité de pouvoir réduire la congestion.

Récapitulatif

Le tableau 1.1 récapitule les opérations de traitement de congestion et les méthodes pour réaliser chacune d'elles.

Opération	Méthodes de réalisation
Détection	Fast retransmit TCP, statistiques réseau, mesures de bout-en-bout, par sonde, définition de seuil, marquage
Notification	accusés de réception TCP, notification réseau vers utilisateur (BCN, QCN), protocole de gestion (SNMP, NETCONF), polling ou notification SDN.
Décision	Notification instantanée (TCP ECN), moyenne des compteurs, SLA, volume de données, volume de congestion
Réaction	Priorisation, reroutage, contrôle d'admission, préemption de tunnels, traffic engineering, congestion avoidance TCP, traffic shaping, traffic policing

TABLE 1.1 – Opérations de traitement de la congestion et les méthodes de réalisation

1.4 Conclusion sur l'état de l'art

Comme nous l'avons vu dans ce chapitre, traiter la congestion passe par plusieurs étapes qui demandent chacune plus ou moins d'efforts et d'investissements de la part de l'opérateur. Plusieurs techniques et plusieurs mécanismes ont été proposés pour détecter le risque de congestion, notifier cette information puis décider de réagir au moment opportun face à ce risque. Ils permettent une certaine précision dans le traitement de la congestion selon ce qu'ils dévoilent de l'état du réseau. Quelques mécanismes demandent l'intervention des utilisateurs et d'autres sont totalement sous le contrôle de l'opérateur. L'essentiel pour celui-ci est d'être impliqué dans le traitement de la congestion dans son réseau au lieu de laisser les machines d'extrémité l'utiliser avec le peu de connaissances qu'ils peuvent avoir sur lui. Que ce soit en apportant une partie des solutions dans le traitement de la congestion ou en étant entièrement maître de son réseau, l'opérateur est le plus qualifié pour assurer la meilleure utilisation et la meilleure qualité de service dans son réseau.

Dans les mécanismes standardisés par l'IETF, ConEx est une solution collaborative qui permet d'utiliser l'information de congestion notifiée par chaque utilisateur pour ne policer que les utilisateurs générateurs de congestion. Mais une analyse plus profonde de ce mécanisme n'a pas été réalisée, encore moins dans un contexte opérateur où ce dernier doit implanter un certain nombre d'éléments importants pour le fonctionnement du mécanisme. Nous proposons donc une évaluation des performances de ConEx pour avoir une meilleure vue des possibilités qu'il peut offrir à l'opérateur ainsi que des étapes nécessaires pour un éventuel déploiement dans le réseau.

Dans un environnement plus contrôlé comme celui des clouds où l'utilisateur n'est pas

ou peu impliqué dans le traitement de la congestion, nous verrons comment nous pouvons utiliser l'information de congestion ou de pré-congestion pour offrir une meilleure qualité de service dans le cloud.

Nous proposerons dans l'architecture réseau particulière de l'intra-DC une solution qui permet au contrôleur de cloud de déployer les mécanismes de détection et de notification de la congestion qui apparaît entre les machines virtuelles des clients de l'opérateur, ainsi que de prendre la décision selon l'état du réseau et de réagir avec un policing des machines les plus génératrices de congestion. Nous proposerons également dans l'environnement inter-DC une architecture d'orchestration, nécessaire pour avoir une vision globale de tous les éléments qui entrent en jeu dans cet environnement, afin de réaliser la connectivité entre les différents sites du client tout en implantant les mécanismes de traitement de la pré-congestion dans un réseau d'opérateur géré par un contrôleur SDN.

CHAPITRE 2

ÉTUDE DU MÉCANISME CONEX

Dans le réseau d'Orange, comme plus largement au niveau de l'internet, il existe une disparité entre deux catégories d'utilisateurs. En effet, durant les heures chargées de la journée, une grande partie du trafic généré provient d'un pourcentage réduit de l'ensemble des utilisateurs que l'on appelle les « *heavy users* ». Dans une étude sur le trafic du réseau d'Orange par exemple, [34] montre que 80% du trafic descendant des réseaux d'accès Fiber To The Home (FTTH) est généré par 15% des utilisateurs. Or le risque de congestion induit par ces utilisateurs peut mener à une dégradation de la qualité de service de l'ensemble des utilisateurs, en particulier ceux dont le trafic est bien moins volumineux que nous appelons « *light users* », alors même que l'opérateur a pour objectif d'offrir une équité dans l'utilisation de son réseau. Dans ce chapitre, nous proposons une étude et une évaluation des performances de ConEx, un mécanisme standardisé par l'IETF qui permet à l'opérateur de limiter les utilisateurs de son réseau en fonction de la congestion qu'ils engendrent, donc sur leur impact sur les autres utilisateurs qui utilisent la même infrastructure.

Comme nous l'avons vu dans le chapitre précédent, il est possible de réagir au risque de congestion en limitant les utilisateurs en entrée de réseau avec un policer. Mais la limitation aveugle selon une quantité de données (mesurée en volume ou en débit) est peu efficace et inutilement pénalisante durant les périodes moins chargées quand les réseaux sont sous-utilisés, d'où l'idée d'une limitation selon la congestion induite par un utilisateur pour ne le limiter que lors des périodes chargées.

Dans la prochaine section, nous allons présenter ConEx qui permet la limitation des utilisateurs selon le niveau de congestion. Ensuite, notre objectif est d'étudier le compor-

tement de ConEx, en particulier sa capacité à améliorer l'équité entre utilisateurs, puis d'examiner son paramétrage pour trouver une plage de fonctionnement qui conduise à des résultats satisfaisants et suffisamment stables, et enfin d'enquêter sur l'impact des paramètres externes au mécanisme pour voir si ConEx peut raisonnablement être déployé dans un environnement réseau réel.

2.1 Congestion Exposure

TCP, qui est le protocole de transport le plus répandu sur internet, ne permet pas un partage des ressources équitable entre les utilisateurs, il ne permet qu'un partage équitable entre flux, ce qui signifie que plus un utilisateur déploie de flux, plus il récupère du débit et une part plus importante des ressources du réseau. Devant cette iniquité entre les *heavy users* et les *light users* qui est plus néfaste en période de congestion, et avec l'objectif des opérateur qui est d'assurer l'équité entre les utilisateurs de leur réseau, l'IETF a considéré qu'il était nécessaire de définir des mécanismes innovants pour pallier ce problème. La solution avait pour objectif de prendre en considération l'état de congestion du réseau pour pouvoir répartir les ressources équitablement entre les utilisateurs, tout en permettant une meilleure collaboration entre les opérateurs et les utilisateurs dans ce processus. Ainsi, ConEx a été défini [7] pour apporter les briques nécessaires afin d'atteindre les objectifs fixés.

Afin de limiter les utilisateurs selon leur niveau de congestion, il faut avoir la possibilité d'identifier les *heavy users* dont le trafic est générateur de congestion. L'opérateur applique pour cela un marquage sur les paquets des flux transitant par son réseau au niveau des files d'attente des équipements en congestion. Le marquage de congestion qui est appliqué sur les agrégats de flux est récupéré individuellement par les machines d'extrémité, ce qui permetterait d'identifier les *heavy users*. Mais les machines d'extrémité s'échangent les informations de congestion au niveau transport (p.ex. grâce aux accusés de réception de TCP) et ne sont pas disponibles dans la couche réseau, ce qui serait plus accessible pour l'opérateur. Avec la définition de ConEx, l'utilisateur peut exposer les informations de congestion qu'il reçoit au niveau transport dans l'en-tête des paquets réseau à destination de l'opérateur dans une approche collaborative. Cette quantité de données spécifique, générée en cas de congestion, est appelé « volume de congestion » d'un utilisateur, et c'est sur cette quantité que l'utilisateur sera évalué, selon les limites que l'opérateur aura posé.

Dans la suite de cette section, nous allons décrire le comportement du mécanisme ConEx, c.-à-d. comment il se propose d'exposer la congestion détectée dans le réseau.

2.1.1 Mécanisme ConEx

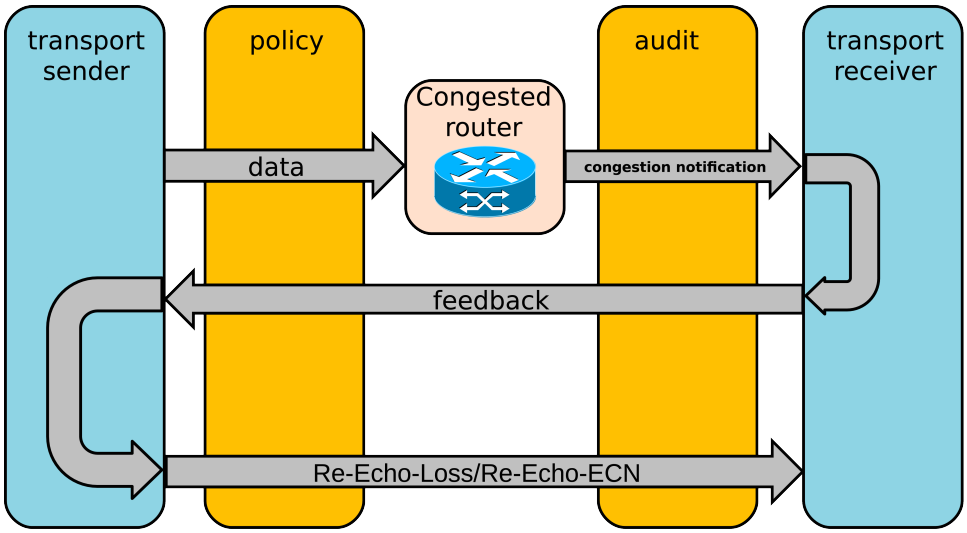

FIGURE 2.1 – Mécanisme ConEx

La figure 2.1 montre l'ensemble du processus d'exposition de la congestion de ConEx, ainsi que les éléments qui entrent en jeu dans son fonctionnement, ici dans le cas d'un trafic TCP qui est la cible principale du mécanisme. C'est pendant l'établissement de la connexion TCP que la capacité à utiliser ConEx est négociée entre les machines d'extrémité. Après que la connexion a été établie par le protocole de transport, un expéditeur TCP émet des paquets sur le réseau portant la mention ConEx-Capable. Sur le chemin d'un paquet, un ou plusieurs routeurs peuvent être en congestion. Le paquet sera soit perdu dans la file d'attente d'un routeur, soit marqué par celle-ci en ECN (en fixant la valeur CE sur les bits ECN de l'en-tête IP [18]). Cette information de perte ou de marquage sera détectée par le destinataire TCP, et ce dernier en informera l'expéditeur au travers des accusés de réception. La nouveauté introduite par ConEx est que l'expéditeur va pouvoir réinjecter l'information de congestion reçue dans l'en-tête des paquets IP pour envoyer le signal ConEx correspondant (Re-Echo-Loss en cas de détection de perte, Re-Echo-ECN en cas de marquage ECN, cf. section 2.1.2).

À l'entrée du réseau est implanté un policer de congestion qui va compter les signaux ConEx envoyés par l'expéditeur et qui va réagir selon la politique spécifiée par l'opérateur du réseau (p.ex. détruire des paquets ou abaisser leur priorité) si l'utilisateur a consommé tous les « droits à congestion » qui lui ont été attribués. À la sortie du réseau, un auditeur est utilisé pour fiabiliser le mécanisme en vérifiant que les expéditeurs ont exposé le bon

niveau de congestion. Cet élément vise à éviter que les sources du trafic ne minimisent la congestion que leurs flux ont rencontrée pour ne pas consommer leurs ressources et éviter la réaction du policer (cf. section 2.1.6). Si les sources sont de confiance, dans le cas où elles sont contrôlées par l'opérateur par exemple, l'auditeur n'est pas nécessaire. Étant donné qu'un audit fiable est une tâche complexe, l'absence d'auditeur simplifie le déploiement de ConEx.

2.1.2 Signaux ConEx

Plusieurs signaux ConEx inscrits dans l'en-tête IP des paquets provenant de l'utilisateur à l'intention du réseau ont été spécifiés par l'IETF [37] et peuvent être envoyés en utilisant par exemple le bit RE de l'en-tête IPv4 [41]), ou l'option destination ConEx en Internet Protocol version 6 (IPv6) [42] :

- Not-ConEx : Ce paquet n'utilise pas ConEx.
- ConEx-Capable : Le protocole de transport utilise ConEx et contient l'un ou plusieurs des signaux suivants :
  - Re-Echo-Loss : Une perte a été détectée.
  - Re-Echo-ECN : Un paquet a été marqué avec ECN.
  - Credit : Utilisé pour anticiper la congestion au niveau de l'auditeur (cf. section 2.1.6)
  - ConEx-Not-Marked : Ne contient aucune indication de congestion.

2.1.3 Re-ECN

Re-ECN est une implantation possible de ConEx proposée pour IPv4 [41]. Elle utilise le bit 48 (bit RE) de l'en-tête IPv4 afin d'étendre le champ ECN à 3 bits et permettre la définition de 8 points de code. La correspondance entre ces points de code et les signaux ConEx est présentée sur le tableau 2.1.

2.1.4 Modifications à apporter à TCP

Le mécanisme ECN classique décrit dans [18] permet au destinataire d'informer l'expéditeur d'un marquage CE une seule fois par RTT. En effet, même si plusieurs paquets dans un flux sont marqués CE durant un même RTT, le destinataire n'a à sa disposition qu'un seul bit (ECE dans l'en-tête TCP) pour notifier le nombre de paquets qui ont été marqués durant le RTT. Cette information est importante pour ConEx mais également pour d'autres mécanismes comme Data Center TCP (DCTCP) par exemple [25]. Ainsi,

TABLE 2.1 – Signaux ConEx avec l'encodage Re-ECN

Champ ECN	bit RE	Signal ConEx
00	1	Credit (ConEx-Capable)
01	1	ConEx-Not-Marked (ConEx-Capable)
01	0	Re-Echo-ECN ou Re-Echo-Loss (ConEx-Capable)
11	1	Paquet marqué CE (ConEx-Capable)
11	0	Paquet marqué CE et signal Re-Echo (ConEx-Capable)
10	0	ECN legacy (Not-ConEx)
00	0	Not-ECN (Not-ConEx)
10	1	Inutilisé

des modifications à TCP sont proposées par l'IETF [43] pour permettre au destinataire de reporter le nombre exact de paquets marqués CE dans le réseau.

La solution consiste à surcharger les trois bits ECE, Congestion Window Reduced (CWR) et Nonce Sum (NS) afin de former un seul champ nommé Accurate ECN (ACE) comme représenté sur la figure 2.2. Ce champ servira de compteur du nombre de paquets marqués CE détectés par le destinataire, et sera renvoyé à l'expéditeur via les accusés de réception au lieu du seul bit ECE en cas de marquage. L'expéditeur devra à son tour maintenir un compteur local Congestion Exposure Gauge (CEG) pour suivre l'évolution du marquage ECN et exposer ces marques sous forme de signaux Re-Echo-ECN au réseau. L'utilisation de ACE est négociée durant l'établissement de la connexion TCP.

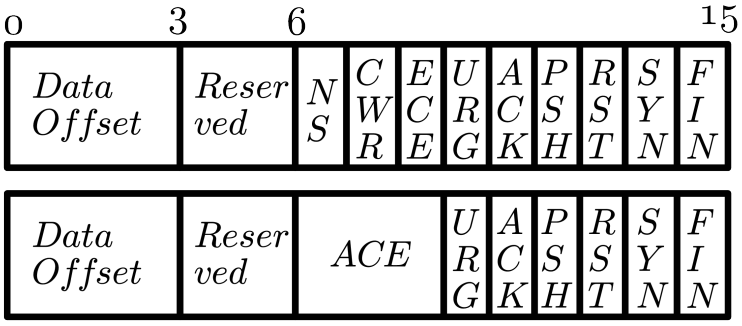

FIGURE 2.2 – Octets 13 and 14 de l'en-tête TCP

L'expéditeur maintient également un second compteur qui suit l'évolution des pertes de paquets. Un compteur Loss Exposure Gauge (LEG) est incrémenté à chaque fois que l'expéditeur détecte une perte de paquet et retransmet celui-ci [44]. La détection d'une perte peut-être plus ou moins exacte selon l'utilisation de l'option SACK. Enfin, le compteur LEG est décrémenté après l'envoi d'un signal Re-Echo-Loss.

2.1.5 Policer de congestion

L'avantage apporté par ConEx est de pouvoir identifier les flux de trafic, ou plus largement dans le cas qui nous intéresse les utilisateurs, selon leur contribution à la congestion dans le réseau, donc selon l'impact de leur trafic sur celui des autres utilisateurs. En entrée de réseau, un policer mesure la quantité de signaux ConEx envoyés par la source du trafic pour évaluer la congestion dont elle est responsable, et réagit lorsque l'utilisateur a consommé les droits à congestion qui lui sont accordés par l'opérateur. L'intérêt du mécanisme d'exposition de congestion par la source proposé par ConEx réside dans le fait que le policer peut être placé en entrée de réseau, et non en sortie comme on aurait à le faire si l'on estimait la congestion à partir des paquets marqués ECN arrivant au destinataire. Un policer en entrée de réseau est toujours plus efficace car il est en mesure de supprimer l'excédent de trafic avant qu'il n'entre dans le réseau.

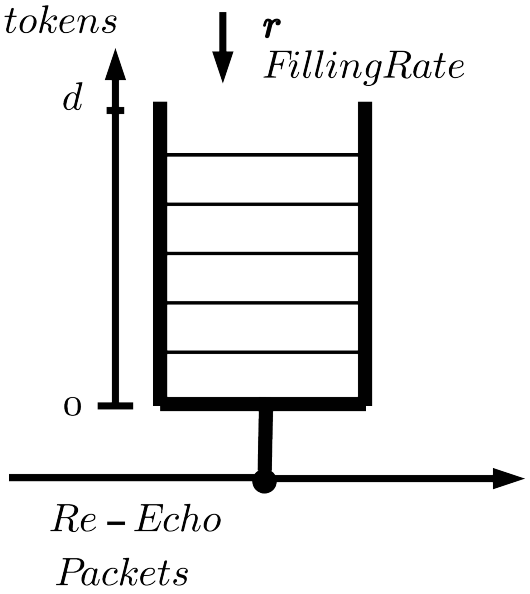

FIGURE 2.3 – Un policer de congestion implanté avec l'algorithme du token bucket

Un policer de congestion peut être implanté grâce à l'algorithme de token bucket comme représenté sur la figure 2.3. Les ressources de l'utilisateur dans le policer sont ici le taux de remplissage r (le débit du volume de congestion accordé) et la profondeur d (la rafale du volume de congestion autorisé) du seau à jeton. Lorsque le policer mesure les signaux ConEx en octets, celui-ci retire le même nombre de jetons qu'il y a d'octets dans le paquet portant un signal Re-Echo-ECN, Re-Echo-Loss, ou Credit émis par l'utilisateur. Quand

le seau à jeton est épuisé par l'utilisateur, le policer réagit sur son trafic typiquement en détruisant ou en abaissant la priorité de ses paquets. Le policer peut être configuré pour compter les signaux en nombre de paquets, mais le résultat est alors moins précis dans le cas d'un trafic contenant des tailles de paquets hétérogènes.

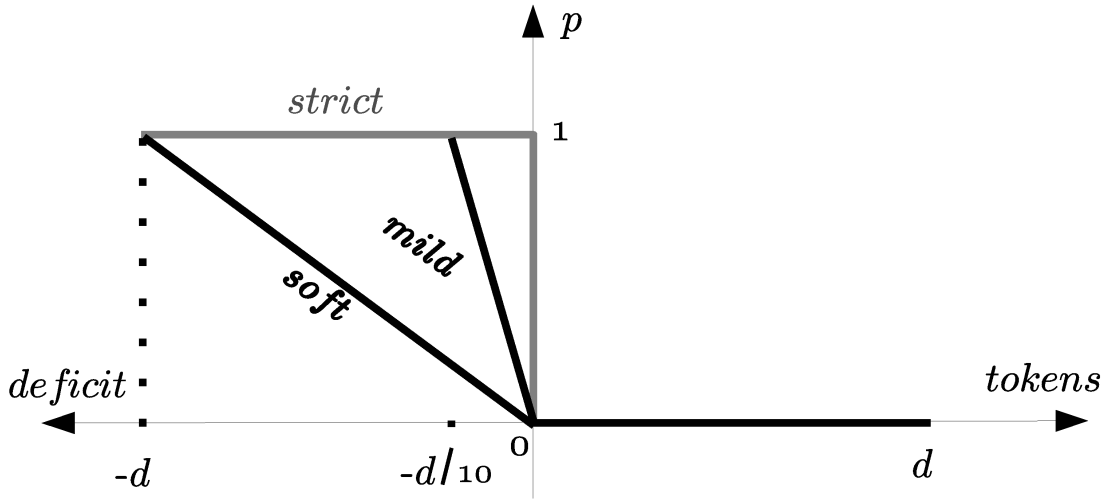

FIGURE 2.4 – Fonctions de destruction de paquets du policer de congestion

Dans notre évaluation des performances de ConEx, le policer de congestion réagira en détruisant les paquets de l'utilisateur car notre objectif est d'empêcher le trafic excédentaire d'entrer dans le réseau. Comme représenté sur la figure 2.4, nous utiliserons trois niveaux de sévérité du policer caractérisés par leur fonction de destruction de paquets : le policer strict détruit tous les paquets lorsque le seau est vide, le policer moyen détruit les paquets avec une probabilité augmentant linéairement de 0 à 1 lorsque la profondeur du seau diminue de 0 à $-d/10$, et enfin, le policer léger détruit les paquets avec une probabilité augmentant linéairement de 0 à 1 lorsque la profondeur du seau diminue de 0 à $-d$.

2.1.6 Auditeur ConEx

Avoir un policer qui sanctionne les utilisateurs selon la quantité de signaux Re-Echo qu'ils émettent peut pousser des utilisateurs malveillants à intentionnellement sous-évaluer la congestion qu'ils rencontrent dans le réseau afin d'échapper au policer même s'ils génèrent un volume trop important de congestion. C'est pour cela qu'une fonction d'audit de la congestion et des signaux Re-Echo est essentielle pour rendre l'information ConEx fiable à l'échelle du réseau, et inciter les utilisateurs à déclarer le bon niveau de congestion qu'ils

rencontrent [37].

La fonction d'audit doit donc permettre de comparer la quantité de signaux Re-Echo envoyés par un utilisateur qui signale la congestion que son flux a rencontrée dans le réseau, et la congestion réelle. Pour cela, l'auditeur doit obligatoirement se trouver en sortie de réseau, ou au moins après la source potentielle de congestion tel un goulot d'étranglement dans un routeur par exemple. Sa présence doit être transparente pour l'utilisateur, sauf s'il détecte que celui-ci n'a pas déclaré la bonne quantité de congestion. Il procède à la vérification de la conformité de la déclaration des utilisateurs en maintenant un état du flux à surveiller et en comparant pour celui-ci les signaux Re-Echo-ECN avec les paquets marqués ECN, et les paquets Re-Echo-Loss avec les pertes subies par le flux [45]. Si l'une de ces deux balances n'est pas équilibrée et qu'elle penche du côté de la congestion réelle, l'auditeur estime que l'utilisateur a sous-évalué la congestion de ce flux dans le réseau et il procède à une sanction visant à rétablir l'équilibre. La sanction envisagée est de détruire un nombre de flux incriminés proportionnel au volume de congestion non déclarée. L'action de l'auditeur est évidemment moins efficace que celle du policer puisqu'appliquée en sortie du réseau.

Comme nous pouvons le constater sur la figure 2.1, il y a un retard intrinsèque dans le mécanisme ConEx d'au moins un RTT entre la notification de congestion qui arrive à l'auditeur et le signal Re-Echo qui correspond à cette notification de congestion. Ce retard peut faire pencher la balance de l'auditeur qui risque de sanctionner injustement un utilisateur dont les paquets Re-Echo ne sont pas encore arrivés à l'auditeur. C'est pour pallier ce problème que le signal Credit a été introduit. Ce dernier doit être envoyé par l'utilisateur au démarrage du flux pour anticiper la congestion que le flux peut rencontrer. Estimer a priori ce volume de congestion de façon précise est toutefois une tâche particulièrement ardue pour la source, ce qui rend ce mécanisme délicat à utiliser.

La fonction d'audit est donc très importante pour rendre ConEx efficace et utile dans le réseau. Mais comme nous pouvons le constater, elle est également très difficile à implanter étant donné l'obligation de maintenir l'état des flux, l'obligation de bien estimer la congestion du réseau pour correctement la comparer à la déclaration de l'utilisateur, l'obligation de faire face au retard intrinsèque des signaux Re-Echo par rapport à la congestion qu'ils exposent. Tout cela rend l'auditeur complexe et vulnérable vis-à-vis des problèmes de sécurité. Plusieurs de ces problèmes ont été traités par [37] et [45] mais l'auditeur pourrait aussi faire l'objet d'une étude encore plus détaillée. Nous allons dans notre cas nous focaliser sur le comportement et les performances génériques de ConEx et du policer de congestion, sans l'intervention de l'auditeur qui par son action sur le trafic pourrait avoir un impact sur le

comportement de ConEx et altérer les résultats obtenus.

Nous allons procéder à l'évaluation des performances de ConEx dans les prochaines sections, en commençant par son comportement avec des flux longs afin d'accorder au mécanisme le temps de réagir, puis avec des flux courts pour observer son comportement avec des temps de réaction réduits, et enfin avec son utilisation pour résoudre le problème du gel des vidéos en streaming en période de congestion afin d'améliorer la qualité de service du trafic vidéo.

2.1.7 Plate-forme expérimentale ConEx

Au début de nos travaux sur ConEx, avant de procéder à l'évaluation des performances, nous avons développé une plate-forme expérimentale qui avait pour but de montrer le fonctionnement du mécanisme ConEx et son potentiel à prendre en compte la congestion dans le réseau afin d'établir l'équité entre les utilisateurs du réseau d'un opérateur. Nous avons présenté cette plate-forme dans le cadre du Salon de la Recherche d'Orange Labs 2013. Cette plate-forme constitue une preuve de concept de ce que nous voulions mettre en évidence. Elle ne constitue pas une base pour pouvoir mener des études de sensibilité complètes vis-à-vis des paramètres. Nous avons donc développé un simulateur qui nous a permis de mener l'évaluation des performances de ConEx afin de mieux établir les possibilités de son déploiement dans le réseau.

2.2 Flux longs

2.2.1 Topologie du réseau simulé

Pour réaliser les simulations, nous utilisons le simulateur Network Simulator 2 (NS2) [46] sur lequel nous avons implanté le mécanisme ConEx en suivant les dernières versions des Requests For Comments (RFC) publiées. Nous utilisons comme encodage de l'information ConEx la proposition IPv4 présentée dans la section 2.1.3.

Le réseau simulé est représenté dans la figure 2.5. D'un côté comme de l'autre du réseau se trouvent 100 utilisateurs. À chaque destinataire à droite du réseau correspond un expéditeur à gauche. Parmi la centaine d'utilisateurs, 90 d'entre eux sont des *light users* n'ayant comme source de trafic qu'un seul flux de type File Transfer Protocol (FTP) qui bien sûr utilise TCP comme protocole de transport. Les 10 utilisateurs restants sont quant à eux des *heavy users* qui génèrent individuellement 36 flux FTP. Avec cette répartition des flux et la nature du partage de ressources de TCP, les *heavy users* à eux seuls sont

responsables de 80% du trafic dans le réseau. Les expéditeurs TCP utilisent CUBIC comme algorithme de contrôle de congestion avec les options SACK [15] et TimeStamps [47]. Les destinataires peuvent signaler de manière précise tous les marquages ECN reçus grâce aux modifications de TCP décrites dans la section 2.1.4. Les expéditeurs à leur tour envoient en direction du policer un signal Re-Echo-ECN pour chaque marquage ECN et un signal Re-Echo-Loss pour chaque paquet perdu détecté. La valeur maximale de la fenêtre TCP est égale à 64KB et la taille d'un paquet est fixée à 1500 octets.

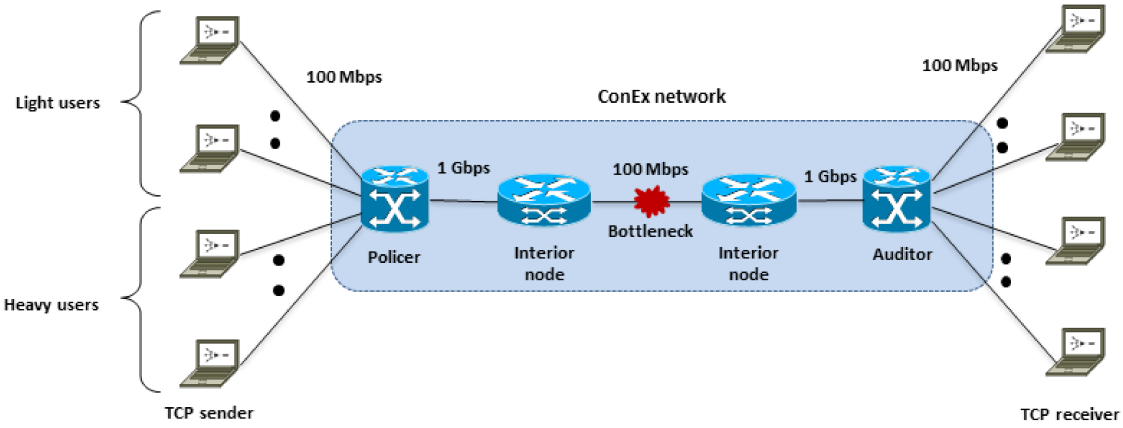

FIGURE 2.5 – Topologie du réseau simulé

Nous allons maintenant décrire la configuration du réseau utilisée pour les simulations. Tous les utilisateurs ont un RTT minimal de 100ms qui est dû aux temps d'émission et de propagation sur les liens. Ils partagent tous un lien de 100Mbit/s qui constituera notre goulot d'étranglement. Sur l'interface de ce lien, nous avons implanté une file RED qui marquera les paquets des utilisateurs ; elle a une taille égale au Bandwidth Delay Product (BDP) afin de contenir 100ms du trafic passant par le lien. La probabilité de marquage des paquets augmente de 0 à $p_{max} = 1$ quand la taille moyenne de la file augmente de 10% à 100% de la taille totale de la file. À l'entrée du réseau se trouve un policer de congestion par utilisateur, implanté avec l'algorithme de token bucket comme décrit dans la section 2.1.5, qui a une profondeur de 64ko (ce qui correspond à 45 paquets sans compter l'en-tête). Il réagira à l'épuisement des jetons dans le seau en détruisant les paquets de l'utilisateur concerné. Les sources étant supposées de confiance dans nos simulations, l'auditeur n'est pas activé en sortie de réseau.

Chaque simulation a une durée de 100s, et est répétée 30 fois afin d'avoir des intervalles

de confiance à 95% adéquats pour chaque métrique. Pour une meilleure lisibilité des figures, ces intervalles ne sont pas représentés quand leur valeur est en deçà de 1% de la valeur estimée de la métrique. Les expéditeurs dans nos simulations ont des sources saturées (c.-à-d. qu'elles envoient au débit maximum possible). Chaque flux démarre aléatoirement et uniformément entre 0 et 300ms après le début la simulation afin de prévenir les problèmes de synchronisation de TCP.

TCP établit une équité entre flux, cela veut dire qu'un utilisateur peut récupérer plus de débit s'il utilise plusieurs flux en parallèle (comme nos *heavy users*). Le policer de congestion implanté en entrée de réseau ne considère pas les flux individuellement, il prend en compte l'ensemble du trafic de l'utilisateur pour surveiller la quantité de congestion provoquée dans le réseau. L'objectif principal de ConEx est d'améliorer l'équité entre les utilisateurs, spécialement entre les *light users* et les *heavy users*. Nous avons donc besoin de surveiller l'impact direct de ConEx sur l'équité entre les utilisateurs, c'est pour cela que nous allons utiliser une métrique spécifique à notre contexte qui a été définie par [48] :

$$iniquité = \frac{Débit\ d'un\ heavy\ user}{Débit\ d'un\ light\ user} \tag{2.1}$$

Dans les prochaines sections, nous allons d'abord évaluer l'influence des paramètres de la simulation sur le comportement de ConEx et ses performances. Nous distinguons deux types de paramètres : internes et externes de ConEx. Les paramètres internes sont ceux qui sont spécifiques à l'implantation de ConEx, p.ex., les paramètres du policer de congestion (taux de remplissage, profondeur du seau, sévérité du policer). Les paramètres externes sont ceux qui proviennent de l'environnement dans lequel opère ConEx : les paramètres du réseau (p.ex., le type de file d'attente, le délai) et l'algorithme de contrôle de congestion de TCP (p.ex., CUBIC ou Compound). Ensuite, nous allons comparer les performances de ConEx avec différentes implantations qui correspondent aux étapes de déploiement successifs du mécanisme.

Le tableau 2.2 récapitule les paramètres de simulation et leurs valeurs. La valeur en gras représente la valeur par défaut utilisée lorsqu'aucune valeur spécifique n'est mentionnée dans le texte.

TABLE 2.2 – Résumé de la valeur des paramètres

Paramètres	Valeurs
Internes	
Taux de remplissage du policer r (paquets/s)	1 – 2 – 3 – 4 – 5 10 – 15 – 20 – 25 – 30 45 – 90 – 120 – 180
Profondeur du policer d (paquets)	5 – 12 – 23 – **45** – 90
Sévérité du policer	Léger–**Moyen**–Strict
Complexité de ConEx	DTConEx–REDConEx ECNConEx–**FullConEx**
Externes	
RTT minimum (ms)	20 – 50 – **100** – 150 – 200
Algorithme de contrôle de congestion TCP	**Cubic**–Compound
Taille de la file q (ms)	10 – 20 – 50 – **100** – 200
Seuil minimal de la file mTh (%q)	2.5 – 5 – **10** – 20 – 50
Seuil maximal de la file MTh (%q)	20 – 40 – 60 – 80 – **100**
Probabilité maximale de destruction/marquage de la file	0.1 – 0.25 – 0.5 – 0.75 – **1**

2.2.2 Sévérité du policer

Impact du policer sur l'iniquité

La figure 2.6 représente l'iniquité moyenne en fonction des différentes valeurs de taux de remplissage du policer attribuées à un utilisateur dans la simulation. Chaque courbe représente un niveau de sévérité du policer (léger, moyen, strict) comme expliqué dans la section 2.1.5. Les autres paramètres sont configurés à leur valeur par défaut, en gras sur le tableau 2.2. La ligne horizontale de couleur rouge au-dessus des autres courbes représente l'iniquité lorsque le policer de congestion est désactivé, seul TCP effectue du contrôle de congestion dans ce cas, et TCP assure une équité uniquement entre flux. Comme les *heavy users* ont 36 flux chacun alors que les *light users* n'en ont qu'un seul, l'iniquité mesurée est égale à 36 comme nous nous y attendions. Quand le policer est activé (les trois autres courbes), ce sont les *heavy users* qui vont le plus subir la réaction du policer. Comme ce dernier force les *heavy users* à réduire leur débit, les *lights users* peuvent exploiter le débit libéré sur le goulot d'étranglement, ainsi l'iniquité entre utilisateurs diminue.

Sur la figure 2.6, l'iniquité atteint pour les trois courbes une valeur minimale, ce qui suggère l'existence d'un taux de remplissage optimal du policer permettant d'obtenir la meilleure équité entre utilisateurs dans le contexte proposé. Des deux côtés de l'optimum, l'iniquité augmente, mais pour deux raisons différentes. À droite, lorsque le taux de rem-

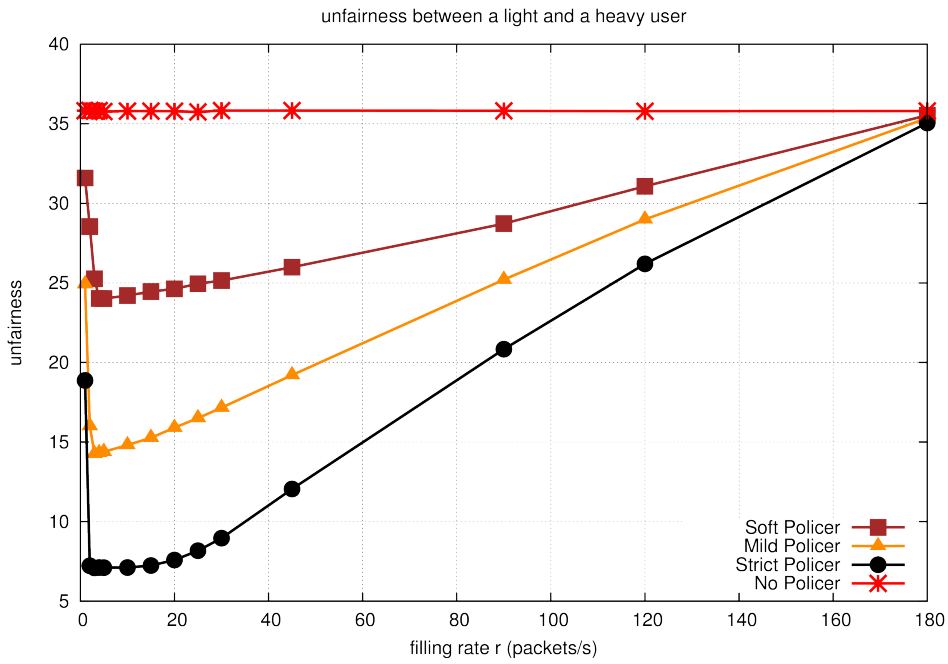

FIGURE 2.6 – Iniquité entre un *heavy user* et un *light user*

plissage r augmente, le seau à jeton se vide moins vite. Par conséquent les *heavy users* sont de moins en moins affectés par le policer. Ils arrivent à garder un débit élevé avec des valeurs de r grandes par rapport à l'optimum, donc l'iniquité est plus grande pour ces valeurs. Lorsque le taux de remplissage est suffisamment élevé, le policer ne réagit plus face au trafic des *heavy users*, conduisant à une valeur d'iniquité qui est égale à celle obtenue sans policer de congestion ($iniquité = 36$). À gauche de l'optimum, le groupe des *light users* est affecté par le policer de la même manière que le groupe des *heavy users*, car le taux de remplissage est insuffisant pour pallier la congestion, même très réduite, qu'induisent les *light users* dans le réseau. Avec la réaction du policer, les *light users* sont contraints de réduire leur débit et l'iniquité augmente pour les valeurs basses de r par rapport à l'optimum.

Que les *light users* subissent la réaction du policer de congestion est contre-productif si le but est de les protéger de l'utilisation excessive des *heavy users* et de réduire l'iniquité entre les deux groupes d'utilisateurs. L'attribution des taux de remplissage du policer doit être établie de sorte que les *light users* évitent l'influence du policer de congestion, tout en prévenant la surcharge du réseau en limitant les *heavy users* et autant que nécessaire. C'est ainsi que l'on obtient la valeur optimale du taux de remplissage.

Impact de la sévérité sur l'iniquité

Pour évaluer l'impact de la sévérité du policer, nous avons utilisé trois niveaux de policers (cf. section 2.1.5), un policer léger, un policer moyen, et un policer strict qui détruisent les paquets avec de plus en plus d'intensité. La figure 2.6 montre que les trois policers atteignent un minimum d'iniquité pour quasiment la même valeur de r, mais qu'ils ne réduisent pas globalement l'iniquité avec la même ampleur. Plus le policer est sévère, plus la probabilité de destruction des paquets des *heavy users* est grande. Les *heavy users* sont donc contraints de réduire encore plus leur débit, ce qui réduit l'iniquité.

La différence de réduction de l'iniquité entre les trois policers est considérable. En effet, lorsque le policer détruit des paquets, la source implantant ConEx réagit en envoyant encore plus de signaux Re-Echo-Loss, ce qui amène éventuellement à une réaction encore plus forte du policer en détruisant plus de paquets, constituant une boucle qui pourrait ne plus prendre en compte l'état du réseau. Avec un policer très sévère, le risque encouru est d'avoir un utilisateur réduisant continuellement son débit à cause de cette boucle alors que le réseau n'est plus en état de congestion. Cet artefact dans le comportement du mécanisme ConEx doit être pris en compte lors de la conception d'un algorithme pour le policer de congestion.

2.2.3 Profondeur du policer

La profondeur du policer correspond à la rafale de signaux ConEx qu'un opérateur permet à un utilisateur d'envoyer. Elle influence la vitesse avec laquelle le policer réagira face à un utilisateur induisant de la congestion dans le réseau. La profondeur du policer influence aussi la fonction de destruction des paquets des policers léger et moyen (plus la taille du policer est réduite, plus la fonction est abrupte) comme présenté dans la section 2.1.5.

Les figures 2.7 et 2.8 montrent l'iniquité par rapport au taux de remplissage pour des valeurs différentes de profondeur du policer respectivement dans le cas d'un policer léger et d'un policer moyen. Dans les deux cas, alors que la profondeur du policer décroît, le policer devient plus réactif à la congestion, et moins permissif par rapport aux *heavy users*.

Dans le cas du policer strict, la fonction de destruction des paquets est indépendante de la profondeur du policer (cf. section 2.1.5), alors seule la vitesse de réaction du policer est affectée par la profondeur d, la sévérité ne l'est pas. Comme nous pouvons le constater sur la figure 2.9, il n'y a quasiment aucune différence dans l'évolution de l'iniquité entre les différentes valeurs de d. Les flux de longue durée peuvent réagir à la congestion en adaptant

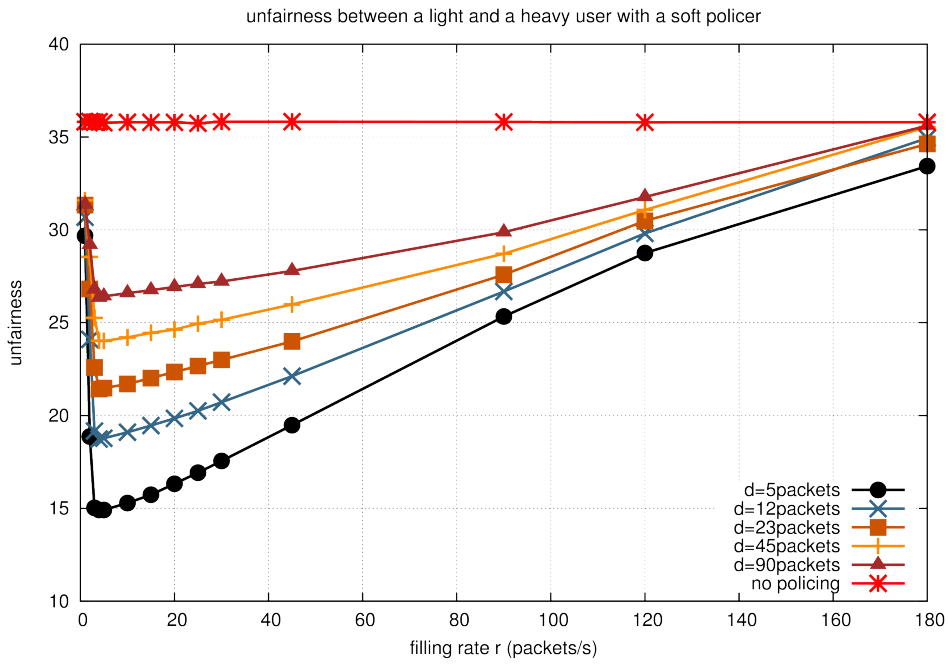

FIGURE 2.7 – L'iniquité pour différentes valeurs de profondeur du seau pour un policer léger

leur débit, ils induisent par conséquent un débit stable de signaux de congestion ConEx. Si ce débit de congestion est supérieur au taux de remplissage du policer, ce n'est qu'une question de temps pour que le seau à jeton soit épuisé totalement et pour que le policer commence à réagir face aux utilisateurs qui induisent le plus de congestion. La réaction dans ce cas aura la même sévérité quelle que soit la valeur de la profondeur du policer.

Sur le long terme, l'iniquité sera bien moins affectée par la profondeur du policer que par le taux de remplissage et par la probabilité de destruction des paquets. Donc la vitesse de réaction du policer, influencée par sa profondeur, a un impact bien moins significatif sur l'iniquité que la sévérité qui est quant à elle influencée par le taux de remplissage et par la fonction de destruction des paquets.

Pour continuer nos évaluations des performances de ConEx, nous allons utiliser un policer de sévérité moyenne et nous allons voir l'influence des autres paramètres impliqués dans le fonctionnement du mécanisme. Nous allons tout de suite évaluer l'impact du RTT sur le mécanisme.

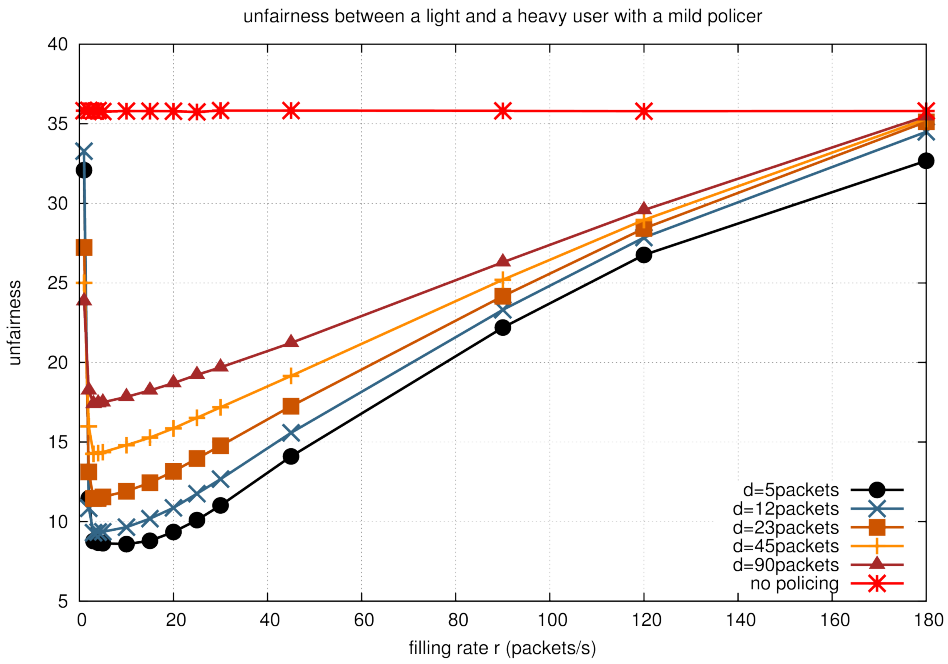

FIGURE 2.8 – L'iniquité pour différentes valeurs de profondeur du seau pour un policer moyen

2.2.4 Round Trip Time

Le RTT réseau qui est dû au temps d'émission et de propagation sur les liens nous importe car il entre en compte dans la boucle de fonctionnement de ConEx. Avoir un court RTT permet à un flux TCP de rapidement augmenter sa fenêtre de congestion, atteignant un débit élevé très rapidement. Cela augmente la probabilité pour les paquets du flux d'être marqués ou détruits. En une seule seconde, un flux peut effectuer plusieurs allers-retours dans le réseau, ce qui mettra ses paquets à chaque RTT dans la file d'attente RED du goulot d'étranglement, paquets qui seront potentiellement marqués ou détruits. Cela peut drastiquement augmenter le nombre de signaux Re-Echo-ECN et Re-Echo-Loss qui sont émis par le flux sur une certaine période de temps. L'utilisateur pourrait alors avoir un débit de congestion très élevé et consommer rapidement les jetons de son policer de congestion, ce qui amènera à une plus grande sévérité de la part du policer.

Dans les simulations, nous faisons varier la valeur minimale du RTT de 20ms à 200ms. Comme le montre la figure 2.10, l'iniquité diminue significativement lorsque le RTT décroît, et la différence constatée sur les courbes est significative, particulièrement pour des valeurs

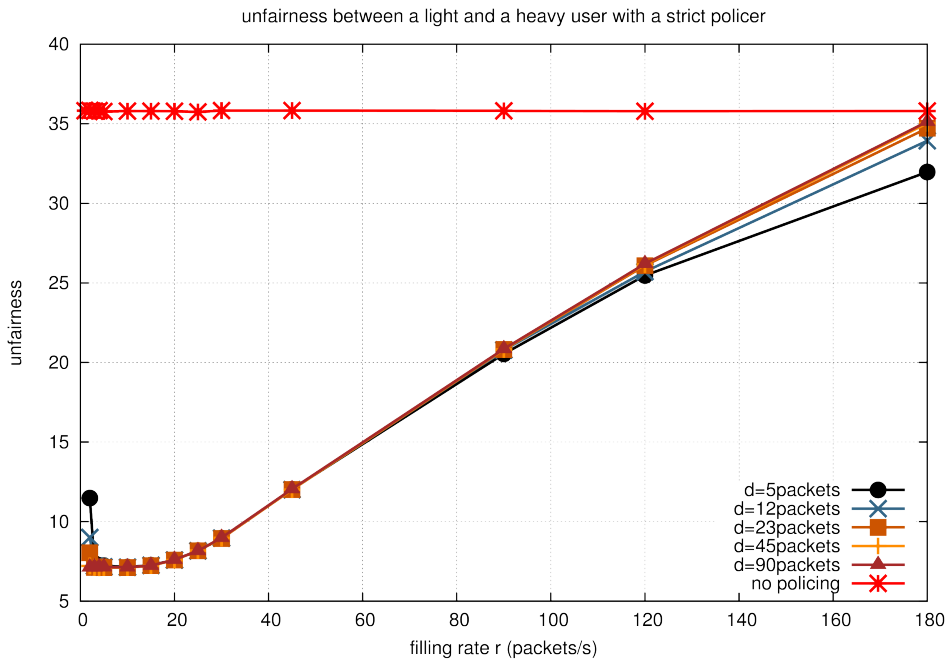

FIGURE 2.9 – L'iniquité pour différentes valeurs de profondeur du seau pour un policer strict

de RTT inférieures à 100ms. Pour les valeurs de RTT les plus faibles, même la valeur de taux de remplissage la plus haute ($r = 180 paquets/s$) n'est pas suffisante pour permettre aux *heavy users* d'éviter la réaction du policer. L'iniquité reste donc toujours basse malgré un taux de remplissage élevé accordé aux *heavy users*.

Dans le cas de RTTs hétérogènes entre utilisateurs, ceux qui ont les flux les plus courts passant par des points de congestion produiront le plus de paquets Re-Echo et consommeront le plus de jetons dans le policer. À taux de remplissage identique, cela se traduit par une plus grande sévérité envers les utilisateurs aux RTTs les plus courts.

Les résultats présentés ici montrent clairement que le RTT est une caractéristique du réseau qui influence beaucoup les performances de ConEx, spécialement dans la réaction du policer vis-à-vis des utilisateurs. Le policer de congestion devrait donc prendre en compte cette caractéristique pour la configuration des taux de remplissage attribués aux utilisateurs, par exemple en fondant l'attribution sur un court RTT.

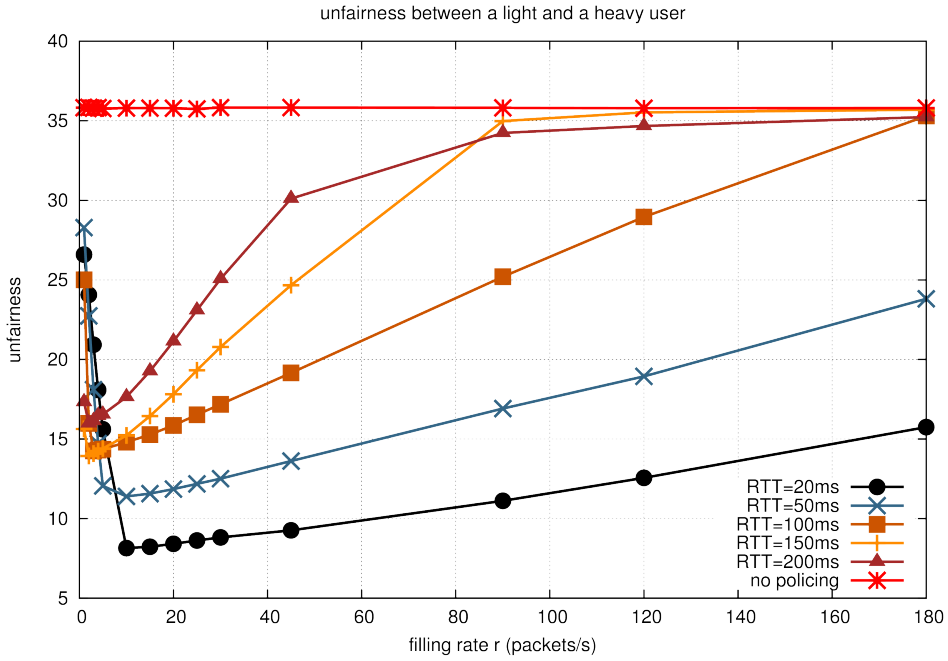

FIGURE 2.10 – L'iniquité pour différentes valeurs de RTT

2.2.5 Algorithme de contrôle de congestion de TCP

Sur la figure 2.11, nous comparons l'influence sur ConEx de CUBIC et Compound, deux algorithmes de contrôle de congestion très utilisés. Cubic est plus agressif que Compound et obtient par conséquent plus de débit sur le goulot d'étranglement, ce qui peut mener à plus de congestion. En conséquence, on pouvait s'attendre à ce que CUBIC consomme plus de jetons que Compound et mène à plus de sévérité de la part du policer face aux *heavy users*, réduisant ainsi plus l'iniquité que Compound. Mais au contraire, les résultats sur la figure 2.11 montrent que Compound est légèrement plus efficace pour réduire l'iniquité que CUBIC, même si ce n'est que d'une très courte marge. Cependant, la différence entre CUBIC et Compound reste quasiment constante lorsque le taux de remplissage r varie, ce qui veut dire qu'ils n'ont pas un impact significatif sur le comportement de ConEx.

2.2.6 Paramètres de la file d'attente

La file RED que nous utilisons dans nos simulations comporte quatre paramètres principaux : la taille de file q, le seuil minimal de la file mTh, le seuil maximal de la file MTh (les deux seuils sont exprimés en pourcentage de la taille q), et la probabilité maximale de mar-

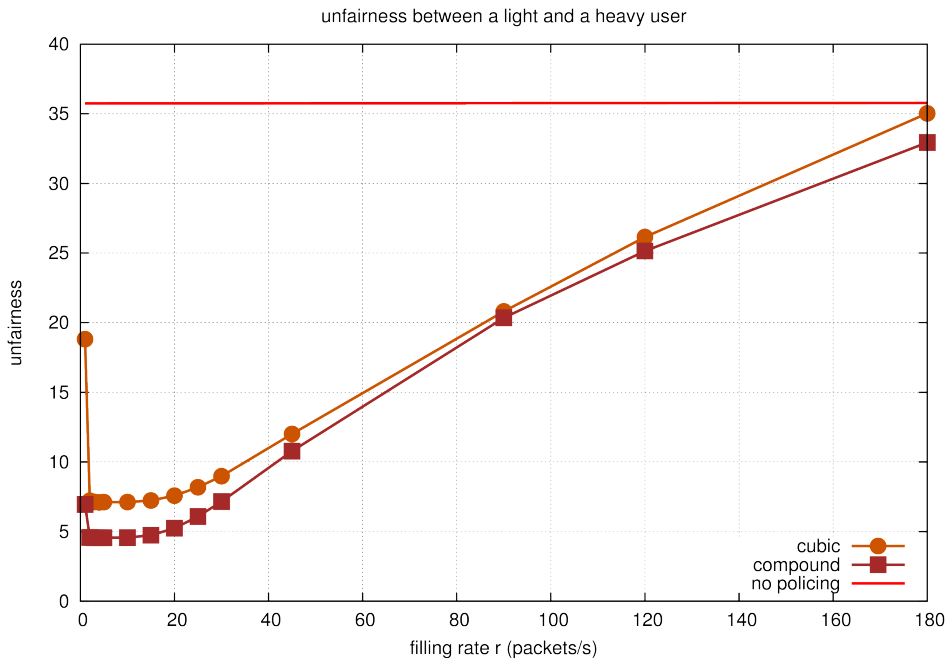

FIGURE 2.11 – L'iniquité pour CUBIC and Compound

quage p_{max}. Dans cette section, nous allons étudier l'influence de chacun de ces paramètres sur le comportement de ConEx, et tenter de quantifier leur impact sur l'iniquité.

Dans notre évaluation des performances de ConEx, la file RED marque les paquets lorsque la taille moyenne de la file d'attente dépasse le seuil minimal mTh, et les paquets n'y sont perdus que lorsque la file déborde. Le marquage ECN produit deux effets qui vont influencer la réduction de la charge de trafic sur le réseau. Premièrement, il invite la source du trafic marqué à réduire son débit d'émission, de la même manière que lors de la détection d'une perte. Deuxièmement, il invite la source de trafic à envoyer un signal Re-Echo-ECN qui consommera des jetons dans le policer de congestion, augmentant la probabilité que ce dernier réagisse.

Taille de la file

Nous faisons varier la taille de la file qui va contenir de 10ms à 200ms de trafic transitant par le goulot d'étranglement (cf. tableau 2.2). Une petite file d'attente se remplit très vite et déborde rapidement, ce qui aboutit à un nombre important de paquets perdus. Avec une file RED, comme la probabilité de marquage augmente avec la taille moyenne de la file, la

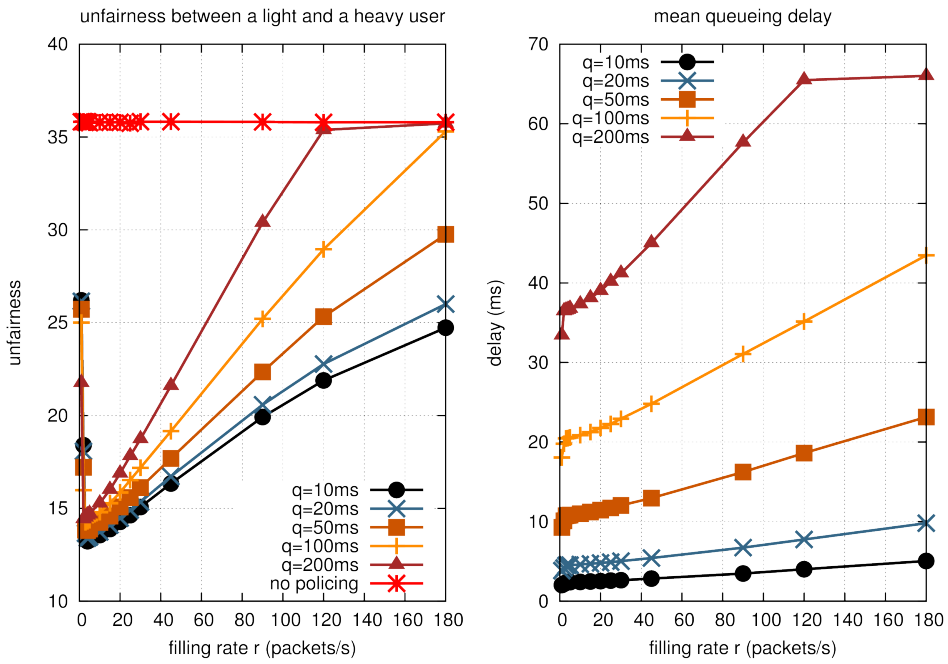

FIGURE 2.12 – L'iniquité pour différentes valeurs de taille de la file

fraction de paquets marqués augmente rapidement dans le cas d'une file de petite taille. Donc plus la taille de file est réduite, plus des signaux Re-Echo-ECN et Re-Echo-Loss sont émis par les *heavy users*. Le policer qui réagit en conséquence les force à réduire leur débit, et comme nous le montre la figure 2.12, l'iniquité décroît significativement avec la taille de la file d'attente.

Le policer qui agit à l'entrée du réseau en limitant le trafic des *heavy users* permet aux *light users* d'avoir un plus grand débit dans le goulot d'étranglement, mais cela conduit également à une réduction de délai dans la file d'attente. Plus le taux de remplissage diminue, rendant le policer plus sévère, plus le délai moyen d'un paquet dans la file diminue comme nous le montre la figure 2.12.

Pour un taux de remplissage donné, le délai présente de grandes variations, ce qui est naturel puisque les files d'attente ont des tailles différentes, mais comparé à l'impact très réduit de la taille de la file sur l'iniquité autour du filling rate optimum, son impact sur le délai est plus significatif. Si le délai dans la file d'attente n'est pas la contribution principale au RTT d'un flux, on peut considérer que l'influence de la taille de la file d'attente sur le comportement de ConEx, autour du taux de remplissage optimum qui nous intéresse, est moins important que sur le délai en lui-même.

Seuil minimal

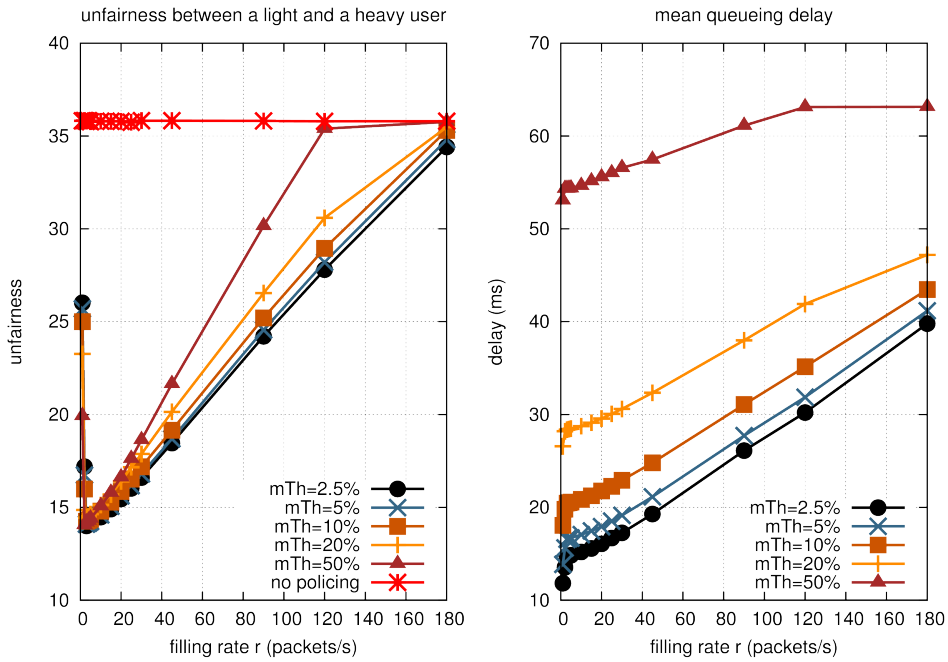

FIGURE 2.13 – L'iniquité pour différentes valeurs du seuil minimal mTh

Le seuil minimal de marquage mTh détermine dans la file RED le moment où les paquets commenceront à être marqués. On peut s'attendre à ce que plus tôt débutera le marquage, plus de signaux Re-Echo-ECN seront envoyés par les utilisateurs, et donc plus sévère sera le policer. Sur la figure 2.13, l'iniquité diminue bien avec la diminution de la valeur seuil minimal. Toutefois, la diminution est très minime par rapport aux variations du seuil minimal de marquage, particulièrement pour les valeurs du taux de remplissage autour de la valeur optimale. Ceci rend ConEx assez insensible aux variations du seuil minimal de marquage, surtout autour du taux de remplissage optimum. Cela peut s'expliquer aussi par le fait que, comme le seuil maximal et la probabilité maximale de marquage sont inchangés, la variation du seuil minimal de marquage aboutit à un marquage plus réactif mais en même temps moins agressif.

En revanche, le délai est grandement influencé par la variation du seuil minimal de marquage (cf. figure 2.13), comme nous l'avons observé précédemment dans le cas de la taille de la file d'attente. Ceci est tout à fait naturel puisqu'une des motivations initiales de l'introduction de l'algorithme de RED est celui de contrôler le délai dans la file d'attente. Ici

encore une fois, nous pouvons conclure que le paramétrage du seuil minimal de marquage, qui a peu d'influence directe sur ConEx lorsque le délai dans la file d'attente n'est pas la contribution principale au RTT, devrait avoir pour objectif principal de contrôler le délai.

Seuil maximal

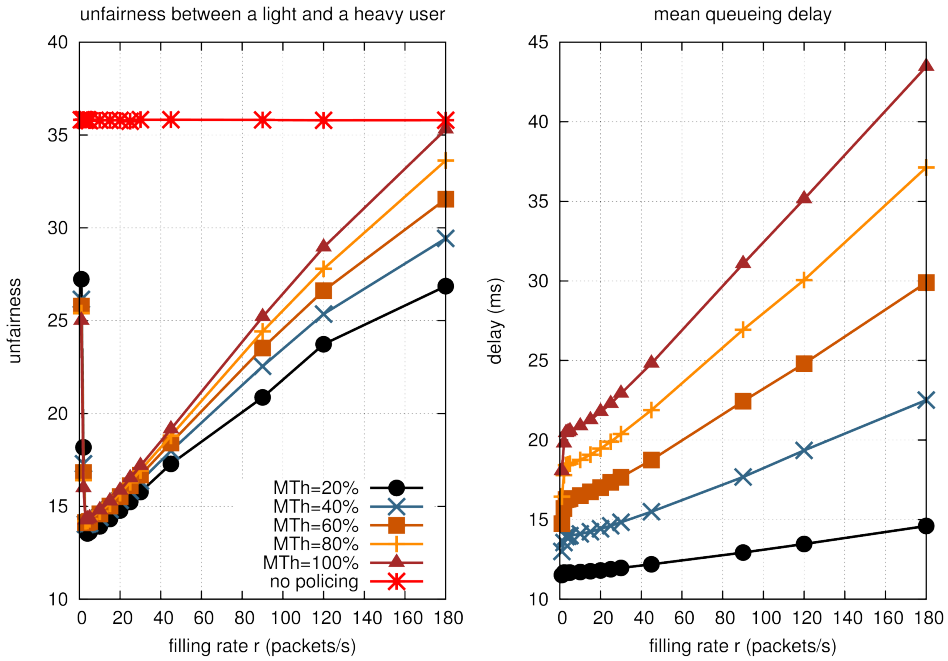

FIGURE 2.14 – L'iniquité pour différentes valeurs du seuil maximal MTh

Lorsque la taille moyenne de la file dépasse le seuil maximal, tous les paquets dans la file sont marqués. Par conséquent, plus le seuil maximal sera bas, plus nous aurons un grand nombre de signaux Re-Echo-ECN émis par les *heavy users*, ce qui conduira à une plus grande sévérité du policer de congestion. Sur la figure 2.14, nous observons que l'iniquité diminue quand on réduit le seuil maximal de marquage à cause du nombre plus élevé de paquets marqués dans la file. Cependant, nous pouvons voir que l'iniquité est peu affectée par la variation du seuil maximal de marquage autour de la valeur optimale du taux de remplissage. Le délai diminue également avec la diminution du seuil maximal, et comme pour le seuil minimal, l'impact de ce dernier sur le délai est très important. À partir de ces résultats, nous pouvons dire que ConEx est peu sensible à la configuration du seuil maximal de marquage de la file d'attente.

Probabilité maximale de marquage

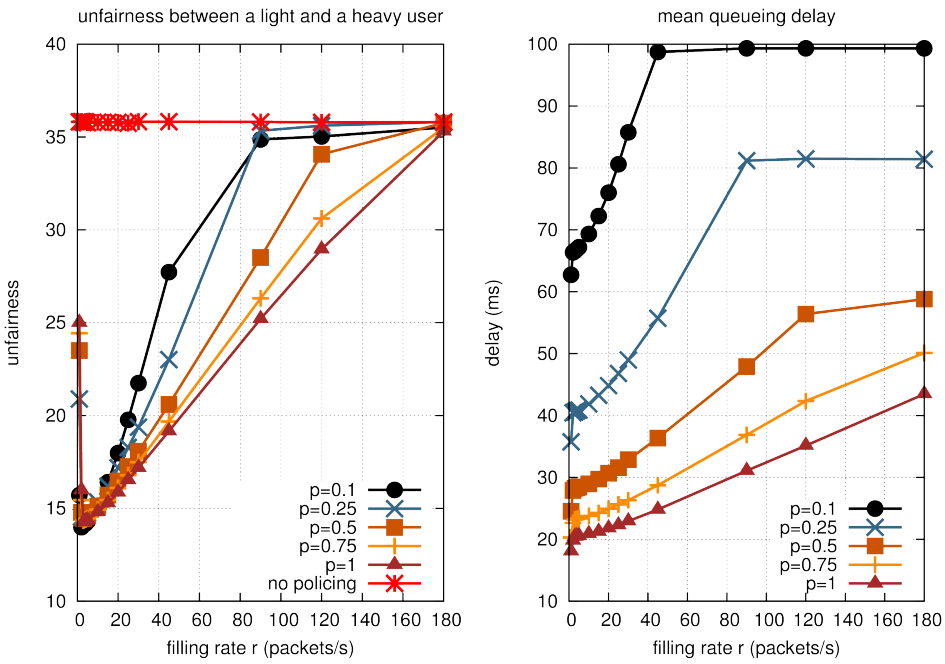

FIGURE 2.15 – L'iniquité pour différentes valeurs de la probabilité maximale de marquage

La probabilité maximale de marquage est atteinte lorsque la taille moyenne de la file est égale au seuil maximal de marquage. Plus la probabilité maximale est élevée, plus les paquets sont marqués et mènent à plus de signaux Re-Echo-ECN de la part des *heavy users*, rendant le policer plus sévère face à leur trafic. Comme représenté sur la figure 2.15, une forte probabilité de marquage réduit l'iniquité mais les différences sont quasiment nulles autour de la valeur optimale du taux de remplissage.

Pour un taux de remplissage du policer bas (p.ex. pour $r < 20 paquets/s$), la tolérance du policer aux signaux Re-Echo est si basse que la probabilité maximale de marquage affecte très peu la diminution de l'iniquité. Nous pouvons observer sur la figure 2.15 que pour un taux de remplissage r supérieur à 90 paquets/s et une probabilité maximale de marquage p_{max} inférieure à 0.25, l'iniquité n'est quasiment pas diminuée car, avec un marquage léger, le débit des signaux Re-Echo est rapidement égal au taux de remplissage du policer, ce qui aboutit à des *heavy users* quasiment pas touchés par le policer.

Le délai dans la file d'attente est très influencé par la variation de la probabilité de marquage. Avec une faible probabilité de marquage, les utilisateurs subissent très peu de

limitation de débit pour un taux de remplissage assez élevé, et peuvent occuper une grande partie de la file d'attente (p.ex. avec $p_{max} = 0.25$), ou même l'occuper entièrement (p.ex. avec $p_{max} = 0.1$), menant à un grand délai dans la file d'attente. Lorsque la probabilité de marquage augmente, la limitation des *heavy users* par le policer est plus importante, ce qui mène à une file d'attente moins remplie donc à un délai réduit. Pour des taux de remplissage faibles, le policer de congestion est encore plus sévère, et la réduction du délai est encore plus nette.

Conclusion sur les paramètres de la file d'attente

De l'étude que nous avons menée sur les paramètres de la file RED, une tendance peut être observée : il y a certainement un impact des paramètres de la file sur l'amélioration de l'équité entre les utilisateurs produite par l'utilisation de ConEx, mais cet impact est bien plus significatif sur le délai dans la file d'attente. Nous pouvons conclure que le paramétrage de la file RED devrait avoir pour objectif premier le contrôle du délai dans la file plutôt que son influence sur ConEx. De plus, quel que soit le paramétrage de la file RED, le policer de congestion conduit toujours à une diminution du délai dans la file grâce à la réduction de la charge de trafic en entrée de réseau.

2.2.7 ConEx avec une complexité croissante

La facilité de déploiement de ConEx est une préoccupation majeure pour un fournisseur de service et pour un opérateur de réseau. Si le fournisseur de service peut mettre à niveau ses serveurs de contenus, il ne contrôle ni les mécanismes de file d'attente dans les réseaux qui mènent vers l'utilisateur, ni la pile TCP/IP des destinataires. L'opérateur de réseau pour sa part contrôle la stratégie de file d'attente implantée dans son réseau et l'algorithme du policer de congestion en entrée du réseau, mais il n'a pas le pouvoir sur les piles TCP/IP des utilisateurs de son réseau. Dans ce contexte, la possibilité d'un fonctionnement de ConEx avec des modifications minimales est un facteur clé dans l'introduction du mécanisme dans le réseau. ConEx permet justement un déploiement incrémental nécessitant peu de modifications pour être opérationnel. Par la suite, il pourra être amélioré, étape par étape, en augmentant la complexité de son implantation pour fournir une indication de plus en plus précise de l'ampleur de la congestion dans le réseau.

Les modifications minimales nécessaires pour implanter ConEx sont les modifications sur l'expéditeur qui va réagir à la détection d'une perte en envoyant un signal Re-Echo-Loss au policer de congestion. Dans ce cas, l'activation d'ECN n'est nécessaire ni pour

TABLE 2.3 – ConEx avec une complexité croissante

Cas	File d'attente	Expéditeur	Destinataire
DTConEx	DropTail	Sans ECN	Sans ECN
REDConEx	RED	Sans ECN	Sans ECN
ECNConEx	RED	ECN classique	ECN classique
FullConEx	RED	ECN précis	ECN précis

l'expéditeur ni pour le destinataire, et la file RED peut être remplacée par une simple file DropTail qui perd les paquets uniquement lorsqu'elle déborde. Dans les paragraphes suivants, nous allons nommer ce cas $DTConEx$. La prochaine étape dans les modifications à apporter est d'activer RED sur les équipements du réseau afin d'améliorer la réactivité à l'apparition de la congestion. ECN n'est toujours pas utilisé, ici ConEx ne va réagir qu'aux paquets détruits par la file RED en envoyant des signaux Re-Echo-Loss. Nous nommerons ce cas $REDConEx$. Une autre étape dans les modifications nécessite l'activation d'ECN par l'expéditeur et le destinataire. Les files RED implantées dans le réseau peuvent marquer les paquets pour signaler le risque de congestion, mais le destinataire n'implante pas les modifications de TCP pour fournir un comptage précis du marquage ECN à l'expéditeur (cf. section 2.1.4), il n'utilise que la version classique d'ECN qui ne signale qu'une seule notification de congestion par RTT quel que soit le nombre de paquets marqués durant ce RTT par la file RED. Nous nommerons ce cas $ECNConEx$. La dernière étape de modifications nécessite l'utilisation de la version précise de TCP par le destinataire, permettant un comptage précis du marquage ECN dans le réseau. Nous nommerons ce cas $FullConEx$. C'est ce mode d'implantation que nous avons évalué jusqu'à présent. Les quatres cas sont résumés sur le tableau 2.3

La figure 2.16 représente l'iniquité en fonction du taux de remplissage du policer pour quatre scénarios où nous faisons varier le nombre de flux des *heavy users* (9, 18, 27, and 36), tandis que les *light users* n'ont toujours qu'un seul flux. Dans chacun des scénarios, la ligne horizontale de couleur rouge représente la valeur de l'iniquité sans utilisation de policer de congestion, alors que les quatre autres courbes représentent les cas de déploiement de ConEx expliqués plus haut. Plus le nombre de flux d'un *heavy user* augmente, plus sa contribution à la congestion augmente. Le *heavy user* aura alors à émettre plus de signaux Re-Echo qui consommeront plus de jetons, et il subira donc plus sévèrement la réaction du policer. En conséquence, l'éventail de valeurs du taux de remplissage permettant une diminution de l'iniquité est plus large lorsque le nombre de flux est élevé.

Dans tous les scénarios, nous remarquons que $FullConEx$ et $ECNConEx$ donnent

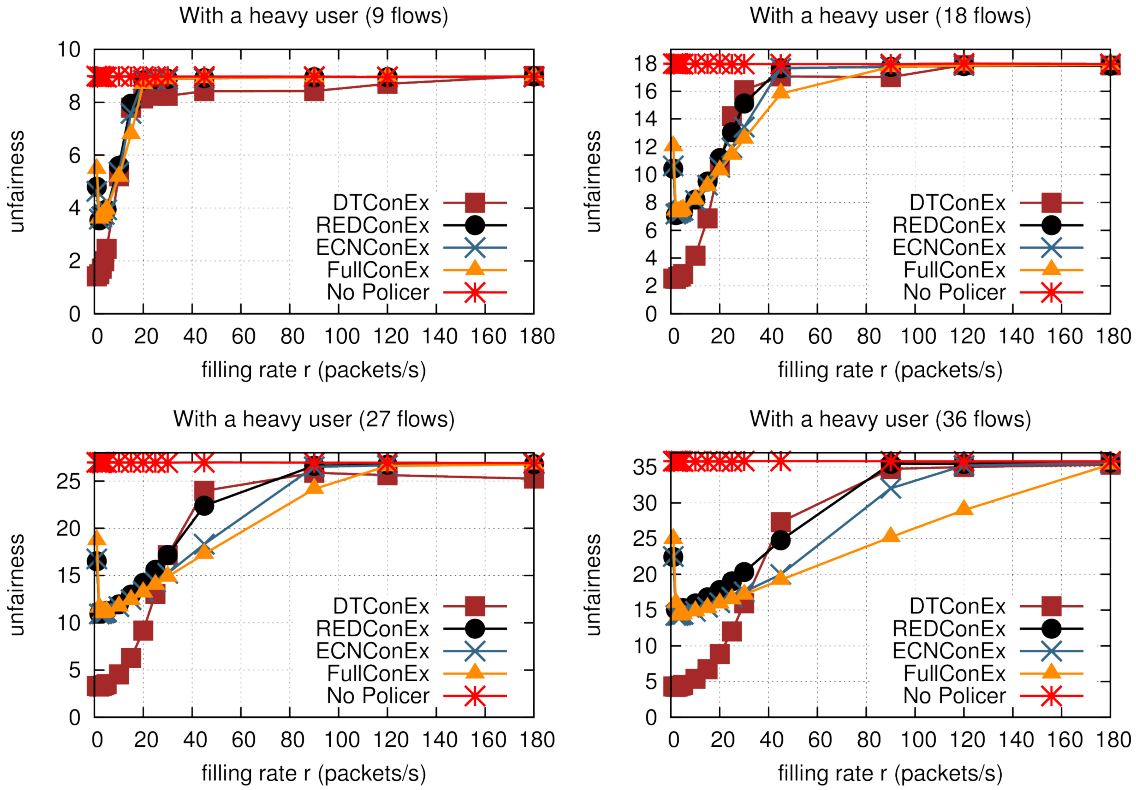

FIGURE 2.16 – Iniquité avec DTConEx, REDConEx, ECNConEx et FullConEx

des résultats très similaires et réduisent plus l'iniquité que *REDConEx*. La raison en est que les deux premiers cas permettent l'utilisation des informations de marquage ECN et les informations sur les pertes de paquets pour envoyer des signaux Re-Echo-ECN et Re-Echo-Loss, ce qui les rend plus précis que *REDConEx* qui n'autorise que les indications de perte. De la même manière, *FullConEx* est légèrement plus performant que *ECNConEx* dans la réduction de l'iniquité car il fournit un comptage plus précis des signaux ECN, et nous pouvons le remarquer particulièrement quand le niveau de congestion augmente (avec 27 et 36 flux par *heavy user*). Cela permet au policer de congestion d'être plus précis dans la limitation des *heavy users*.

Le cas *DTConEx* fournit encore moins d'informations sur la congestion car il n'y a génération d'information que lorsque la file DropTail déborde. Pourtant, l'iniquité y est encore plus basse que pour les autres cas pour des valeurs de taux de remplissage qui se trouvent autour de l'optimum. *DTConEx* est plus efficace car il ne force pas les *light users* à réduire leur débit aussi tôt que dans les autres cas. En effet, dans *REDConEx*,

ECNConEx et *FullConEx*, la file d'attente commence à détruire ou à marquer les paquets quand la taille moyenne de la file dépasse le seuil minimal, forçant les *heavy users* ainsi que les *light users* à réagir en réduisant leur débit. Au contraire, la file DropTail ne perd des paquets que lorsque toute la file est entièrement remplie, ce qui donne l'opportunité aux *light users* d'augmenter leur débit, tandis que les *heavy users* sont limités par le policer en entrée de réseau et libèrent le goulot d'étranglement.

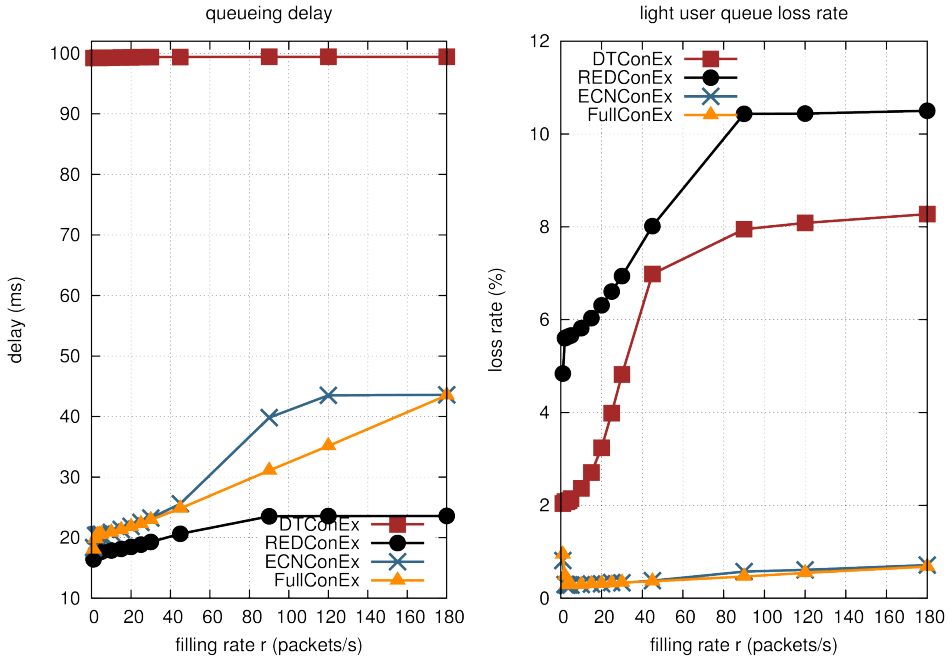

FIGURE 2.17 – Délai moyen et taux de pertes dans la file pour un *light user*

La figure 2.17 représente le délai moyen et le taux de perte subis par un *light user* dans la file d'attente en fonction du taux de remplissage pour le scénario avec 36 flux par *heavy user*. Contrairement à une file RED, DropTail n'agit pas pour réduire le délai dans la file d'attente comme nous pouvons le voir dans le cas *DTConEx*. La pression des flux des utilisateurs fait que la file est toujours pleine, ce qui leur fait subir le délai maximum possible égal à 100ms. Pour *REDConEx*, *ECNConEx* et *FullConEx*, le délai dans la file est réduit par l'action de RED qui détruit les paquets pour le premier cas, et les marque pour les deux derniers, ce qui donne un signal aux utilisateurs pour réduire leur débit. *REDConEx* réduit plus le délai dans la file qu'*ECNConEx* et *FullConEx* grâce à l'action de RED qui libère la file d'attente en détruisant des paquets, alors que dans les deux autres cas les paquets ne sont pas écartés de la file mais uniquement marqués. Le

policer de congestion contribue également à la réduction du délai dans la file en limitant la charge du trafic en entrée de réseau. Son effet est de plus en plus visible quand le taux de remplissage diminue.

En réduisant la pression du trafic sur le goulot d'étranglement, le policer de congestion réduit aussi le taux de perte subi par les *light users*, ce qu'on peut constater sur la figure 2.17 lors de la diminution du taux de remplissage du policer. Pour toutes les valeurs de taux de remplissage, $REDConEx$ atteint des taux de perte bien plus hauts que $DTConEx$ car, avec la même charge de trafic, la file RED commence à détruire les paquets bien plus tôt. Finalement, $ECNConEx$ et $FullConEx$, pour lesquels les paquets sont marqués ECN au lieu d'être détruits, le taux de perte enregistré est bien plus bas pour les *light users* que dans les cas $REDConEx$ et $DTConEx$.

Récapitulatif des performances de chaque niveau de complexité

Le tableau 2.4 récapitule les performances de chaque mode d'implantation en termes d'amélioration de l'équité, de taux de perte subi, de délai moyen dans la file, et de facilité de déploiement. Un déploiement initial de ConEx avec un minimum de modifications comme dans le mode $DTConEx$ est envisageable avant de considérer le déploiement complet avec $FullConEx$ pour avoir un niveau de performances plus élevé.

TABLE 2.4 – Récapitulatif des performances

Cas	Équité	Taux de perte	Délai dans la file	Facilité de déploiement
DTConEx	****	**	*	****
REDConEx	*	*	****	***
ECNConEx	**	****	**	**
FullConEx	***	****	***	*

2.3 Flux courts

Les flux courts représentent une part importante des flux qui traversent l'internet (p.ex., Domain Name System (DNS), contenus webs). Ces flux ne comportent que quelques paquets ; ils terminent leur transmission de données durant la phase de slow-start (en peu de RTTs) avant d'atteindre leur débit de « fair-share » [49]. Cette section vise à étudier comment ConEx, qui est un mécanisme en boucle fermée demandant un certain nombre de RTT pour rassembler les informations de congestion, se comporte avec des flux de courte durée, et s'il apporte une amélioration à leur durée de session.

Pour l'évaluation des performances, nous utilisons la même topologie que dans la section 2.2, mais nous modifions les sources de trafic en flux courts dont la taille ne dépasse pas 10 paquets en tout. Nous utilisons le modèle de trafic agrégé proposé dans [50], qui utilise une distribution gamma pour le temps d'interarrivée des flux, conduisant à l'arrivée d'un nouveau flux chaque 6ms en moyenne. Nos 10 *heavy users* vont générer 80% des flux tandis que les *light users* vont générer les 20% restants. Afin d'avoir de la congestion dans le réseau, un trafic de charge non-ConEx de 90Mbit/s va être généré au niveau du goulot d'étranglement dont le débit est fixé à 100Mbit/s. Un policer strict est utilisé comme présenté dans la section 2.1.5. La métrique que nous allons surveiller est la durée de session d'un flux court.

Chaque simulation dure 600 secondes, 30 simulations sont réalisées pour obtenir chaque point avec des intervalles de confiance à 95% que nous allons représenter sur les figures. Comme nous pouvions nous y attendre, les résultats obtenus au travers des différentes simulations réalisées avec des flux courts montrent que ce type de trafic ne fait pas très bon ménage avec un mécanisme tel que ConEx. Nous ne présentons donc ici qu'un seul résultat, caractéristique du comportement observé.

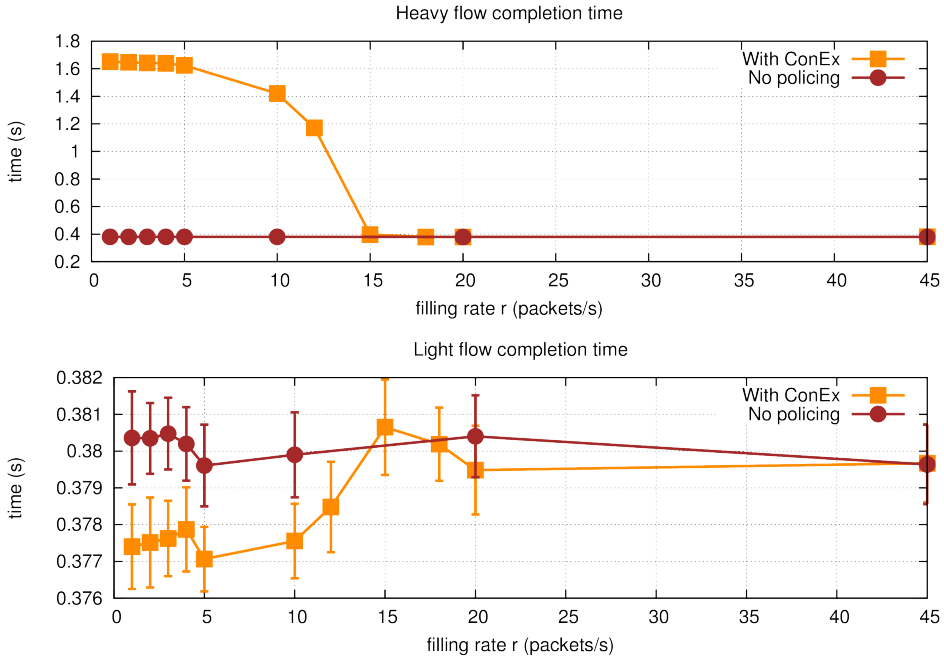

FIGURE 2.18 – Durée de session du flux d'un light et d'un *heavy user*

La figure 2.18 représente la durée de session moyenne du flux d'un *heavy user* et d'un

light user avec et sans l'utilisation de ConEx (implantation de FullConEx dans cette simulation). Les flux TCP, d'une taille de 10 segments, ont une fenêtre initiale de 2 segments, et arrivent à terminer leur transmission de données au bout d'un peu plus de $3 \times RTT = 300ms$, ce qui ne leur permet pas de fournir beaucoup d'informations de congestion à ConEx. Néanmoins, un *heavy user* peut subir la réaction du policer de congestion si le taux de remplissage est suffisamment bas ($r < 15$ paquets/s). Dans ce cas, les pertes de paquets induites par le policer augmentent significativement la durée de session de ces flux. La durée de session d'un flux de 10 segments augmente de 380ms sans intervention du policer jusqu'à 1.64s quand le policer est activé avec le taux de remplissage le plus faible.

La limitation des *heavy users* a pour objectif de libérer le goulot d'étranglement en faveur des *light users*. En effet, nous remarquons que lorsque les *heavy users* subissent l'influence du policer, les *light users* bénéficient d'une réduction de la durée de session de leurs flux, mais la réduction est d'à peine quelques millisecondes, ce que l'on peut difficilement considérer comme une amélioration significative.

Les flux courts ne bénéficient quasiment pas de l'utilisation de ConEx. En effet comme ils se terminent en quelques RTTs, ils ne sont pas en mesure de collecter les informations sur la congestion, ni de réagir à l'action du policer. Quand les flux courts perdent des paquets, ils voient leur durée de session augmenter énormément, de quelques millisecondes à quelques secondes, car ils ont parfois besoin d'attendre un Retransmit TimeOut (RTO) pour retransmettre les paquets et se terminer.

Comme prévu, ConEx n'est pas bénéfique lorsqu'il est appliqué à des flux courts, et nous pouvons nous attendre à d'encore moins bonnes performances si, comme le suggère [49], la fenêtre initiale de TCP est augmentée à 10 segments, cela représente un scénario encore moins favorable que le scénario simulé. Néanmoins, les performances de ConEx observées avec les flux courts n'atténuent pas l'intérêt que nous portons au mécanisme car les flux longs sont la principale source de congestion. Les résultats obtenus ici montrent que si un policer de congestion est utilisé, il serait plus avantageux de se focaliser uniquement sur les flux longs qui, compte tenu de leur durée, parviennent à récupérer les informations sur la congestion du réseau, sont en mesure de la signaler au réseau, et peuvent réagir efficacement aux limitations du policer. La désignation des flux longs, sur lesquels ConEx et le policer de congestion doivent être utilisés, peut être effectuée au niveau des machines d'extrémité lors de la négociation de ConEx en appliquant par exemple aux flux longs un code DSCP spécifique.

2.4 Trafic de vidéo streaming : le cas de YouTube

Nous observons ces dernières années une croissance impressionnante du trafic de vidéo streaming autant sur les réseaux fixes que sur les réseaux mobiles d'Orange : 36%, 26% et 39% du trafic descendant pour le FTTH, l'ADSL et le mobile respectivement [34]. Ceci nous a amené à évaluer comment ConEx pouvait améliorer les performances de ce trafic en contrôlant la congestion dans le réseau. Nous avons choisi comme cas d'utilisation la très populaire plate-forme de streaming vidéo YouTube.

2.4.1 Modèle de serveur YouTube

Plusieurs articles ont analysé la nature du trafic généré par YouTube. Parmi eux, [51] et [52] proposent un algorithme qui reproduit le comportement des serveurs de Youtube, que nous avons implanté sur NS2.

Un serveur émet une vidéo en deux phases. La première phase est appelée Initial Burst où 40 secondes des données de la vidéo demandée sont envoyés au débit maximum possible pour remplir suffisamment la mémoire tampon pour la lecture de la vidéo. La seconde phase est appelée le Throttling ; le serveur y envoie le reste des données de la vidéo en plusieurs morceaux avec un débit d'envoi égal à 1.25 fois le débit d'encodage de la vidéo. La taille d'un morceau est de 64ko, et les morceaux sont envoyés sur une socket TCP qui réserve 2Mo de mémoire d'envoi.

2.4.2 Modèle de lecteur YouTube

Nous utilisons l'approche d'estimation du comportement du lecteur YouTube la plus précise proposée par [53], que nous implantons sur NS2. Cette approche repose sur l'état de la mémoire tampon du lecteur du client. Le lecteur commence à lire la vidéo lorsque la mémoire tampon dépasse un premier seuil $\theta_0 = 2.2s$. Si le contenu de la mémoire est épuisé et descend à moins de $\theta_1 = 0.4s$, la vidéo se fige jusqu'à ce que la mémoire dépasse de nouveau θ_0, puis la lecture de la vidéo continue. Nous récupérons après chaque simulation le nombre de fois N où la vidéo s'est figée et la durée moyenne L de ces périodes afin de calculer la qualité de l'expérience (QoE) des utilisateurs suivant le modèle proposé par [54] en appliquant l'équation suivante :

$$QoE(L, N) = 3.50 \exp(-(0.15L + 0.19) \times N) + 1.50 \tag{2.2}$$

2.4.3 Résultats des simulations

Nous utilisons la même topologie que dans la section 2.2 avec 10 *heavy users* et 50 *light users*. Le scénario simulé est le suivant : durant les 100 premières secondes de la simulation, les *heavy users* ont 20 flux FTP au débit maximum qu'ils peuvent atteindre. Il n'y a pas encore de *light user* dans la simulation, les 10 *heavy users* peuvent se partager équitablement le débit disponible au niveau du goulot d'étranglement. Durant les 100 secondes suivantes de la simulation, les *light users* commencent à demander une vidéo aux serveurs en initiant des requêtes réparties aléatoirement et uniformément sur ces 100 secondes. La vidéo demandée a une durée de 300s et un débit de 1128kbit/s ce qui correspond aux recommendations pour mettre en ligne une vidéo de 360p sur YouTube (1000kbit/s pour le débit de la vidéo, et 128kbit/s pour le débit audio [55]). Les *heavy users*, qui sont responsables de 80% du trafic, auront à se partager le goulot d'étranglement avec les nouveaux arrivants. À $t = 500s$, tous les *light users* devraient avoir visualisé leur vidéo de 300s si celle-ci ne se fige à aucun moment, et les *heavy users* devraient continuer à utiliser le goulot d'étranglement jusqu'à la fin de la simulation 100s plus tard. La QoE moyenne des *light users* est calculée à la fin de chaque simulation.

Une simple file DropTail est utilisée au niveau du goulot d'étranglement. Un policer strict est implanté à l'entrée du réseau et tous les utilisateurs utilisent CUBIC en tant qu'algorithme de contrôle de congestion de TCP sauf mention contraire.

La figure 2.19 montre le débit des *heavy users* et des *light users* en fonction du temps. Nous retrouvons les trois périodes que nous avons décrites : avant l'arrivée des *light users* (0s-100s), en présence des *light users* (100s-500s), et après le départ présumé des *light users* consécutive à la fin de la vidéo (500s-600s).

La figure 2.20 représente la QoE moyenne d'un *light user*, le nombre moyen de fois où la vidéo s'est figée, et la durée moyenne du gel de la vidéo pour les trois cas suivants : en utilisant CUBIC en tant qu'algorithme de contrôle de congestion pour les *heavy users* sans l'intervention d'un policer ConEx, en utilisant CUBIC pour les *heavy users* avec un policer ConEx, et en utilisant Low Extra Delay Background Transport (LEDBAT) [56] en tant qu'algorithme de contrôle de congestion pour les *heavy users* sans policer ConEx.

CUBIC sans policer

Quand aucun policer de congestion n'est utilisé, les flux TCP avec CUBIC se partagent le goulot d'étranglement équitablement. Avec la répartition des flux définie pour les simulations, les *heavy users* récupérent 80% du débit du goulot d'étranglement et les *light users*

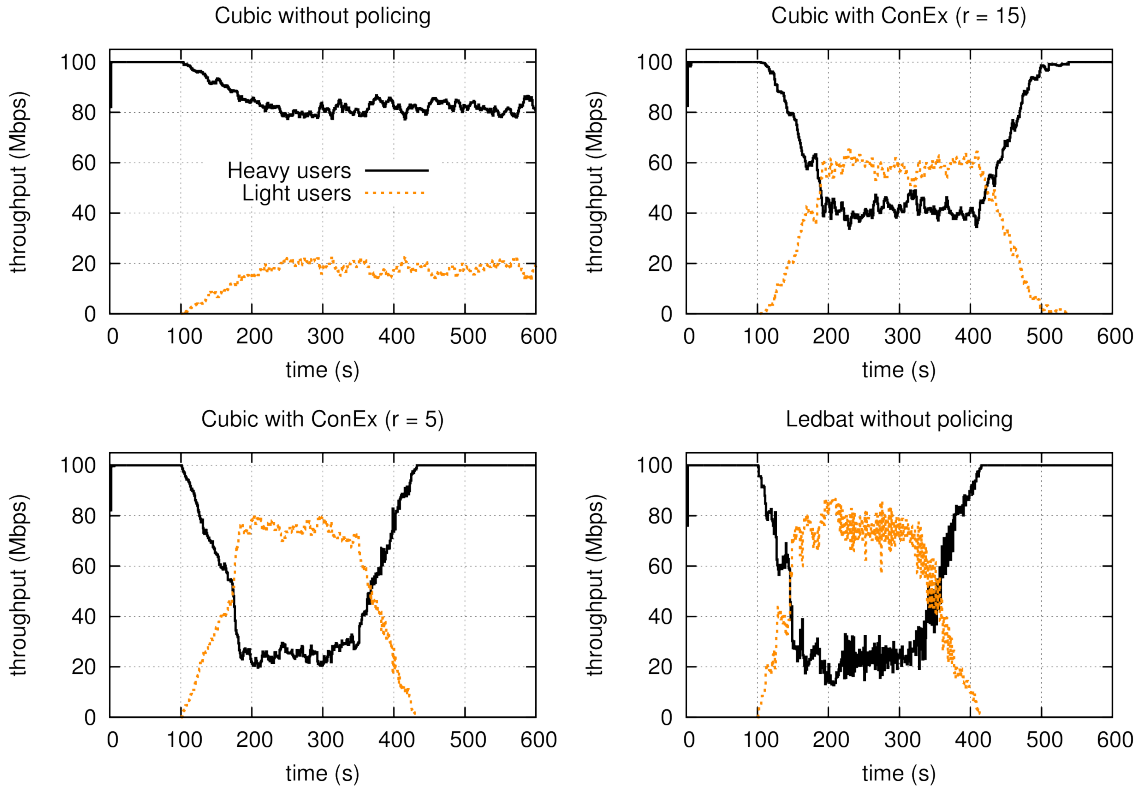

FIGURE 2.19 – Débit des *heavy users* et des *light users* en fonction du temps

ne peuvent pas finir la lecture de la vidéo avant la fin de la deuxième période. Les flux des *light users* sont encore actifs durant la troisième période. Les *light users* voient leur vidéo se figer de nombreuses fois, et pour de longues périodes, allant jusqu'à dix secondes comme nous pouvons le voir sur la figure 2.20. Ceci aboutit à une $QoE = 1.5$, ce qui est la valeur la plus basse que l'on puisse obtenir avec l'équation (2.2). Nous pouvons nous attendre à ce que dans un cas réel, des utilisateurs avec une telle QoE arrêtent de regarder la vidéo au bout de quelques gels.

CUBIC avec policer

Nous activons l'utilisation de ConEx et du policer de congestion pour limiter les *heavy users* et améliorer la QoE des *light users*. La figure 2.20 montre que ConEx arrive parfaitement à atteindre cet objectif : plus le taux de remplissage diminue, plus la QoE des *light users* s'améliore, ceci s'expliquant par la réduction notable du nombre et de la durée des

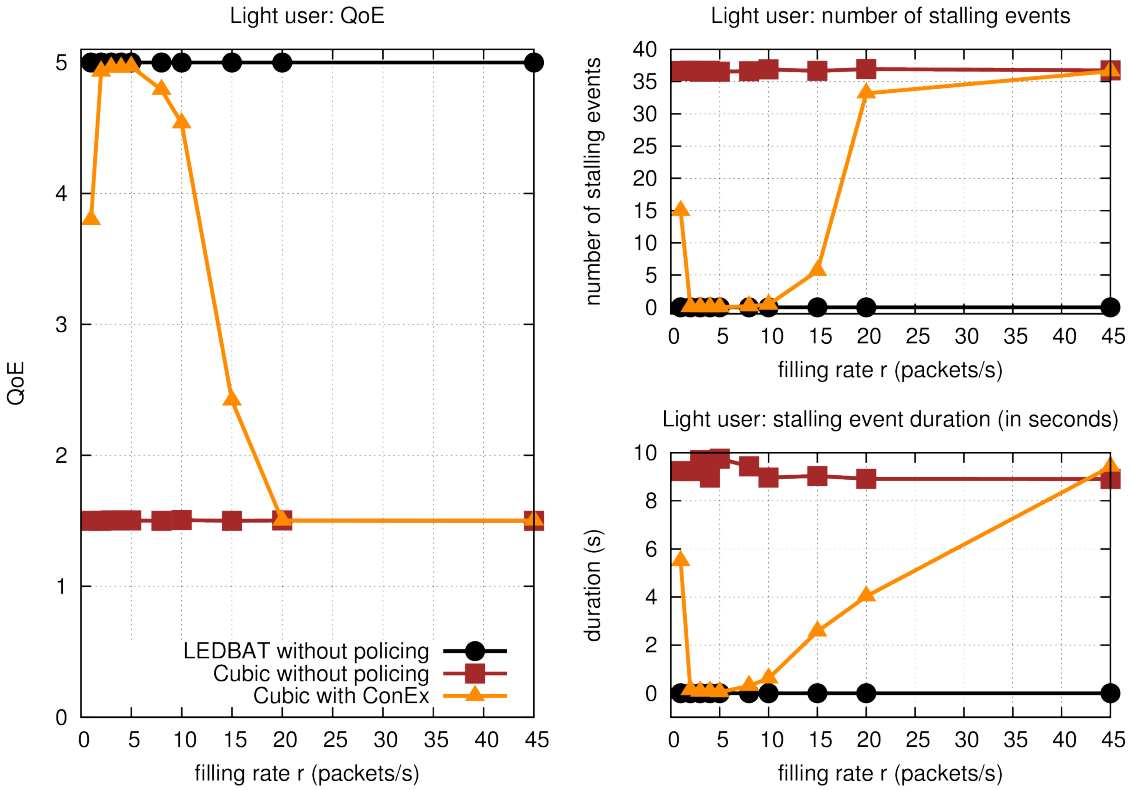

FIGURE 2.20 – QoE des *light users*, le nombre de gels de la vidéo et la durée d'un gel

épisodes de gel de la vidéo. Pour un taux de remplissage $r < 15$ paquets/s, les *light users* bénéficient d'une très bonne qualité d'expérience ($QoE > 4$). Le gain en QoE provient de la réaction des *heavy users* qui sont contraints de résuire leur débit suite à l'intervention du policer de congestion durant la deuxième période de temps comme nous pouvons le voir sur la figure 2.19. Comme représenté pour des taux de remplissage $r = 5$ paquets/s et $r = 15$ paquets/s, le paramétrage du taux de remplissage permet de contrôler la répartition de débit entre les *heavy users* et les *light users*. Quand le policing est actif, les *light users* sont capables de terminer la lecture de la vidéo avant la fin de la deuxième période. Enfin, durant la troisième période, les *heavy users* peuvent augmenter leur débit de nouveau.

LEDBAT sans policer

Les *heavy users* pourraient éviter la réaction du policer de congestion en étant moins agressifs face aux flux vidéo. Ils pourraient soit reporter leurs activités jusqu'à une période

où il y a moins de charge de trafic dans le réseau [34], soit ils pourraient utiliser un algorithme de contrôle de congestion moins agressif, tel que LEDBAT [56], qui libère les ressources du réseau lorsqu'il détecte un risque de congestion. LEDBAT a été conçu pour utiliser le débit disponible sur un lien, et rapidement libérer les ressources en présence d'autres flux considérés comme plus prioritaires, des flux TCP CUBIC par exemple.

Quand LEDBAT est utilisé (implanté sur NS2 par [57]) au lieu de CUBIC par les *heavy users*, les résultats sur la figure 2.19 montrent que, sans avoir besoin de policer, les *heavy users* réduisent rapidement leur débit lorsque les *light users* (qui continuent d'utiliser CUBIC) démarrent leur vidéo. Les *light users* sont alors capables de regarder leur vidéo avec une très bonne QoE (cf. figure 2.20), des résultats similaires à ceux obtenus en utilisant CUBIC avec policer ($r = 5$). Ensuite, lorsque les *light users* finissent leur vidéo, LEDBAT est capable de rapidement reprendre les ressources libérées au niveau du goulot d'étranglement.

Le comportement observé avec LEDBAT pourrait susciter quelques interrogations : où est l'utilité de ConEx ? Pourquoi ne pas passer directement à l'utilisation de LEDBAT ? En fait, l'utilisation d'un algorithme de contrôle de congestion tel que LEDBAT pour le transfert des données pourrait être notre objectif, mais pour favoriser son adoption, il est nécessaire d'avoir un mécanisme incitatif qui récompense les utilisateurs adoptant un tel algorithme. Comme il est suggéré dans la charte de ConEx [58], ConEx peut être déployé pour encourager les *heavy users* à migrer vers des algorithmes de contrôle de congestion similaires à LEDBAT. L'utilisation de ce dernier évite aux *heavy users* de consommer les jetons pour des applications telles que le transfert de fichier en FTP, les préservant pour des applications plus critiques, tout en permettant aux *light users* d'avoir une meilleure qualité d'expérience.

Dans cette étude, nous avons utilisé un mode de diffusion progressif des vidéos (mode dit progressive download) puisque cela nous permettait de bien quantifier les bénéfices de ConEx et de LEDBAT. Si le transfert de la vidéo reposait sur un mode de diffusion adaptif sur HyperText Transfer Protocol (HTTP) (mode dit HTTP adaptive streaming), un comportement proche de celui observé ici peut être attendu : les *light users* diminuent la résolution de leur vidéo en cas de congestion, mais après que les *heavy users* ont réduit leur débit grâce à LEDBAT ou avec l'intervention du policer de congestion, les *light users* peuvent améliorer la résolution de leur vidéo et bénéficier d'une meilleure QoE.

2.5 Résumé et conclusion

ConEx est un mécanisme qui permet à l'utilisateur d'informer l'opérateur du réseau du niveau de congestion qu'il a rencontré sur son chemin. Cela permet à l'opérateur d'implanter, aux heures chargées, des politiques de limitation des utilisateurs proportionnelles au niveau de la congestion auquel cet utilisateur a contribué dans le réseau.

Dans la section 2.2, nous avons vu que ConEx est en mesure de différencier les *light users* et les *heavy users*, et permet d'améliorer l'équité entre utilisateurs. Plusieurs paramètres (paramètres de configuration du policer, RTT, algorithme de contrôle de congestion de TCP, paramètres de configuration de la file RED) sont impliqués dans le mécanisme ConEx et peuvent influencer plus ou moins significativement sa capacité à améliorer l'équité entre utilisateurs.

De façon naturelle, ConEx dépend de la configuration du policer de congestion. En particulier, ConEx est très sensible à la sévérité du policer à cause de son interaction directe avec les signaux Re-Echo-Loss, comme nous l'avons expliqué dans la section 2.2.2. Comme tout mécanisme opérant en boucle fermée entre l'expéditeur et le destinataire, ConEx est très sensible au RTT. La rapidité de la récupération des informations de congestion, au travers d'un court RTT, a en particulier un impact significatif sur la capacité de ConEx à améliorer l'équité entre utilisateurs. Les résultats obtenus avec CUBIC et Compound montrent que ConEx est plutôt peu sensible à l'algorithme de contrôle de congestion de TCP implanté par les machines d'extrémité. Cela veut dire que les utilisateurs devraient recevoir un traitement similaire, quel que soit l'algorithme de contrôle de congestion qu'ils utilisent, rendant ConEx relativement indépendant des types de machines d'extrémité.

Enfin, notre étude du type de file d'attente implanté par l'opérateur dans son réseau (p.ex. RED) montre que les paramètres de la file d'attente ont un impact limité sur le comportement de ConEx, particulièrement en termes d'amélioration de l'équité au travers d'un policer de congestion, comparé à l'impact important qu'ont ces paramètres sur les autres caractéristiques du trafic tels que le délai dans la file. En conséquence, les paramètres de la file RED peuvent être optimisés pour contrôler le délai dans la file sans influer significativement sur les performances de ConEx.

Nous avons également montré que ConEx conserve sa capacité à améliorer l'équité entre utilisateurs même avec une implantation minimale du mécanisme (avec seulement la capacité de réagir à la détection d'une perte en émettant un signal Re-Echo-Loss) et l'utilisation d'une simple file DropTail (cf. section 2.2.7). Un déploiement initial de ConEx avec un minimum de modifications est donc possible, comme le suggère [7], avant de devoir

considérer un déploiement plus complet de ConEx si un niveau de performances plus élevé est requis. Les performances de chaque étape de modifications sont résumées sur le tableau 2.4.

Dans la section 2.3, nous avons mis en évidence les faibles performances de ConEx en présence de flux de courte durée. Ces résultats étaient prévisibles, en effet, ces flux courts n'apportent pas suffisamment d'informations sur la congestion à ConEx, et les limiter au travers d'un policer est le plus souvent contre-productif et peut même s'avérer très préjudiciable à leur durée de session. Il est par conséquent plus intéressant de se concentrer sur les flux plus longs qui sont en mesure de reporter plus précisément le volume de congestion rencontré et qui ont une meilleure réaction face au policer.

Dans la section 2.4, nous avons vu comment des services de diffusion de vidéo comme YouTube peuvent bénéficier de ConEx. Les résultats montrent l'amélioration que nous pouvons obtenir de l'utilisation conjointe de ConEx et de LEDBAT. ConEx peut être utilisé pour limiter les *heavy users* qui ne libèrent pas volontairement les ressources réseau en cas de congestion, préservant ceux qui le font au travers d'un algorithme de contrôle de congestion tel que LEDBAT. Ceci peut motiver les *heavy users* à être plus coopératifs durant les heures chargées de la journée, l'utilisation de LEDBAT évitant aux *heavy users* une réaction du policer de congestion tout en permettant aux *light users* d'avoir une meilleure QoE.

Pour conclure, ConEx peut être considéré comme une approche crédible pour l'amélioration de l'équité entre utilisateurs dans le cas d'une charge réseau importante, tout en étant transparent aux heures non chargées. Si le comportement de ConEx est influencé par son environnement (p.ex., la topologie du réseau, la charge de trafic), dans toutes les configurations étudiées son comportement reste robuste, suggérant que nous avons une marge opérationnelle raisonnable pour que l'opérateur décide de le déployer dans son réseau. Considérant son comportement peu satisfaisant dans le cas des flux courts, et le fait que les flux longs sont la principale source de congestion, ConEx devrait se focaliser sur les flux longs. Afin de minimiser le coût de son implantation dans le réseau, ConEx devrait d'abord être déployé avec le moins de modifications possibles, c.-à.-d. avec la seule modification des sources de trafic (p.ex. les serveurs) pour générer les signaux ConEx. Si de meilleures performances du mécanisme sont nécessaires, en particulier vis-à-vis du délai, l'amélioration de ConEx peut être envisagée, de préférence pour une implantation en $FullConEx$.

CHAPITRE 3

TRAITEMENT DE LA CONGESTION DANS LE DATA CENTER

Avec la généralisation des techniques de virtualisation sur les serveurs et aujourd'hui sur les équipements du réseau, plusieurs applications et services essentiels à l'internet se trouvent au sein des data centers. Les opérateurs offrent des services cloud (p.ex., puissance de calcul, stockage) flexibles et à la demande à travers des Application Programming Interfaces (APIs), et ils doivent assurer au trafic dans le data center une bonne qualité de service. Dans le data center, nous pouvons diviser le trafic en deux catégories : le trafic dit « nord-sud », entre l'extérieur et les serveurs du data center telles des requêtes web, et le trafic dit « est-ouest » entre les serveurs internes au data center. La nature des applications dans le data center comme le stockage distribué et le MapReduce [59] pour le Big Data, a rendu le trafic est-ouest très important ce qui peut mener à de la congestion au sein du réseau [11]. Or comme nous allons le voir dans ce chapitre, l'utilisation des réseaux virtuels, pour l'isolation des *tenants* (les utilisateurs du cloud), au-dessus du réseau physique se traduit par une difficulté dans l'identification des sources de la congestion, ce qui rend la réaction à la congestion compliquée pour les administrateurs du cloud.

Dans ce chapitre, nous nous proposons d'apporter une solution à ce problème. Nous nous fonderons sur les principes de traitement de la congestion que nous avons vus au chapitre précédent appliqués au réseau d'un data center, en prenant en compte les spécificités de cet environnement telles que l'utilisation de réseaux virtuels. Avant de présenter notre solution, nous aborderons d'abord son contexte technique, à savoir celui du cloud dans le data center. Ensuite, nous présenterons notre implantation de la solution dans un contrôleur de cloud open source. Enfin, nous montrerons le fonctionnement de la solution grâce à une plate-

forme expérimentale.

3.1 Architecture Intra-DC

L'architecture historique qui a longtemps été utilisée dans la conception des réseaux de data centers, pour connecter les serveurs d'extrémité servant à fournir les capacités de traitement de données et le stockage, est une architecture en couche comme représenté sur la figure 3.1. Cette architecture compte trois couches qui interconnectent les serveurs entre eux et connectent les serveurs aux réseaux extérieurs : la couche d'accès, la couche d'agrégation et la couche de cœur [1].

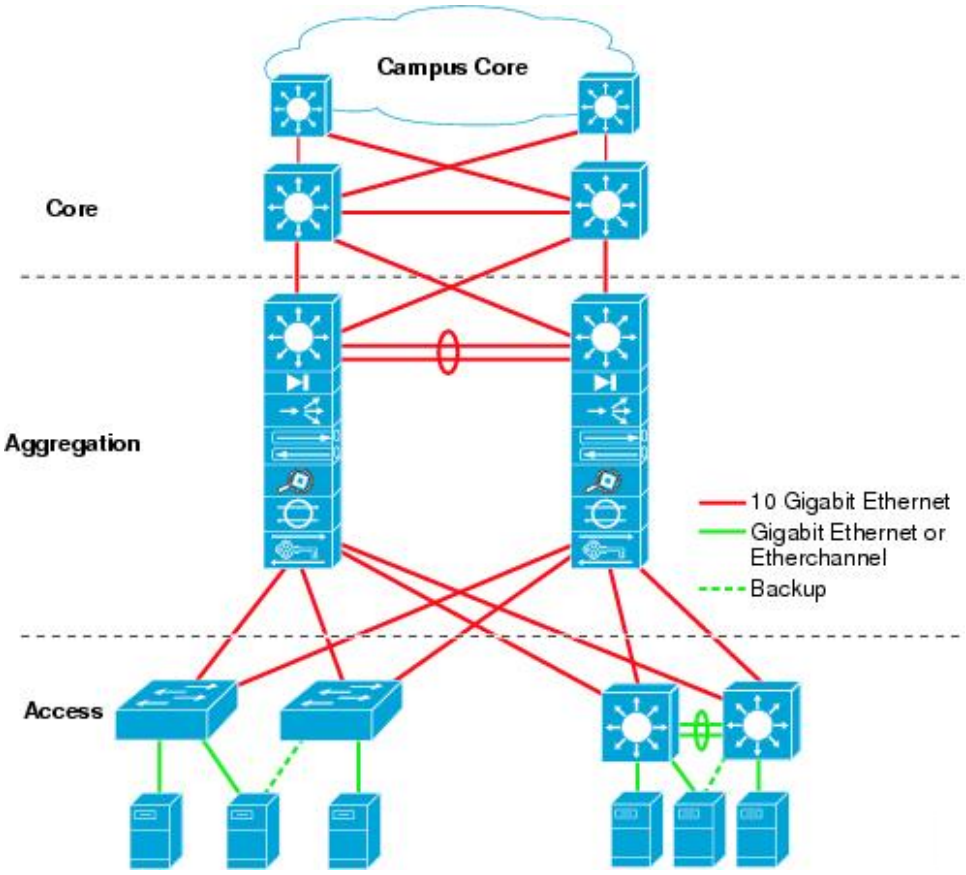

FIGURE 3.1 – Architecture en couche du réseau d'un data center (source [1])

Chacune de ces couches a une ou plusieurs fonctions dans l'architecture :

— La couche d'accès : elle est formée de commutateurs qui offrent essentiellement une

connectivité de niveau 2 (p.ex. Virtual Local Area Network (VLAN)) aux serveurs d'extrémité.

— La couche d'agrégation : elle est formée de commutateurs et de routeurs qui fournissent de la connectivité de niveau 2 entre les commutateurs du réseau d'accès, de niveau 3 entre les serveurs d'extrémité, et également une connectivité entre la couche d'accès et la couche de cœur pour le trafic à destination ou en provenance de l'extérieur du data center. Nous trouvons à ce niveau également un accès aux services d'optimisation et de sécurisation du réseau tels que la répartition de charge, le pare-feu, ou la détection d'intrusion.

— La couche de cœur : elle se charge surtout de la connectivité des machines du data center avec l'extérieur. Les routeurs de la couche de cœur utilisent ainsi des protocoles de routages internes (p.ex., OSPF) et des protocoles de routage externe (Border Gateway Protocol (BGP)).

L'augmentation du trafic à l'intérieur d'un data center et la nécessité de garantir le passage à l'échelle, la disponibilité, et les performances de l'infrastructure ont mené à plusieurs propositions d'améliorations de l'architecture de base en couches. On peut citer l'introduction de la topologie fat-tree [60] pour régler les problèmes de défaillance et de sursouscription des liens, et l'utilisation d'Equal-Cost Multi-Path (ECMP) pour mieux utiliser tous les liens disponibles dans l'infrastructure [61], entre autres propositions.

3.1.1 Le cloud et l'*overlay*

Le cloud computing exploite l'infrastructure du data center pour offrir aux utilisateurs des ressources informatiques (p.ex., puissance de calcul et stockage) et des services (p.ex., applications) à la demande. Le modèle IaaS (Infrastructure-as-a-Service) offre par exemple aux utilisateurs du cloud la possibilité de lancer des machines virtuelles pour le traitement de données et des espaces de stockage, de manière simple au travers d'interfaces de programmation applicatives. La facilité d'utilisation, de déploiement et la haute disponibilité du cloud computing lui ont permis de connaître un très grand essor ces dernières années. Durant cette période, de nombreuses entreprises ont adopté cette approche en investissant sur des infrastructures de cloud privé pour offrir des ressources informatiques à la demande à leurs employés, ou en louant les ressources chez un fournisseur de cloud public tel qu'Orange ou Amazon Web Services (AWS). L'essor du cloud computing se confirme par exemple par l'augmentation significative des revenus d'AWS [62] grâce à ses offres Elastic Compute Cloud (EC2) et Simple Storage Service (S3).

Le cloud computing repose sur la virtualisation des ressources physiques du data center

pour permettre le partage de l'infrastructure entre plusieurs clients du cloud. La virtualisation des environnements de travail, qui consiste en l'utilisation d'un même serveur physique pour lancer plusieurs instances de systèmes d'exploitation isolées au travers d'un hyperviseur, n'est pas une technique nouvelle, mais elle a connu une grande évolution, avec l'intérêt croissant pour le cloud computing, dans le but d'améliorer le partage des ressources matérielles entre les instances virtuelles. Kernel-based Virtual Machine (KVM) [63] par exemple est un hyperviseur intégré au noyau Linux qui permet d'exploiter l'assistance de virtualisation offerte par les processeurs récents pour accélérer l'utilisation du matériel par les instances virtuelles et atteindre des performances proches de celles obtenues à partir d'une exploitation directe des ressources.

Les machines virtuelles utilisées par les clients du cloud étant créées de manière dynamique sur les serveurs physiques selon la demande et la disponibilité des ressources, l'interconnexion des machines virtuelles (Virtual Machines (VMs)) doit alors être réalisée dynamiquement afin de construire les réseaux des clients. Les commutateurs reliant les machines hôtes accueillant les instances des clients doivent donc avoir connaissance des nouvelles machines virtuelles créées pour isoler et transmettre leur trafic sur le réseau physique, mais le passage à l'échelle et l'automatisation d'une telle approche semblent tout de suite complexes à réaliser étant donnée la nature dynamique des machines virtuelles.

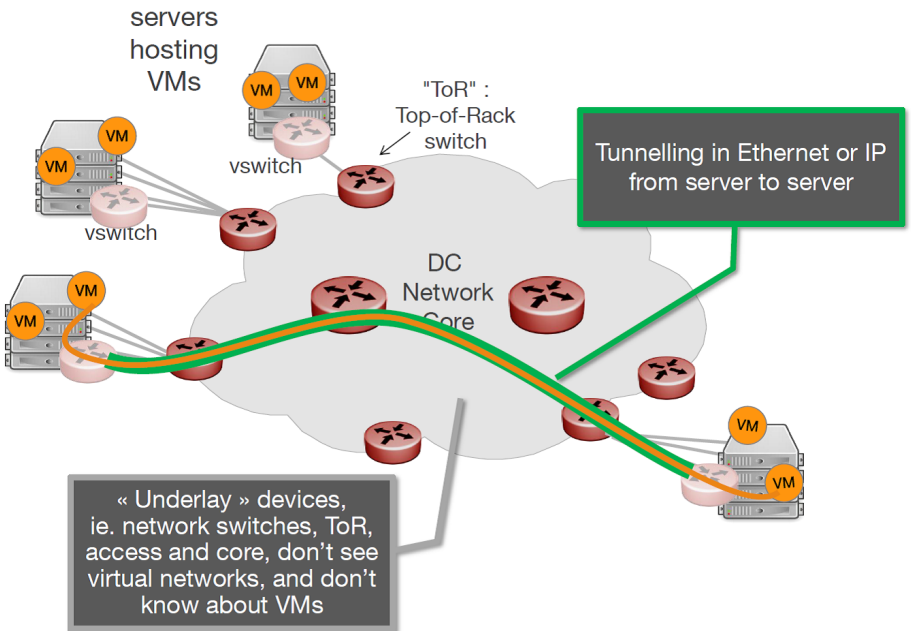

FIGURE 3.2 – Utilisation de l'*overlay* sur l'infrastructure du data center

La solution qui a été proposée pour remédier à ce problème et permettre la création et l'isolation des réseaux des clients sur l'infrastructure physique partagée du data center repose sur l'utilisation des techniques d'encapsulation (p.ex., Virtual Extensible Local Area Network (VxLAN) [64]) pour créer un réseau virtuel du client au-dessus du réseau physique du data center comme représenté sur la figure 3.2. Au-dessus de l'infrastructure réseau physique du data center dite d'« *underlay* », formée par des commutateurs et des routeurs, des tunnels sont créés entre les serveurs d'extrémité pour encapsuler les paquets des utilisateurs et former des réseaux de niveau 2 dit d'« *overlay* » dédiés aux clients. L'encapsulation et la décapsulation des tunnels sont réalisées par des commutateurs virtuels (vSwitchs) installés sur les serveurs hôtes. Les équipements réseau du data center transportent le trafic des tunnels d'un serveur à l'autre sans avoir connaissance des réseaux virtuels (d'*overlay*) des clients et du trafic des machines virtuelles encapsulé dans ces tunnels.

3.1.2 Software Defined Networking

Pour offrir la possibilité au client du cloud de créer et de supprimer dynamiquement ses réseaux d'*overlay*, certains systèmes de déploiement de cloud (comme OpenStack que nous allons présenter plus tard) ont adopté l'approche SDN. Nous rappelons que le SDN est une architecture qui sépare le plan de contrôle, responsable de l'intelligence derrière la prise de décision pour la transmission des paquets dans le réseau, du plan de données qui transmet physiquement les paquets d'après les décisions du plan de contrôle (cf. figure 3.3). Le plan de contrôle peut être alors centralisé dans un contrôleur qui a une vue globale sur le réseau et les ressources disponibles. La communication entre le contrôleur et les équipements se fait au travers d'un protocole de commande tel qu'OpenFlow [65] qui fournit les informations sur la topologie du réseau au contrôleur et qui installe les règles de transmission des paquets sur les tables de transmission des équipements contrôlés. Le contrôleur du réseau offre alors une abstraction du réseau aux applications via des APIs.

Un *tenant* utilise donc les APIs fournies par le contrôleur du cloud pour établir ses réseaux d'*overlay* sur l'infrastructure physique du data center. Il maîtrise ainsi ses réseaux et leur adressage sans nécessiter l'intervention du réseau d'*underlay* qui ne s'occupe que de transmettre les paquets des tunnels entre les serveurs physiques. Mais cette séparation de l'*underlay* avec l'*overlay* crée aussi une brèche dans la qualité de service, qui se trouve fragilisée par l'absence de coordination entre les deux niveaux. Ainsi il est difficile pour les administrateurs de l'infrastructure réseau du data center de pouvoir remédier efficacement à la dégradation de la qualité de service dans le réseau puisque la visibilité sur le trafic des utilisateurs est occultée par la couche d'*overlay*.

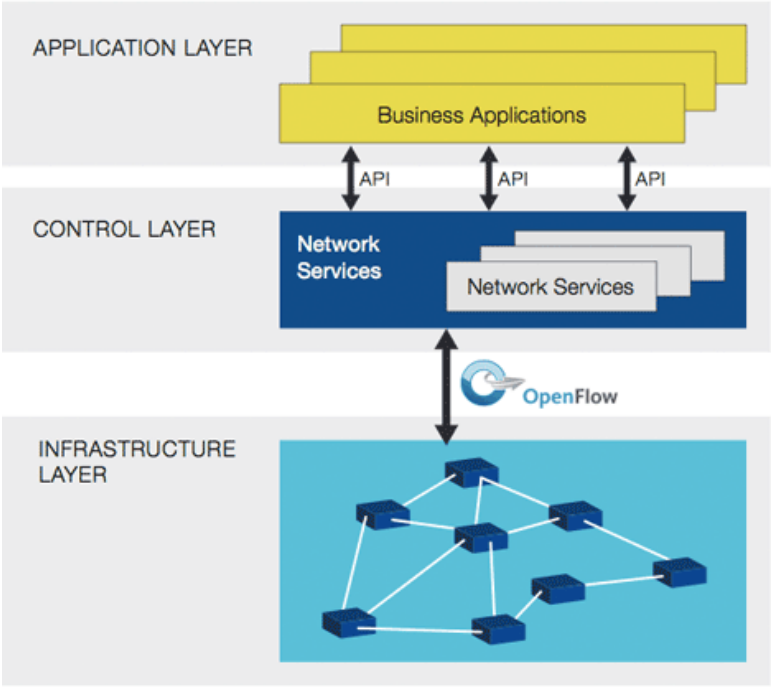

FIGURE 3.3 – Architecture SDN comme définie par l'Open Networking Foundation (source [2])

En effet, au sein d'un data center, le trafic des VMs de certains *tenants*, que nous appellerons « *heavy tenants* », peut engendrer une grande charge sur le réseau et provoquer de la congestion sur les équipements, ce qui peut ensuite mener à la dégradation de la qualité de service des VMs des autres *tenants*, les « *light tenants* », partageant les ressources du cloud. Nous voudrions limiter l'impact à la source du trafic en installant un policer qui limite le débit des machines provoquant le plus de congestion. Mais le réseau d'*underlay* n'a qu'une connaissance des tunnels et non de ce qui y est transporté, il n'est donc possible d'identifier et de limiter que les serveurs hôtes qui hébergent les machines virtuelles dont le trafic mène à de la congestion. Mais ces serveurs peuvent également héberger les VMs d'autres *tenants* dont le trafic est moins significatif. La connaissance précise des VMs se trouve au niveau des réseaux d'*overlay* et des contrôleurs du cloud responsables de leur établissement. Nous proposons donc une solution qui permet au contrôleur de l'*overlay* de prendre en compte l'état de congestion de l'*underlay* pour identifier les VMs responsables du trafic provoquant la congestion dans le réseau du data center et d'appliquer une limitation de débit à celles-ci.

3.2 Solution de traitement de la congestion dans le data center

3.2.1 Architecture de principe

Notre solution de traitement de la congestion dans un réseau de data center repose sur les principes suivants : marquage des flux par les équipements du réseau, comptage des paquets marqués pour chaque utilisateur (*tenant*), et réaction par limitation de débit lorsque l'utilisateur a consommé l'ensemble des ressources qui lui sont accordées.

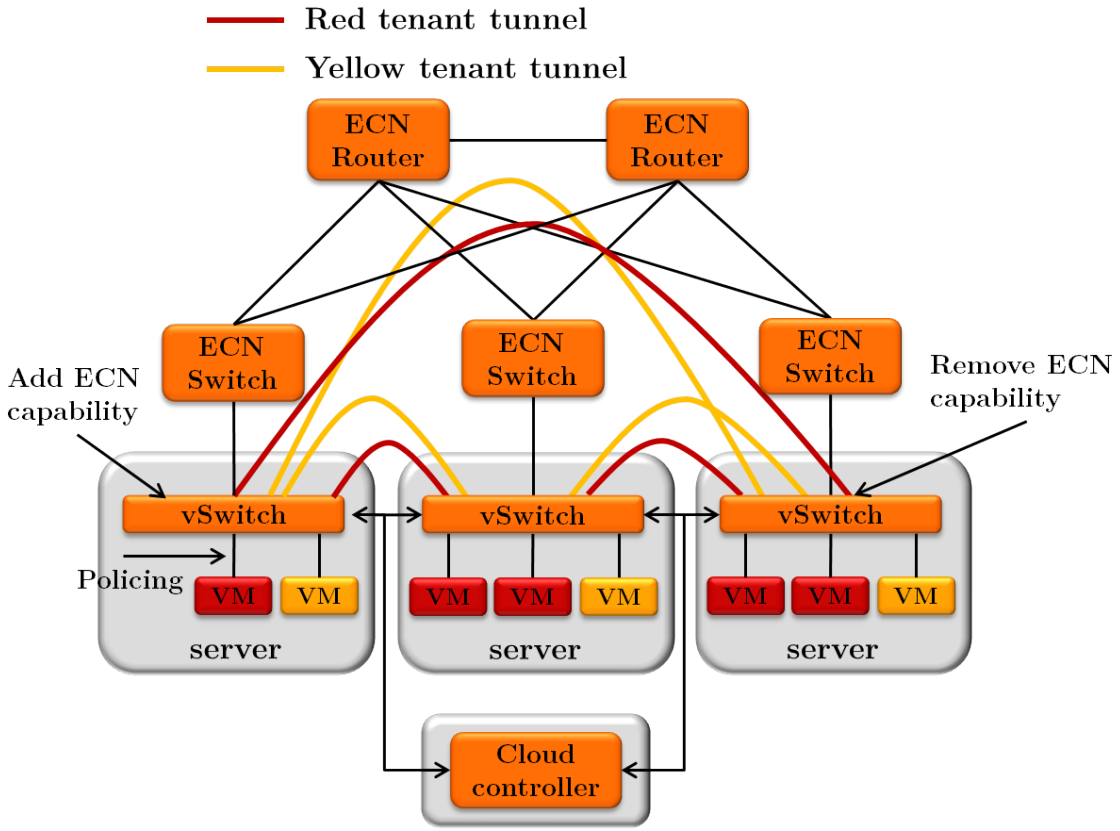

FIGURE 3.4 – Solution de traitement de la congestion dans le réseau d'un data center

Nous pouvons voir sur la figure 3.4 un exemple de réseau de data center mettant en œuvre notre solution. Ici, plusieurs serveurs hébergent les machines virtuelles des *tenants*. Les machines virtuelles sont connectées à des commutateurs virtuels (vSwitch), situés sur les serveurs, qui servent de terminaisons pour les tunnels des *tenants*.

Nous utilisons dans notre solution le marquage ECN [18] comme indicateur de congestion dans le réseau. Ici, les *tenants* n'interviennent pas sur leurs VMs pour négocier l'utilisation d'ECN puisque tout le processus de traitement de la congestion sera géré par le contrôleur du cloud. Grâce aux vSwitchs, le contrôleur du cloud active pour le trafic des VMs l'utilisation du marquage ECN. La capacité de marquage ECN est ajoutée avant l'encapsulation dans les tunnels puis est enlevée à la décapsulation rendant ainsi la solution transparente pour les *tenants*. L'IETF, dans [66], a spécifié comment le marquage ECN devrait être propagé de l'en-tête IP intérieure (des VMs) vers l'en-tête IP extérieure (du tunnel) lors de l'encapsulation et dans l'autre sens lors de la décapsulation.

Les tunnels qui transportent le trafic des VMs traversent l'*underlay* (le réseau physique du data center) dont les équipements (commutateurs et routeurs) sont capables de marquer les paquets du tunnel en ECN au niveau des files d'attente des interfaces de sortie, lorsque celles-ci se remplissent et qu'il y a un risque de congestion et de perte de paquets. Le marquage est réalisé sur l'ensemble des tunnels qui passent par le point de congestion et affecte tous les *tenants* proportionnellement à l'intensité de leur trafic.

Lors de la décapsulation, le marquage des paquets des tunnels est propagé sur les paquets des VMs transportés permettant ainsi une communication entre l'*underlay* et l'*overlay* pour le traitement de la congestion, le premier indiquant au second qu'il y a un risque de congestion auquel il faut remédier. En comptant ces paquets marqués en ECN, nous pouvons identifier les *tenants* ainsi que les VMs qui induisent le plus de congestion dans le réseau et qui donc perturbent l'utilisation du réseau par les autres *tenants*.

3.2.2 Estimation de la congestion

Périodiquement, sur un intervalle Δt, les serveurs qui hébergent les VMs et les vSwitchs remontent le nombre d'octets marqués ECN pour chaque VM source au contrôleur du cloud. Ce dernier garde en mémoire pour chaque *tenant* une liste de ses VMs qui induisent de la congestion et le nombre d'octets N_i marqués pour chaque machine virtuelle i. Ensuite, le contrôleur du cloud calcule sur la période Δt le débit de congestion D_t induit pour chaque *tenant* par ses n machines virtuelles :

$$D_t = \frac{8 \times \sum_{i=1}^{n} N_i}{\Delta t} \tag{3.1}$$

Afin de ne pas fonder la réaction à la congestion sur des fluctuations trop rapides de l'état du réseau, nous utilisons une moyenne de type Exponentially Weighted Moving

Average (EWMA) du débit de congestion pour lisser la valeur de celui-ci sur le temps :

$$\overline{D_t} = \alpha \times D_t + (1 - \alpha) \times \overline{D_{t-1}} \tag{3.2}$$

Avec :

— $\overline{D_t}$ le débit de congestion moyen pour un *tenant*

— D_t le débit de congestion calculé sur l'intervalle Δt

— $\overline{D_{t-1}}$ le débit de congestion moyen calculé lors de la dernière période

— α le facteur de lissage de la moyenne EWMA

Le contrôleur utilise $\overline{D_t}$ pour évaluer le niveau de congestion induit par un *tenant*. Deux seuils sont spécifiés pour chaque *tenant*, un seuil supérieur D_{max} qui correspond au débit maximal de congestion autorisé, et un seuil inférieur D_{min} qui correspond au débit de congestion tolérable pour un *tenant* (cf. figure 3.5).

Lorsque le débit de congestion moyen d'un *tenant* augmente et dépasse le seuil supérieur D_{max}, le contrôleur prend la décision de réagir à la congestion induite par ce *tenant* et installe un policer de débit sur les machines virtuelles du *tenant* limitant leur trafic sur le réseau. Le contrôleur consulte la liste des VMs du *tenant* et commence par limiter la machine virtuelle qui a le plus contribué à la congestion dans le réseau. Lors de la prochaine période de calcul, si le paramètre $\overline{D_t}$ du *tenant* reste supérieur à D_{max}, alors le contrôleur installe un policer sur la seconde machine virtuelle qui induit le plus de congestion et ainsi de suite jusqu'à ce que le débit de congestion descende en dessous de D_{max} ou qu'il n'y ait plus de machines virtuelles qui induisent de la congestion.

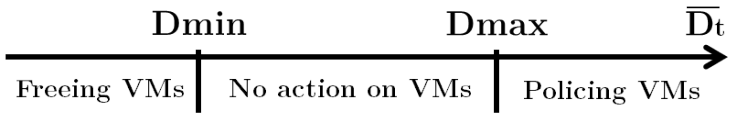

FIGURE 3.5 – Seuils de réaction du contrôleur du cloud

Quand $D_{min} < \overline{D_t} < D_{max}$, le contrôleur arrête de limiter de nouvelles machines virtuelles. Il laisse les policers sur les VMs qui sont déjà limitées mais ne rajoute plus de policers sur le reste des VMs du *tenant*. Quand $\overline{D_t}$ descend sous D_{min}, le contrôleur commence à libérer les machines virtuelles de la limitation de débit en supprimant les policers. À chaque période pour laquelle $\overline{D_t}$ est inférieur à D_{min}, une nouvelle machine virtuelle du *tenant* est libérée jusqu'à ce que toutes les machines virtuelles limitées soient libérées ou que $\overline{D_t}$ passe au-dessus de D_{min}. Ainsi, la solution proposée permet de limiter

le trafic d'un *heavy tenant* en période de congestion pour qu'il ne monopolise pas toutes les ressources du réseau du data center et ne perturbe pas le trafic des autres *light tenants*.

L'utilisation de deux seuils permet d'éviter le phénomène d'oscillation entre la limitation et la libération des machines virtuelles à chaque franchissement du seuil, laissant le temps au débit de congestion de se stabiliser vers une valeur reflétant l'état du réseau.

3.2.3 Contrôle du trafic des utilisateurs

Dans [67], les auteurs proposent une solution de traitement de la congestion dans le data center qui repose également sur la limitation des machines virtuelles selon le niveau de congestion. Ils suggèrent l'installation d'un policer de congestion centralisé par *tenant* qui pourrait alimenter en jetons d'autres policers de congestion distribués sur les serveurs du data center. Les jetons seraient consommés par les signaux ConEx envoyés par les VMs du *tenant* sur le policer. Dans notre environnement géré par un contrôleur de cloud, l'implication des VMs dans le traitement de la congestion n'est pas nécessaire, ce qui nous évite le recours à un auditeur pour vérifier la conformité de la déclaration des VMs par rapport à l'état du réseau. La distribution des jetons vers les policers distribués n'est pas nécessaire dans notre cas, et il n'y a donc pas besoin d'un algorithme d'allocation de ressources entre les policers. Dans notre solution, les paquets marqués sont remontés vers le contrôleur qui évalue la congestion induite par un *tenant* dans le réseau, et n'installe des policers que sur les VMs qui lui appartiennent en cas du dépassement du seuil configuré.

Enfin, il faut préciser qu'il est possible d'appliquer cette solution de traitement de la congestion individuellement pour chaque machine virtuelle au lieu d'avoir des seuils par *tenant*, ce qui reviendrait à appliquer les calculs que nous avons présentés plus haut comme s'il s'agissait d'un *tenant* possédant une seule et unique machine virtuelle ($n = 1$). Mais, dans ce cas-là, dans l'objectif de remettre les compteurs à zéro, le *tenant* pourrait avoir la possibilité de lancer de nouvelles VMs si celles-ci se retrouvent limitées par un policer, tandis qu'avec notre solution dans laquelle le *tenant* a un débit de congestion global partagé entre les VMs, le remplacement d'une VM après sa limitation ne remet pas à zéro le débit global.

Dans la prochaine section, nous allons présenter l'implantation que nous avons faite de notre solution dans un contrôleur de cloud open source (OpenStack). Nous allons ensuite présenter la plate-forme expérimentale que nous avons réalisée pour illustrer le comportement de notre solution et montrer comment un opérateur de cloud pourrait l'utiliser pour contrôler la charge du réseau d'interconnexion de son data center.

3.3 Implantation de la solution

3.3.1 Description de l'environnement OpenStack

OpenStack [3] est un système d'exploitation pour le cloud computing utilisé pour déployer et contrôler les infrastructures de cloud. Il fournit l'accès aux ressources de calcul, de stockage et au réseau (cf. figure 3.6) sur une infrastructure de data center. Il utilise une architecture modulaire dont le code d'implantation est en open source. Chaque module est responsable du contrôle d'une partie de l'infrastructure (p.ex. machines virtuelles, réseaux d'*overlay*) et expose une API qui permet une configuration et une gestion des ressources par les *tenants* du cloud, par son administrateur ou par d'autres modules. OpenStack autorise le déploiement d'une infrastructure cloud avec un partitionnement plus ou moins fin des ressources, selon le besoin de l'utilisateur, avec un partage en agrégats d'hôtes, en cellules, en zones de disponibilité ou en régions pour permettre l'accès aux ressources sur des data centers distants par exemple.

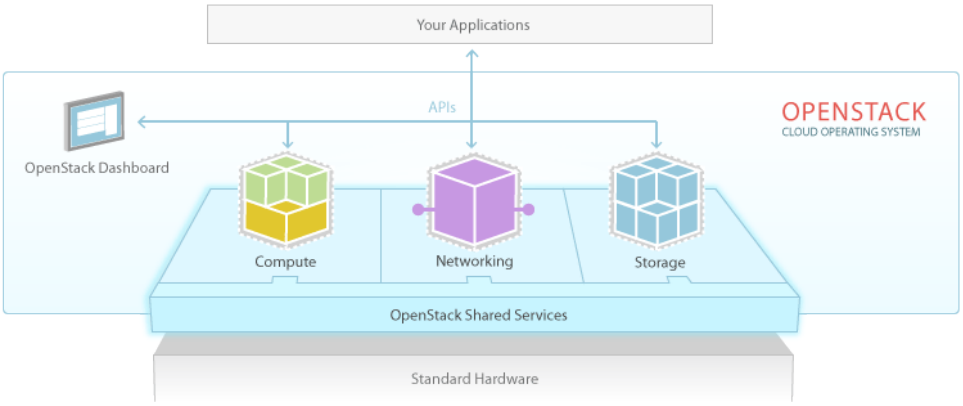

FIGURE 3.6 – Diagramme représentant la plate-forme OpenStack (source [3])

Les modules qui nous seront les plus utiles dans notre implantation de la solution de traitement de la congestion et dans le déploiement de la plate-forme expérimentale sont le module Nova qui permet à l'utilisateur de lancer et de gérer ses ressources de calcul (VMs), et bien évidemment le module Neutron qui fournit les services réseau et l'interconnexion pour les machines du cloud. Nous allons présenter ces deux modules dans la suite.

Nova

Nova est un des modules principaux d'OpenStack. Il est en charge de mettre à disposition et de contrôler de nombreuses ressources de traitement de données comme des machines virtuelles par exemple. Nova fonctionne avec un large choix de technologies de virtualisation. Il s'appuie sur des hyperviseurs tels que KVM ou Xen pour lancer des machines virtuelles sur les serveurs hôtes, ainsi que sur Linux Containers (LXC) ou Docker pour déployer des containers Linux. Ces derniers sont une technologie d'isolation d'environnements entre plusieurs utilisateurs sur un même noyau Linux, tandis que les machines virtuelles ont chacune leur propre noyau. Nova est séparé en plusieurs composants (cf. figure 3.7), les plus importants sont : Nova-API sur le nœud de contrôle du cloud qui répond aux requêtes de gestion des machines virtuelles des *tenants*, et Nova-compute sur les serveurs hôtes qui pilote les hyperviseurs et mécanismes d'isolation de l'environnement (p.ex., KVM, LXC).

En plus de lancer les VMs et autres containers, Nova permet de relier les ports virtuels des machines et des containers aux vSwitchs contrôlés par Neutron qui assurent leur interconnexion au réseau d'*overlay* auquel elles appartiennent. La communication entre les composants d'un même module, par exemple entre Nova-API et Nova-Compute, s'effectue au travers d'appels Remote Procedure Call (RPC), tandis que la communication entre les différents modules, par exemple entre Nova et Neutron, s'effectue au travers de leurs APIs respectives.

Neutron

Neutron est le module d'OpenStack qui est en charge de l'interconnexion des machines virtuelles, du déploiement des réseaux d'*overlay* et des tunnels entre les serveurs hôtes, et offre plusieurs services réseau tels que le Dynamic Host Configuration Protocol (DHCP), le routage, et le pare-feu entre autres. Plusieurs technologies d'isolation des réseaux des *tenants* sont utilisées sur l'infrastructure du data center comme la séparation classique en VLANs et le déploiement d'*overlay* avec des VxLANs. L'adressage IP des réseaux d'*overlay* est géré statiquement ou au travers d'un DHCP déployé spécifiquement pour répondre aux requêtes des machines virtuelles dans ce réseau.

Pour connecter les machines virtuelles aux interfaces physiques du réseau du data center, Neutron utilise des vSwitchs qui se situent sur les serveurs hôtes. Une implantation de vSwitch très utilisée avec Neutron est Open vSwitch (OVS) [68]. OVS implante la plupart des protocoles de contrôle, de gestion et d'isolation du réseau, et permet l'installation de

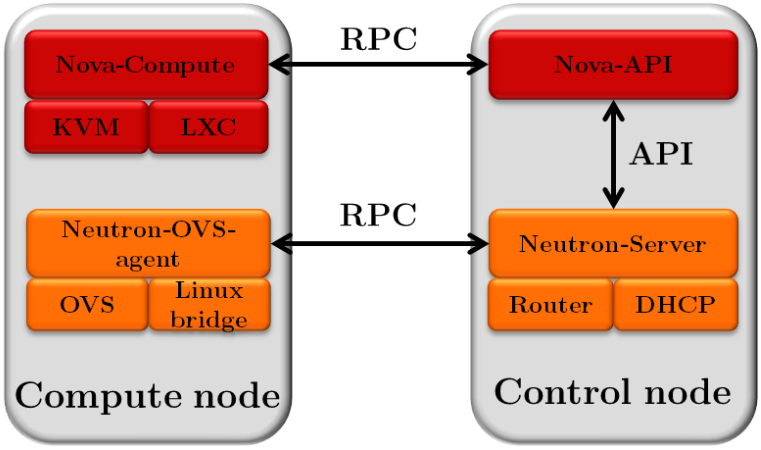

FIGURE 3.7 – Quelques éléments de base du déploiement d'OpenStack

règles de transfert des paquets au travers de flux de commandes de type OpenFlow pour aiguiller le trafic entre les interfaces du vSwitch commandé.

Neutron est lui-même divisé en plusieurs composants, citons principalement le serveur et les agents. Le serveur est le composant central de Neutron. Il se situe sur le contrôleur du cloud. Il est responsable entre autres du traitement des requêtes venant de l'API, de la gestion de la base de données, ainsi que de la communication avec les agents. Ces derniers se situent sur les serveurs hôtes et contrôlent les vSwitchs qui s'y trouvent. Les agents traitent les commandes de contrôle qui leurs sont envoyées par le serveur pour par exemple relier les machines virtuelles aux vSwitchs ou installer les règles de transfert qui permettront aux VMs d'être connectées à leur réseau d'*overlay*.

QoS

Depuis la version Liberty d'OpenStack (octobre 2015), un service de QoS a été ajouté à Neutron qui a pour but d'apporter les règles de qualité de service aux réseaux d'OpenStack dont ils manquaient. Pour être cohérents avec la décomposition de Neutron, nous avons implanté notre solution de traitement de la congestion au sein de l'extension QoS. C'est pourquoi nous allons d'abord donner un bref aperçu de cette extension avant d'expliquer notre implantation.

Le service QoS se décompose en deux éléments : le plugin QoS qui se situe sur le contrôleur du cloud aux côtés du serveur Neutron, et l'extension QoS à l'agent Neutron qui se trouve sur les serveurs hôtes. Le plugin possède sa propre API pour répondre aux requêtes de QoS, il transmet ensuite les commandes à l'extension de l'agent pour mettre

en place les règles de QoS.

Le modèle de QoS dans Neutron est composé de politiques et de règles. Les politiques sont associées soit à une VM en affectant son port à la politique, soit sur un réseau, ce qui revient à affecter tous les ports des VMs du réseau à la politique. Une politique est simplement une liste de règles de QoS qui sont à appliquer sur les ports qui lui sont associés. Deux règles ont été implantées jusqu'ici pour le service QoS : la limitation de débit par un policer et le marquage DSCP du trafic. La règle de « limitation de débit » que nous allons utiliser dans notre solution de traitement de la congestion se présente ainsi : à la création de la règle, deux paramètres sont demandés pour le policer, max-kbps qui est le débit maximal autorisé et max-burst qui est la taille de burst maximale autorisée au-delà de max-kbps. Lorsque la règle est créée avec ces paramètres au sein d'une politique, tous les ports associés à la politique sont limités par un policer.

3.3.2 Solution de traitement de la congestion dans Neutron

L'implantation de notre solution de traitement de la congestion dans Neutron consiste en l'ajout d'une nouvelle règle de « limitation de congestion » au service QoS qui sera appliquée aux réseaux des *tenants* du cloud. Cette règle demande les paramètres suivants :

- max-cong-kbps : le débit maximal de congestion (en kbit/s) autorisé, ce qui correspond à D_{max} dans la section 3.2.
- min-cong-kbps : le débit tolérable de congestion (en kbit/s) D_{min}
- smoothing-factor : le facteur de lissage α pour le calcul de la moyenne EWMA
- interval-s : l'intervalle Δt pour remonter les paquets marqués et pour le calcul de la moyenne EWMA
- max-kbps : le débit maximal autorisé par le policer à installer en cas de dépassement du seuil D_{max}
- max-burst : la taille de burst maximale du policer

La figure 3.8 montre le déploiement du réseau d'OpenStack (avec l'utilisation d'Open vSwitch). Nova démarre les machines virtuelles et gère un commutateur Linux qbr qui sert à installer des règles de pare-feu pour les VMs. Ensuite, le trafic des VMs traverse deux Open vSwitchs br-int et br-tun gérés par l'agent Neutron sur les serveurs hôtes. Le premier permet l'installation des règles de QoS pour les VMs et le second permet d'envoyer les paquets des VMs vers leurs tunnels VxLAN avec l'identifiant de tunnel adéquat. Les tunnels traversent alors le réseau du data center où le marquage en ECN est activé.

La règle de limitation de congestion qui est créée grâce à l'API du plugin QoS dans Neutron est envoyée vers les agents Neutron sur les serveurs hôtes. L'agent installe les

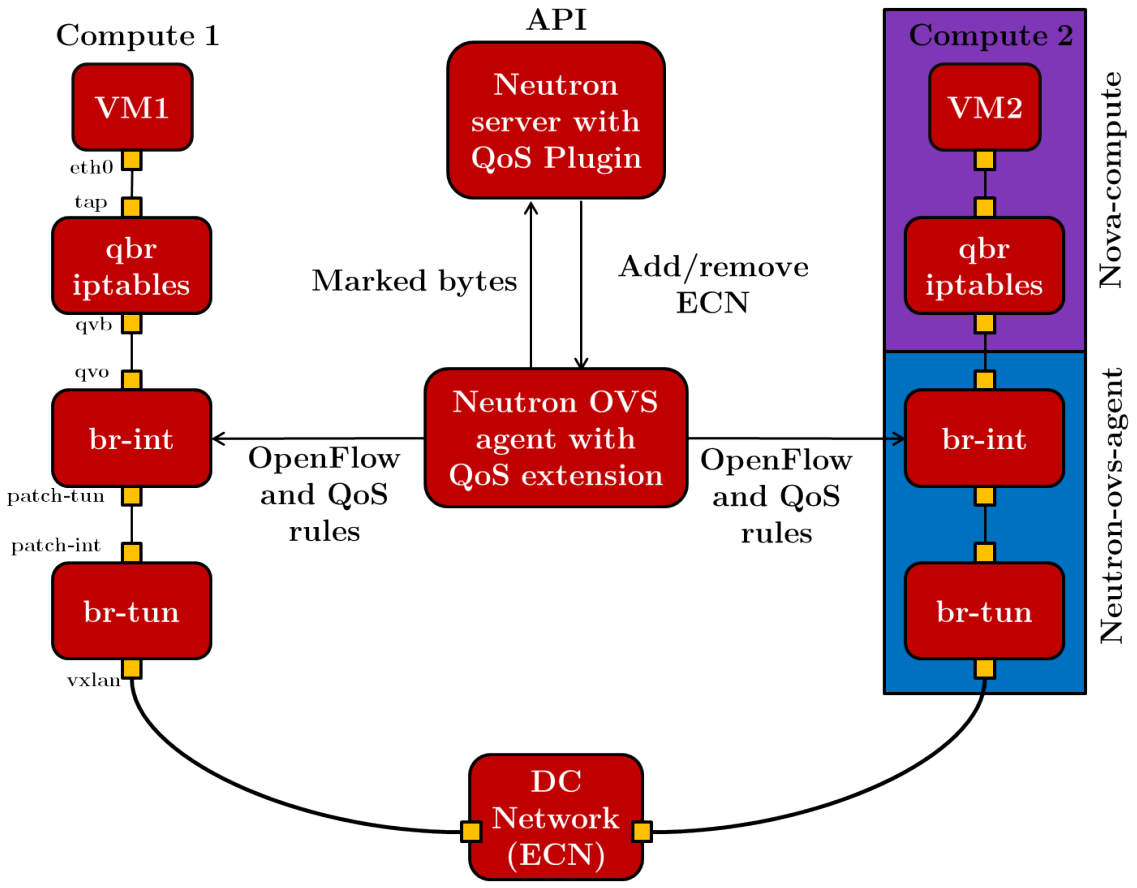

FIGURE 3.8 – Déploiement réseau d'OpenStack sur les serveurs hôtes

règles OpenFlow sur le br-int pour ajouter et enlever le marquage ECN, puis envoie périodiquement le nombre d'octets marqués au plugin QoS du serveur Neutron. Ce dernier calcule le débit de congestion du *tenant* comme présenté dans la section 3.2 et le compare aux paramètres de la règle de limitation de congestion. Lorsque le seuil D_{max} est dépassé et que le serveur prend la décision de limiter une VM d'un client, il crée une règle de limitation de débit avec les paramètres max-kbps et max-burst-kbps de la règle de limitation de congestion, et associe la VM désignée à la nouvelle règle créée, ce qui installe un policer sur son port attaché au vSwitch br-int. Si une règle de limitation de débit existe déjà avec les paramètres spécifiés, la VM est ajoutée dans la liste des VMs associées à cette règle. Lorsque le débit de congestion descend en dessous de D_{min} et qu'il faut enlever le policer d'une VM, il suffit de la dissocier de la règle de limitation de débit à laquelle elle a été associée.

Nous allons maintenant présenter une plate-forme expériementale qui illustre le fonctionnement de notre solution de traitement de la congestion implantée dans OpenStack.

3.4 Plate-forme expérimentale

3.4.1 Présentation de la plate-forme

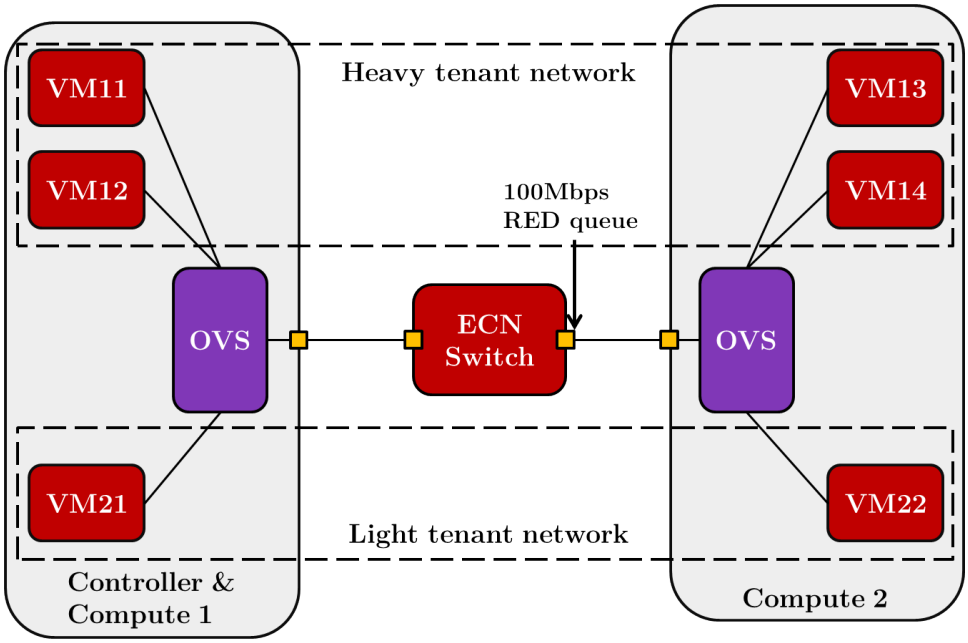

FIGURE 3.9 – Plate-forme expérimentale OpenStack

La figure 3.9 représente la plate-forme expérimentale que nous avons déployée. Elle est composée de deux serveurs et d'un commutateur. Un cloud est déployé sur la plate-forme grâce à OpenStack (version Liberty) avec un serveur qui joue le rôle de contrôleur ainsi que de machine hôte et l'autre serveur qui sert de seconde machine hôte. Le déploiement de Neutron inclut le service QoS (plugin côté serveur et extension côté agent OVS) que nous avons modifié pour l'ajout de notre règle de « limitation de congestion ». Sur le cloud, nous avons deux *tenants* : un *heavy tenant* qui a un réseau composé de 4 VMs, et un *light tenant* qui a un réseau composé de deux VMs. Les VMs du premier serveur hôte communiquent une à une avec les VMs du second serveur hôte en passant par un commutateur avec un lien de sortie bridé à 100Mbit/s. Ce lien est capable de marquer les paquets en ECN sur une file d'attente RED qui peut contenir 10Mbit/s de trafic avec une probabilité de marquage

augmentant de 0% à 100% lorsque la taille de la file augmente de 1Mbit/s à 10Mbit/s.

Sur cette plate-forme expérimentale, nous voulons montrer deux utilisations possibles de notre solution de traitement de la congestion. Dans un premier temps, les deux *tenants* n'ont pas les mêmes débits et n'induisent pas le même niveau de congestion. En revanche, ils ont les mêmes seuils de débit maximum de congestion, ils ont donc le même traitement par rapport au niveau de congestion qu'ils induisent sur le commutateur, et nous voudrions montrer comment le contrôleur limite le *heavy tenant* pour éviter la congestion et rétablir l'équité entre les deux *tenants*. Nous verrons dans ce cas comment certains paramètres influencent le comportement du contrôleur vis-à-vis de la congestion. Dans un second temps, les deux *tenants* ont sensiblement les mêmes débits et induisent le même niveau de congestion. En revanche, ils n'ont pas les mêmes seuils de débit maximum de congestion, et nous voudrions voir comment le contrôleur effectue un traitement différencié entre les *tenants*.

3.4.2 Résultats

Réaction au *heavy tenant*

Les deux VMs du *heavy tenant* sur le premier serveur hôte envoient chacune du trafic d'un débit de 50Mbit/s vers deux autres machines virtuelles situées sur le second serveur hôte. La VM du *light tenant* sur le premier serveur hôte n'envoie en revanche que 10Mbit/s de trafic à la VM sur le second serveur. Cela fait un total de 110Mbit/s de trafic sur les liens du commutateur qui sont à 100Mbit/s. Chaque expérimentation dure 500s. Les paramètres des règles de limitation de congestion appliquées aux deux *tenants* sont identiques et sont les suivants :

— max-cong-kbps = 10Mbit/s
— min-cong-kbps = 5Mbit/s
— smoothing-factor = 0.1
— interval-s = 20s
— max-kbps = 10Mbit/s
— max-burst = 10Mb

Sur la figure 3.10, nous pouvons voir en haut l'évolution du débit du heavy et du *light tenant*, et en bas l'évolution de leur débit moyen de congestion. Au début de l'expérimentation, aucun *tenant* n'est limité, le trafic atteint le débit maximum possible sur la plate-forme (qui est inférieur à 100Mbit/s en raison des performances matérielles de la plate-forme), et la file d'attente RED commence à marquer des paquets.

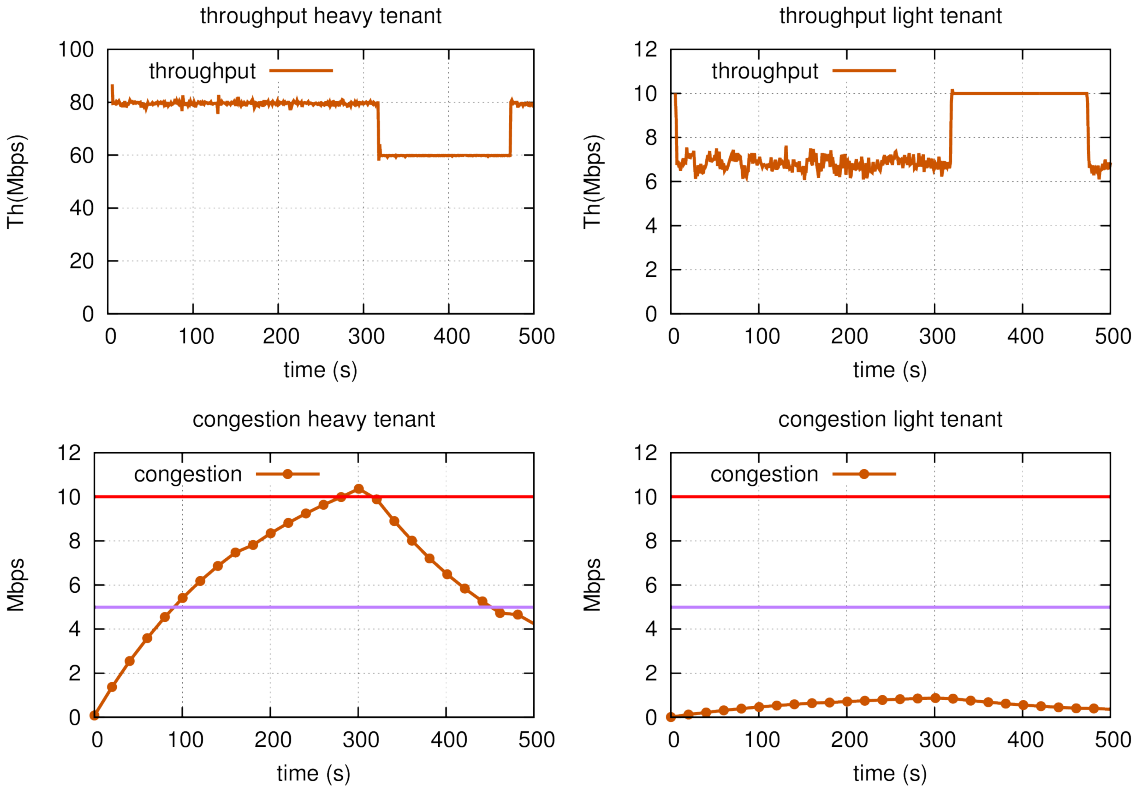

FIGURE 3.10 – Débit et congestion des *tenants*

En bas de la figure 3.10, les lignes rouge et pourpre représentent max-cong-kbps et min-cong-kbps respectivement. Nous voyons que le débit moyen de congestion du *heavy tenant* augmente. Lorsqu'il dépasse le débit maximum de congestion, une de ses VMs est limitée par un policer à 10Mbit/s, son débit total devient alors égal à 60Mbit/s (50Mbit/s de la VM non limitée + 10Mbit/s de la VM limitée). Le commutateur n'est plus en congestion, et la file d'attente ne marque plus les paquets, son débit moyen de congestion diminue alors et il n'y a pas besoin de limiter l'autre VM. En même temps, le *light tenant*, dont la contribution à la congestion dans le réseau est très basse, largement inférieure à D_{max}, récupère du débit sur les liens du commutateur suite à la limitation du *heavy tenant*. Le débit de sa VM atteint 10Mbit/s qui est à son maximum. Enfin, lorsque le débit de congestion du *heavy tenant* descend en dessous de min-cong-kbps, la VM qui était limitée est libérée, et le débit du *light tenant* redescend à son niveau départ.

Ici, nous voyons que le *light tenant* n'est plus perturbé par le *heavy tenant* grâce au contrôleur du cloud qui réagit à la congestion qui se trouve dans le réseau. La réaction

dans ce cas survient relativement tardivement du fait de l'évolution du débit moyen de congestion du *heavy tenant* qui est dépendant de deux paramètres : le facteur de lissage α et la période de calcul Δt. Alors que cette dernière valeur peut être dépendante de la plate-forme et de la communication entre le contrôleur et les agents sur les serveurs hôtes, donc potentiellement difficile à maîtriser, le facteur α est lui configurable de façon à obtenir une réaction plus ou moins rapide à la congestion.

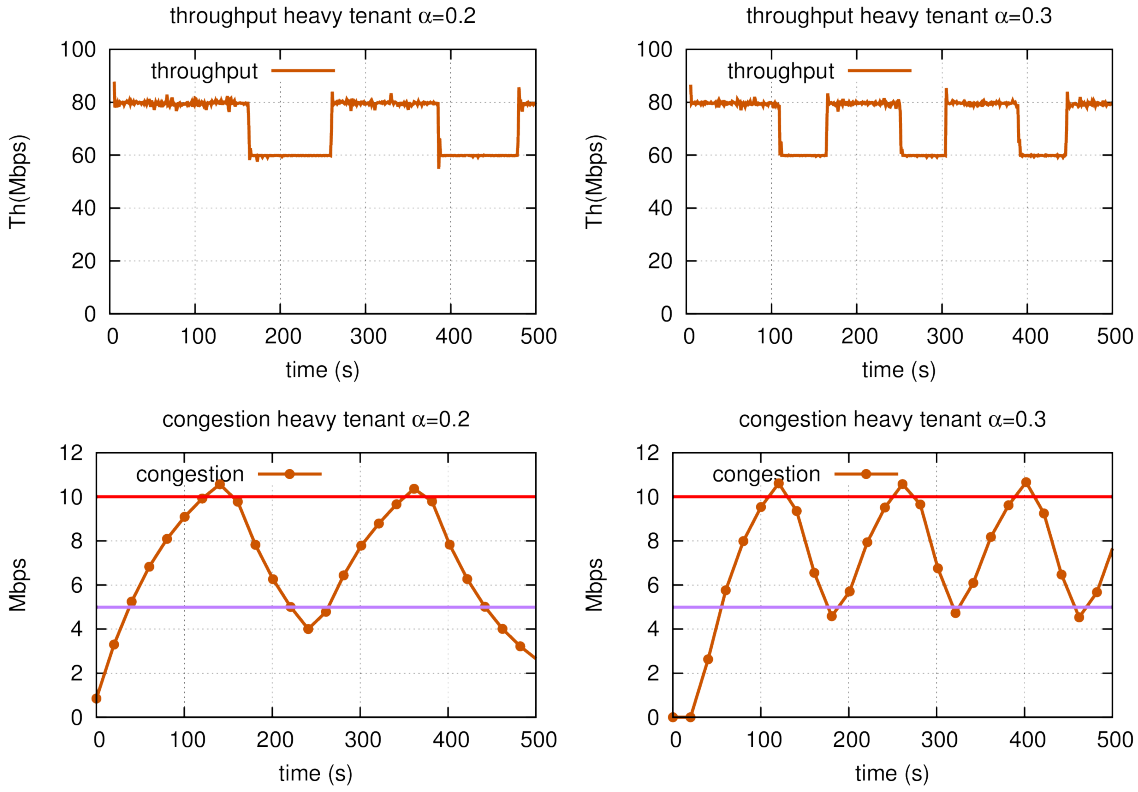

FIGURE 3.11 – Débit et congestion du *heavy tenant* pour α égal à 0.2 et 0.3

Sur la figure 3.11, nous pouvons voir le débit et l'évolution de la congestion du *heavy tenant* pour $\alpha = 0.2$ à gauche de la figure et $\alpha = 0.3$ à droite. On constate que le débit moyen de congestion varie plus rapidement pour les valeurs élevées de α : il augmente plus rapidement en présence de congestion, la machine virtuelle du *tenant* étant limitée plus vite, mais diminue aussi plus rapidement quand celle-ci disparaît, la machine virtuelle du *tenant* étant libérée plus vite. Sur la durée de l'expérimentation, le débit de congestion du *heavy tenant* oscille entre les deux seuils de congestion, limitant et libérant la VM successivement, car la valeur de α, la nature du trafic utilisé et le nombre réduit de machines virtuelles et de

tenants font que le réseau bascule périodiquement entre deux états. Dans un environnement plus complexe, comportant un nombre significativement plus important de flux de trafic, le débit de congestion évoluera selon l'état du réseau et l'activité des différents *tenants* se partageant le cloud. Ainsi, la moyenne du débit de congestion d'un *heavy tenant* ne devrait pas montrer des variations aussi rapides, restant au-dessus du seuil minimal tant que les files d'attente du réseau du data center marqueront les paquets, et les VMs ne seront libérées qu'une fois la congestion disparue.

La bonne valeur de α est liée à l'intensité de la congestion qui est constatée lors de la collecte des informations sur l'état du réseau. Sur notre plate-forme, α est configuré manuellement lors de la création de la règle de limitation de congestion, mais il serait plus judicieux et plus simple pour les utilisateurs de la règle d'avoir une configuration automatique où le facteur α s'adapterait à l'intensité de la congestion par rapport aux seuils de congestion du *tenant*.

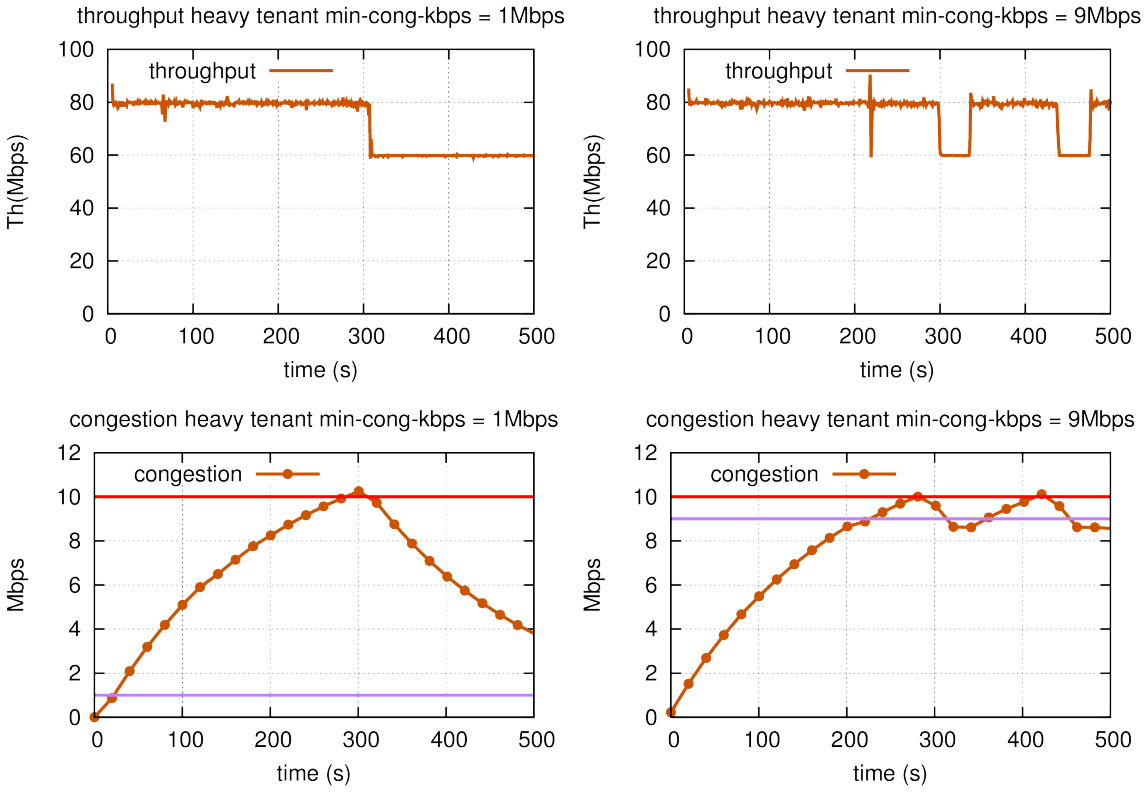

FIGURE 3.12 – Débit et congestion du *heavy tenant* pour min-cong-kbps égal à 1Mbit/s et 9Mbit/s

Le seuil de congestion minimum détermine le moment auquel le contrôleur libère les VMs qui sont limitées par un policer : il détermine quand le contrôleur peut considérer que la congestion moyenne est assez basse, donc que le réseau ne risque plus de congestion. La figure 3.12 représente le débit et l'évolution de la congestion du *heavy user* lorsque le seuil minimal est configuré à 9Mbit/s et à 1Mbit/s. Dans le premier cas, le seuil bas est très proche du seuil haut. Très rapidement après la limitation de la VM du *heavy tenant*, celle-ci est libérée et la congestion remonte, et ainsi de suite. Dans ce cas-là, il n'y a pas suffisamment d'espace entre le seuil haut et le seuil bas pour que la congestion dans le réseau disparaisse avant de libérer les VMs ou se stabilise autour d'une valeur pour laquelle le contrôleur n'aurait besoin ni de limiter davantage de VMs de ce *heavy tenant*, ni d'en libérer.

Avec min-cong-kbps égal à 1Mbit/s, le seuil est très bas, et l'expérimentation se termine avant que la VM qui a été limitée ne soit libérée. Ici, le contrôleur est très conservateur, il ne libère les VMs que lorsque la congestion moyenne est très basse, ce qui serait un indicateur de la disparition de la congestion dans le réseau. Dans ce cas-là, l'inconvénient serait d'avoir une libération trop lente des VMs, tandis que le réseau ne risque plus de congestion.

En conclusion des résultats présentés ici, il est recommandé d'avoir un seuil de congestion très bas, et d'avoir un facteur de lissage α adapté pour avoir une diminution rapide de la moyenne de la congestion lorsque l'intensité instantanée de la congestion remontée par les agents est nulle ou proche de zéro.

Réaction différenciée entre les *tenants*

Nous voulons ensuite montrer comment notre solution de traitement de la congestion peut apporter un traitement différencié des *tenants* selon leur contrat avec l'administrateur du cloud. Ici, les VMs du *heavy tenant* ont un débit global de 70Mbit/s, tandis que la VM du *light tenant* a un débit plus faible, à 40Mbit/s. Le débit total sur la plate-forme est donc de 110Mbit/s. La congestion dont est responsable le *light tenant* est plus importante que dans les expérimentations précédentes, mais elle reste moins importante que celle du *heavy tenant*. En revanche, les *tenants* ont des valeurs différentes de seuil maximal de débit de congestion max-cong-kbps : 10Mbit/s pour le *heavy tenant*, et 5Mbit/s pour le *light tenant*.

Le débit et l'évolution de la congestion du *light* et du *heavy tenant* sont représentés sur la figure 3.13. Étant donné que la contribution à la congestion du *heavy tenant* est la plus importante, son débit moyen de congestion augmente plus vite que celui du *light tenant*. Toutefois, le seuil maximum de ce dernier étant plus bas, son débit moyen de

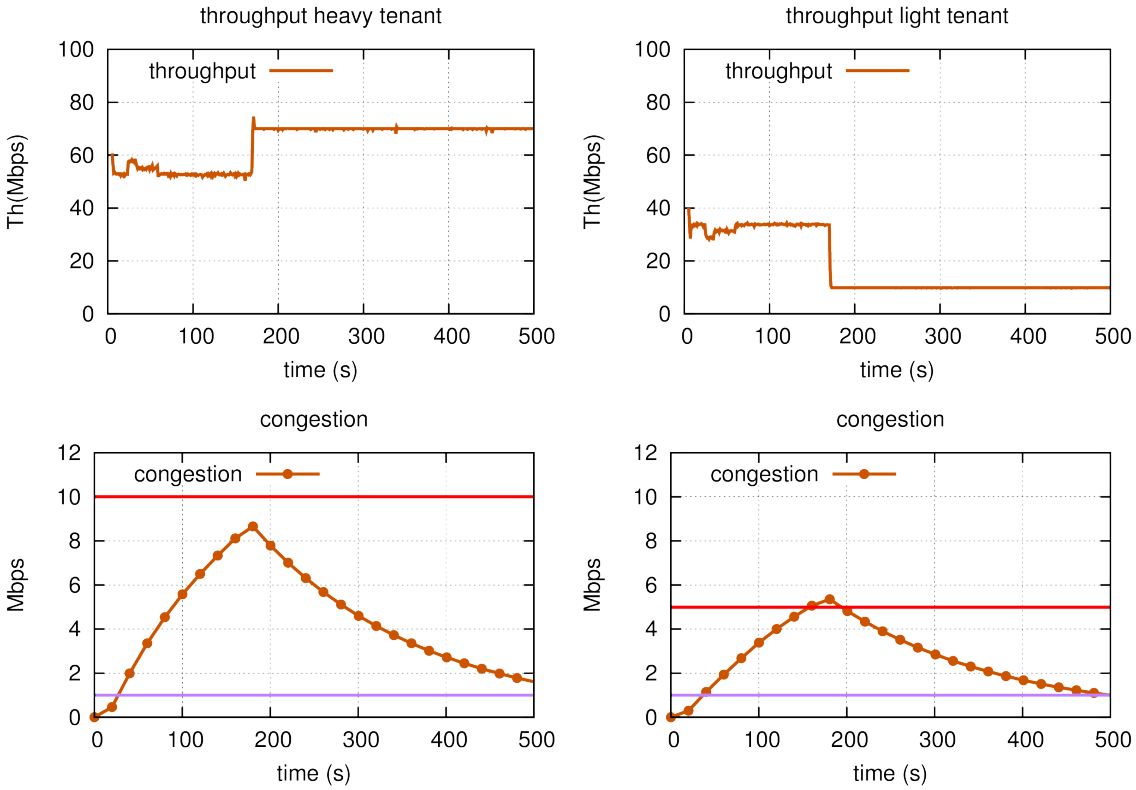

FIGURE 3.13 – Débit et congestion du *light* et du *heavy tenant* avec un traitement différencié

congestion dépasse le seuil autorisé en premier et le contrôleur réagit en limitant sa VM avec un policer, ce qui réduit la congestion dans le réseau. Bien que le *heavy tenant* soit celui qui a le plus d'impact sur la congestion du réseau, c'est le *light tenant* qui est limité en cohérence avec la règle de limitation de congestion qui lui a été appliquée. Ainsi, le contrôleur peut appliquer un traitement différencié entre les *tenants* selon les règles de limitation de congestion configurées par l'administrateur du cloud.

3.5 Résumé et conclusion

Dans ce chapitre, nous avons présenté une solution de traitement de la congestion dans un réseau intra-data center. La particularité de cet environnement est l'utilisation des réseaux d'*overlay* au-dessus des équipements d'*underlay* pour isoler les *tenants* du cloud se partageant l'infrastructure du data center. Notre solution établit une communication, entre la couche d'*underlay* et la couche d'*overlay* pour que les informations sur l'état du

réseau de la première puissent être exploitées par la seconde afin de traiter la congestion dans le réseau en agissant sur les sources de trafic.

Le marquage ECN effectué par les équipements du data center, pour signaler la congestion sur les paquets des tunnels traversant l'*underlay*, est propagé sur les paquets encapsulés des VMs. Les serveurs hôtes remontent le marquage des VMs au contrôleur du cloud qui réagit lorsqu'un *tenant* dépasse un seuil maximum de congestion préconfiguré. La réaction consiste à installer des policer sur les VMs du *tenant* qui a dépassé le seuil de congestion pour limiter son trafic et son impact sur les autres *tenants*.

Nous avons implanté notre solution dans Neutron, le module réseau d'OpenStack, en ajoutant une règle de limitation de congestion au service QoS. À travers une plate-forme expérimentale, nous avons validé le principe de fonctionnement et montré comment cette approche pouvait être utilisée par l'administrateur du cloud pour contrôler le trafic des *tenants*. Deux exemples d'utilisation sont présentés. Dans le premier exemple, l'objectif est de limiter l'impact des *heavy tenants* sur les autres *tenants* en cas de congestion et apporter une meilleure équité dans l'utilisation du réseau. Nous y arrivons en donnant à tous les *tenants* les mêmes seuils maximaux de congestion, les *heavy tenants* atteignant le seuil autorisé les premiers, ce sont leurs VMs qui sont limitées, tandis que les *light tenants* peuvent continuer à utiliser le réseau sans subir d'impact. Dans le second exemple, l'objectif est de permettre à l'administrateur du cloud d'apporter une limitation de la congestion différenciée entre les *tenants* selon le contrat établi. En accordant à certains *tenants* des seuils maximaux de congestion plus importants, l'administrateur peut prioriser le trafic de certains *tenants* en cas de congestion.

Les résultats que nous avons présentés sont des résultats préliminaires visant à illustrer le fonctionnement de la solution et montrer comment nous pouvons adapter son comportement en modifiant ses paramètres.

CHAPITRE 4

ORCHESTRATION DU CLOUD ET DU WAN

La capacité du cloud à fournir des ressources informatiques aux clients de manière flexible, suivant la demande, et la possibilité de déployer facilement des ressources à grande échelle font de lui une technologie en pleine expansion. Le cloud d'un client peut ainsi se déployer sur plusieurs sites de data centers situés à des distances géographiques importantes. L'interconnexion entre les data centers et les sites du client se fait au travers d'un réseau de transport dont les contraintes en bande passante par exemple peuvent affecter la qualité des services du cloud. Ces contraintes devraient donc être prises en compte dans le déploiement des services du cloud et dans l'établissement des VPNs interconnectant les sites du client et les sites du cloud. Or il existe une séparation physique et logique entre le monde intra-data center, que nous avons vu dans le chapitre précédent, et le réseau de transport inter-data centers qui ne permet pas la prise en compte de la totalité de la chaîne de service dans le déploiement du cloud. En effet, les administrateurs du cloud et les opérateurs du WAN ne sont pas toujours de la même organisation. C'est pour cela que nous proposons dans ce chapitre une architecture d'orchestration des ressources du cloud (p.ex., machines virtuelles, stockage) et du WAN (p.ex., débit) pour fournir un service cloud de bout-en-bout offrant une qualité de service maîtrisée.

Nous avons vu dans le chapitre précédent comment le cloud arrive à offrir une interface simple pour le contrôle dynamique de ses ressources. La programmabilité du réseau de transport peut être apportée avec le paradigme SDN et l'utilisation d'un contrôleur centralisé qui gère l'ensemble des équipements du réseau dans leur gestion du trafic des utilisateurs. De plus, cela permet d'offrir une abstraction du réseau contrôlé et une simpli-

cité de son utilisation par les applications au travers d'APIs.

Avec l'introduction d'un contrôleur SDN, les interconnexions entre data centers et les sites du client peuvent être créées et supprimées dynamiquement. De plus, avec la vision globale que possède le contrôleur sur son réseau, il est possible de prendre en compte l'état du réseau et de surveiller les VPNs créés pour réagir dans le cas où une dégradation de la QoS du réseau de transport affecterait la qualité des services du cloud.

Pour simplifier le contrôle du réseau dans notre architecture, nous proposons Control Application Programming Interface (CAPI) qui est un ensemble de modules, que nous avons implanté dans le contrôleur SDN, et une API qui donne à l'orchestrateur la possibilité d'établir et de superviser dynamiquement les VPNs d'interconnexion. Nous proposons d'utiliser les principes et mécanismes de PCN [6] pour organiser le traitement de la pré-congestion dans le réseau de transport. Nous rappelons que la pré-congestion survient lorsque la charge de trafic dépasse un pourcentage du débit offert par les liens du réseau. L'apparition de la pré-congestion nous donne une indication sur l'état du réseau et offre ainsi la possibilité au contrôleur du réseau et à l'orchestrateur de réagir avant la saturation totale des liens et les pertes de paquets.

Enfin, pour illustrer le fonctionnement de notre architecture d'orchestration et l'utilisation de CAPI, nous avons développé une plate-forme de démonstration qui repose sur des composants open source pour offrir les différentes fonctionnalités de la solution. Cette plate-forme a été présentée dans le cadre du Salon de la Recherche d'Orange Labs 2015.

4.1 Architecture et qualité de service

Dans cette section, nous allons présenter une architecture fondée sur un orchestrateur permettant l'interconnexion de data centers distants au travers d'un réseau de transport. L'objectif est de pouvoir contrôler les ressources des clouds situés dans les data centers (p.ex., machines virtuelles, stockage) ainsi que les VPNs qui fournissent la connectivité entre les data centers dans le réseau de transport, afin d'être en mesure de garantir un niveau de qualité de service satisfaisant aux utilisateurs du cloud. Par ailleurs, l'opérateur du réseau est également intéressé par surveiller l'état de son réseau et le niveau d'occupation des liens pour prévenir d'une surcharge qui dégraderait la qualité de service de ces interconnexions importantes.

Dans la configuration d'une interconnexion de data centers, plusieurs modèles de rôle peuvent être envisagés où le fournisseur des services cloud et l'opérateur du réseau de transport peuvent faire partie de la même organisation ou pas. Dans le premier cas, au-

cun arrangement spécifique n'est requis, alors que dans le second, l'interconnexion devrait reposer sur un accord commercial et/ou technique dont les termes sont définis lors de la négociation du SLA. Une interface est alors nécessaire entre les deux entités, ce qui est réalisable au travers d'une API.

D'abord, nous nous intéresserons à l'architecture d'orchestration pour la gestion des services cloud et réseau de bout-en-bout, puis nous nous intéresserons à l'infrastructure de surveillance du réseau et du trafic des VPNs.

4.1.1 Architecture d'orchestration

L'architecture que nous proposons repose sur des composants open source pour offrir les plates-formes cloud et le contrôle SDN. L'utilisation de composants open source nous permet de bénéficier de larges communautés contribuant aux différents projets, d'avoir un code source accessible pour comprendre le comportement des mécanismes implantés, et donc d'avoir la possibilité de modifier les fonctionnalités proposées ainsi que d'en ajouter si nécessaire. Enfin, les composants open source nous évitent d'être liés à du matériel et/ou logiciel spécifique à un constructeur et nous garantissent un niveau élevé d'interopérabilité.

La plate-forme cloud choisie ici est OpenStack que nous avons présentée dans le chapitre précédent, et la plate-forme de contrôle SDN choisie est ODL dont une présentation sera faite dans la section 4.2, mais il faut préciser que n'importe quel composant proposant les mêmes fonctionnalités conviendrait à l'architecture retenue.

La figure 4.1 présente l'architecture qui englobe l'infrastructure cloud et le réseau de transport. Le cloud est ici déployé sur des data centers géographiquement distants. Sur chacun d'eux, un contrôleur de cloud (OpenStack dans notre cas) gère la configuration de l'infrastructure (p.ex., instanciation des VMs, interconnexion entre les VMs). Le réseau de transport est géré par un contrôleur SDN (ODL dans notre cas) qui fournit un service VPN pour l'interconnexion des data centers. Les sites des data centers sont connectés au réseau de transport au niveau des nœuds PE situés en périphérie du réseau. Le raccordement se fait au travers d'Attachment Circuits (ACs), typiquement fondés sur des ports Ethernet ou des VLANs. Ainsi, le trafic cloud est acheminé de manière transparente par le réseau de transport dans des VPNs qui relient les PEs. Dans le cas d'un L2-VPN par exemple, le trafic est d'abord encapsulé dans des Pseudo Wires (PWs) Ethernet puis dans des LSPs MPLS qui transitent via les nœuds intérieurs du réseau (Label Switch Routers). Ces VPNs sont établis, maintenus et supprimés par le contrôleur SDN qui a une vue globale sur le réseau et ses équipements.

Un orchestrateur est nécessaire pour garantir la cohérence dans la configuration des

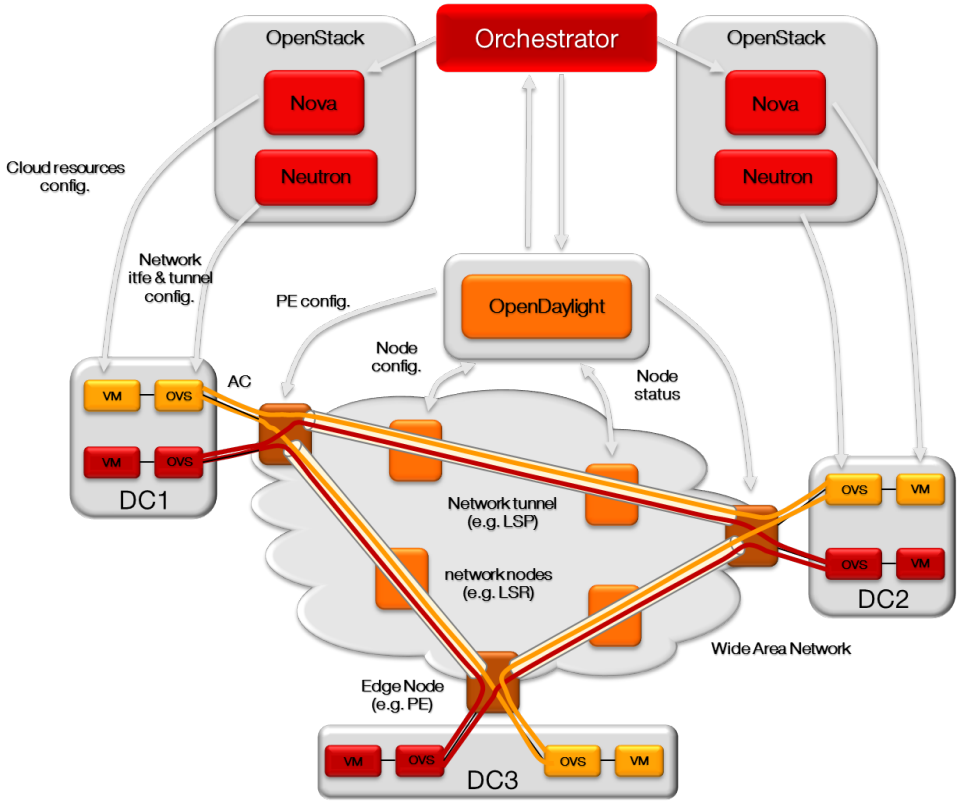

FIGURE 4.1 – Architecture d'orchestration du cloud et du WAN

ressources des clouds dans les data centers distants, et entre le cloud et le WAN qui interconnecte ces data centers. L'orchestrateur peut être vu ici comme un surcontrôleur qui utilise les APIs offertes par les contrôleurs du cloud et du WAN afin d'offrir un service de bout-en-bout aux *tenants* du cloud. Ainsi, l'utilisation des ressources peut-elle être gérée à partir de l'orchestrateur afin de satisfaire les contraintes de disponibilité des ressources cloud et du réseau. L'orchestrateur n'a pas de vue détaillée des éléments du cloud et du réseau. Les contrôleurs du cloud et du WAN, qui lui servent d'abstraction, mettent à sa disposition des informations sur les ressources disponibles au travers de leurs APIs. Pour instancier des machines virtuelles ou pour créer un VPN entre deux PEs du réseau par exemple, il doit faire une requête au travers de l'API appropriée. L'orchestrateur peut représenter un point d'entrée unique pour le *tenant* qui voudrait accéder à ses services cloud et réseau.

Une telle solution permet deux niveaux de réaction en cas de dégradation de la qualité de service du VPN : un premier niveau de réaction qui vient du contrôleur du réseau, et un

second niveau qui vient de l'orchestrateur. Dans le premier niveau de réaction, le contrôleur surveille en permanence l'état des liens qui constituent le chemin d'un VPN dans le réseau. Par exemple, dans le cas d'une charge réseau élevée qui risquerait de dégrader la qualité de service d'un VPN, le contrôleur du réseau peut de manière autonome prendre la décision de rerouter le VPN afin d'éviter les liens chargés, puis mettre l'information du changement opéré au niveau de l'API afin d'en informer l'orchestrateur. Une autre situation concerne le cas d'une défaillance d'un lien réseau affectant un VPN. Ici, la restauration du VPN par le contrôleur est nécessaire pour la protection du réseau.

La réaction de second niveau vient de la part de l'orchestrateur dans le cas où le contrôleur SDN n'a pas été en mesure de faire face à une surcharge réseau ou dans le cas où l'orchestrateur veut prendre en charge le contrôle des ressources réseau. Grâce à l'API offerte par le contrôleur du réseau, l'orchestrateur peut surveiller l'état du VPN et réagir, si nécessaire, non seulement en agissant sur le VPN mais également sur les ressources cloud, par exemple en relocalisant les services cloud d'un data center surchargé vers un autre site avec une charge moindre. La réaction de l'orchestrateur et la migration des ressources cloud peuvent être des décisions automatiques, ou manuelles prises par l'administrateur ou le *tenant* du cloud lui-même.

4.1.2 Surveillance de l'état du réseau

Le WAN, qui se charge de transporter le trafic du cloud d'un data center à un autre, a plusieurs contraintes (p.ex., bande passante, délai, pertes), et les VPNs reliant les sites distants ont besoin d'une qualité de service qui va directement influencer la qualité des services cloud transportés. Afin de prévenir la dégradation de la qualité de service, un suivi de l'état du réseau est nécessaire afin de réagir dès la détection d'un risque potentiel pour les VPNs. Nous présentons dans ce qui suit une infrastructure de surveillance de la charge du réseau en se fondant sur les principes de PCN [6] pour collecter les informations de pré-congestion. Nous rappelons que la pré-congestion survient lorsque le débit d'arrivée des paquets sur un lien dépasse un seuil pré-configuré sur le lien.

PCN repose sur la mesure du trafic sur le lien et sur le marquage des paquets pour la détection et le rapport de la pré-congestion par les nœuds internes du domaine PCN. Il réutilise les bits ECN [18] de l'en-tête IP pour signaler la pré-congestion [69]. Les points de code utilisés par PCN sont :

— "00" pour not-PCN
— "01" pour Not-Marked (NM)
— "10" pour threshold-marked (ThM)

— "11" pour excess-traffic-marked (ETM)

Le compteur installé sur les liens est configuré avec deux seuils, un seuil threshold-rate et un seuil excess-rate, les deux étant inférieurs à la capacité du lien. Le compteur suit l'évolution de l'agrégat de trafic PCN sur l'interface de sortie. Lorsque le débit de ce trafic dépasse le threshold-rate, tous les paquets PCN sont marqués ThM, et lorsque le débit du trafic dépasse l'excess-rate, seule la fraction de paquets ayant dépassé ce seuil est marquée ETM.

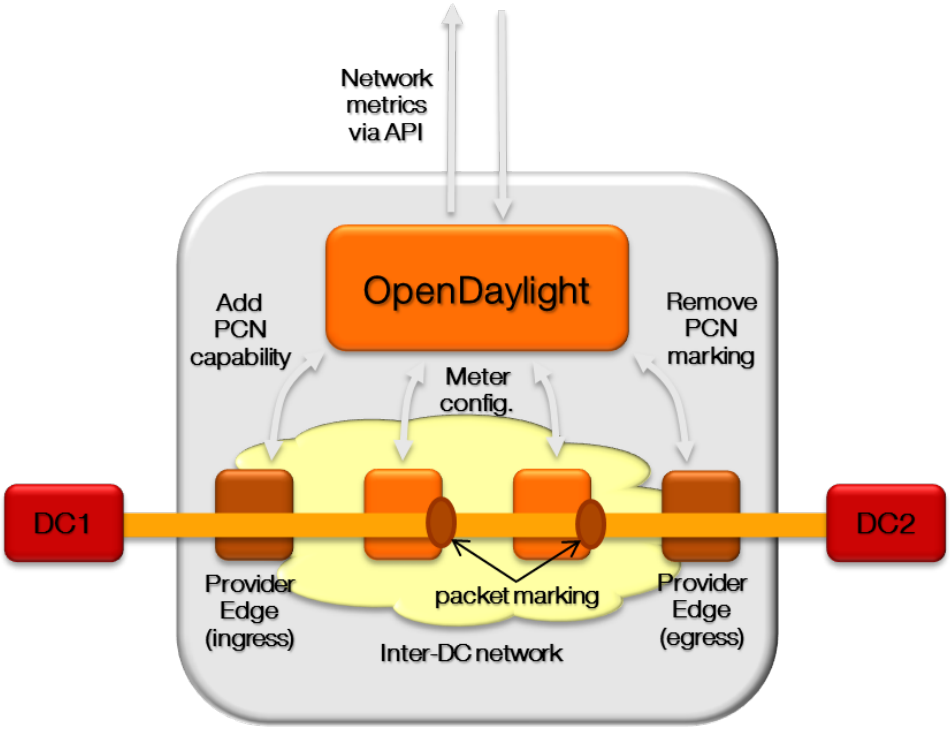

FIGURE 4.2 – Infrastructure de surveillance du réseau

La figure 4.2 montre l'infrastructure de surveillance de l'état du réseau, à travers les mécanismes PCN et le contrôle SDN, que nous utilisons dans notre infrastructure d'orchestration décrite plus haut. Un VPN a été mis en place par le contrôleur du réseau (ODL) afin de connecter le premier data center (DC1) et le second data center (DC2). Pour rendre le mécanisme autonome vis-à-vis des machines d'extrémité situées dans les data centers, l'utilisation de PCN dans le réseau est entièrement gérée par le contrôleur SDN qui ajoute la capacité de marquage PCN aux paquets du VPN à l'entrée du réseau, et qui enlève le marquage PCN à la sortie du réseau, rendant le mécanisme totalement transparent aux ma-

chines d'extrémité. Des compteurs PCN sont également installés sur les nœuds internes par le contrôleur SDN et sont configurés avec des seuils appropriés pour marquer les paquets PCN si les seuils sont dépassés par le débit du trafic agrégé sur le lien.

À la sortie du réseau, le contrôleur collecte les statistiques sur le nombre total d'octets dans le VPN, le nombre d'octets marqués ThM, le nombre d'octets marqués ETM, et calcule ainsi le débit et les fractions de paquets marqués. Ces métriques sont calculées périodiquement sur un intervalle de temps Δt en utilisant une moyenne EWMA avec un facteur de lissage α :

$$Th_i = \alpha \times N_i/\Delta t + (1-\alpha) \times Th_{i-1} \tag{4.1}$$

$$F_{thm_i} = \alpha \times N_{thm_i}/N_i + (1-\alpha) \times F_{thm_{i-1}} \tag{4.2}$$

$$F_{etm_i} = \alpha \times N_{etm_i}/N_i + (1-\alpha) \times F_{etm_{i-1}} \tag{4.3}$$

Où :

— N_i est le nombre total d'octets sur l'intervalle i
— N_{thm_i} est le nombre total d'octets ThM sur l'intervalle i
— N_{etm_i} est le nombre total d'octets ETM sur l'intervalle i
— Th_i est lé débit du VPN surveillé
— F_{thm_i} est la fraction d'octets ThM
— F_{etm_i} est la fraction d'octets ETM

Dans [6], les seuils ThM et ETM de PCN sont utilisés pour mettre en œuvre des fonctions d'admission et de terminaison de flux respectivement à l'entrée du réseau. Dans le contexte de notre architecture, nous proposons d'utiliser les deux seuils pour refléter les deux niveaux de réaction introduits dans la section 4.1.1. Le seuil ThM peut être configuré à 50% de la capacité du lien par exemple, tandis que le seuil ETM est configuré à 80% de celle-ci. Avec la connaissance des équipements réseau qui constituent le chemin du VPN et des informations obtenues à partir des compteurs des liens, si la charge du trafic dépasse le premier seuil, le contrôleur du réseau peut réagir en reroutant le trafic du VPN vers un chemin moins encombré. Si toutefois la charge du trafic continue à augmenter et dépasse le second seuil, par exemple dans le cas où il n'est pas possible de rerouter le trafic, l'orchestrateur en est informé grâce à l'API du contrôleur. L'orchestrateur peut alors agir sur les ressources du cloud pour éviter la dégradation de la qualité de service. Les métriques calculées sont exposées grâce à l'API et peuvent constituer des entrées des

algorithmes déployés par le contrôleur du réseau et l'orchestrateur.

L'utilisation des concepts et des mécanismes PCN dans notre architecture apporte plusieurs avantages. D'abord, nous n'avons pas besoin de maintenir l'état de chaque VPN dans le cœur du réseau car nous avons uniquement besoin des compteurs et du marquage du trafic agrégé sur les liens, ce qui rend la solution de surveillance du réseau simple et garantit un passage à l'échelle. Même si la fonction de mesure de trafic disponible sur les nœuds du réseau n'est capable de marquer les paquets que par rapport à un seuil unique, le système peut toujours offrir une information sur la charge du réseau aux contrôleurs. Les statisques par VPN sont disponibles directement sur les nœuds de sortie qui terminent les VPNs où elles sont collectées par le contrôleur du réseau.

Ensuite, l'utilisation de PCN est gérée de bout-en-bout par le contrôleur SDN, rendant le système autonome et n'ayant besoin d'aucune intervention des utilisateurs. Enfin, cette approche fournit à l'architecture d'orchestration une infrastructure pour connaître la charge du réseau, libre au contrôleur du réseau et à l'orchestateur d'implanter les algorithmes et les comportements qui réagissent à l'information de pré-congestion.

Bien que cette approche apporte plusieurs avantages, elle présente également quelques défis. PCN ne réserve pas explicitement de ressources sur les liens pour les VPNs, la qualité de l'interconnexion est évaluée après sa création. Elle est déduite des statistiques collectées périodiquement à la sortie du réseau par le contrôleur. La fréquence avec laquelle le contrôleur reçoit les statistiques des VPNs et l'intervalle Δt sur lequel sont calculées les métriques influencent directement la réactivité du contrôleur du réseau et de l'orchestrateur. Pour avoir une réaction rapide, Δt devrait être petit, mais il est contraint par un valeur minimale qui est le temps entre deux collectes de statistiques, temps qui dépend du contrôleur SDN, des équipements du réseau et des protocoles utilisés pour la collecte des données. Les compteurs, le marquage et le calcul des métriques introduisent quelques paramètres (p.ex., le facteur de lissage α) qui influencent également la réactivité du contrôleur du réseau et de l'orchestrateur. En revanche, nous pouvons nous appuyer sur des études significatives qui ont été menées sur PCN par le groupe de travail de l'IETF [70], et par plusieurs recherches sur le mécanisme en interne à Orange, pour permettre une bonne récupération de l'état des VPNs.

4.2 CAPI : une API pour le contrôle du réseau

Dans cette section, nous allons présenter CAPI, une API et un ensemble de modules pour le contrôle du réseau proposés dans le cadre de notre architecture. Tous les éléments

sont implantés au sein du contrôleur de réseau, qui est dans notre cas ODL que nous allons présenter avant de décrire notre API et le comportement de nos modules.

4.2.1 OpenDaylight

ODL est une plate-forme ouverte permettant la programmabilité du réseau et la mise en œuvre des concepts SDN. Son objectif est « d'accélérer l'adoption de SDN et de créer une fondation solide pour NFV » [71]. Comme OpenStack, ODL repose sur une architecture modulaire qui offre de nombreux composants pour le déploiement d'un contrôleur SDN avec plusieurs interfaces dites « sud » (protocoles qui communiquent avec les équipements réseau comme OpenFlow ou NETCONF) et offre également une abstraction du réseau aux applications qui utilisent les interfaces dites « nord » (APIs des modules d'ODL).

Outre sa nature open source et le fait que plusieurs équipementiers réseau (p.ex., Cisco, Ericsson) soient impliqués dans son enrichissement, ODL fournit une infrastructure qui contient plusieurs outils et technologies essentiels au développement de notre solution de surveillance du réseau. Sa facilité d'insertion dans notre architecture pour offrir une API de contrôle du réseau à l'orchestrateur a motivé son utilisation en tant que plate-forme de contrôle SDN.

4.2.2 CAPI

L'architecture modulaire d'ODL est fondée sur les spécifications de l'Open Services Gateway initiative (OSGi) [72], qui offrent un modèle de gestion de cycle de vie dynamique (p.ex., mises à jour) pour les modules ainsi que plusieurs autres services très utiles (p.ex., contrôle d'accès) pour le maintien et l'utilisation de l'architecture. Comme représenté sur la figure 4.3, CAPI est implanté en tant que module additionnel à la plate-forme de contrôle ODL, et est ainsi capable de bénéficier des services réseau fondamentaux offerts par ODL tels que le gestionnaire de la topologie et le gestionnaire des équipements. La couche Service Abstraction Layer (SAL) offre la connectivité de tous les modules aux protocoles de l'interface sud, aux APIs de l'interface nord, et aux bases de données.

Deux bases de données sont disponibles dans ODL : la base de configuration où les données sur la configuration courante du réseau (p.ex., identifiants de VPN, endpoints) sont sauvegardées, et la base opérationnelle où les données concernant l'état courant du réseau (p.ex., statistiques du réseau) sont sauvegardées. Les bases de données sont modélisées suivant le langage YANG [32] qui définit leur syntaxe et leur sémantique. L'accès aux bases de données se fait au travers du protocole RESTCONF [73]. Ce protocole est fondé

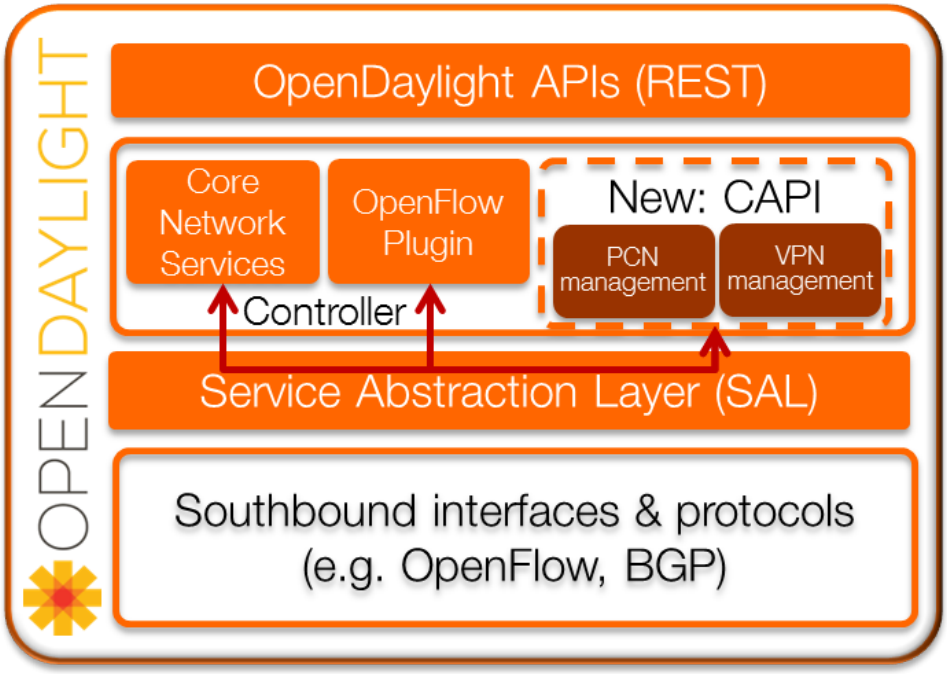

FIGURE 4.3 – Nouveaux modules introduits sur la plate-forme ODL

sur HTTP et utilise les opérations Create, Retrieve, Update and Delete (CRUD) pour manipuler les données de configuration et d'état au travers de Uniform Resource Identifiers (URIs) en suivant les principes REpresentational State Transfer (REST) [74]. Cela fait de CAPI une API RESTful, offrant la simplicité de ce genre d'interfaces.

Afin de présenter CAPI, prenons un exemple de son utilisation depuis l'orchestrateur. Soit deux DCs, le premier étant attaché au PE-1 au travers d'AC-1, tandis que le second est attaché au PE-2 au travers d'AC-2. Afin d'interconnecter au travers d'un VPN des VLANs localisés dans les deux data centers, l'orchestrateur doit simplement accéder à la base de données de configuration, grâce à une requête PUT, avec l'URI du VPN à créer par CAPI dans ODL : *http ://opendaylight :8181/restconf/config/capi :vpns/vpn/VPN12*. Les éléments de la requête sont formatés sur la base d'une description en JavaScript Object Notation (JSON) comme suit :

```
1 {
2   "vpn":
3   [
```

```
    {
    "id": "VPN12",
    "endpoint":
    [
      {
      "id": "DC1",
      "node": "PE-1",
      "connector": "AC-1",
      "vlan-id": "101"
      },
      {
      "id": "DC2",
      "node": "PE-2",
      "connector": "AC-2",
      "vlan-id": "101"
      }
    ]
    }
  ]
}
```

Il est à noter que les requêtes faites au contrôleur du réseau n'ont aucune notion des utilisateurs du cloud. En effet, le contrôleur du réseau ne devrait pas être conscient de l'environnement cloud et ne devrait être concerné que par le contrôle du réseau et l'établissement des VPNs. La correspondance entre les *tenants* du cloud et leurs VPNs est établie au niveau de l'orchestrateur qui seul possède une vue de l'ensemble de l'architecture.

La requête ainsi envoyée grâce à l'API de CAPI est ensuite traitée par le module de gestion des VPNs de CAPI. Dans le cas de l'utilisation d'OpenFlow en tant que protocole d'interface sud, les commutateurs OpenFlow se connectent au plugin OpenFlow d'ODL lors de leur activation ou instanciation dans le réseau. Grâce au Link Layer Discovery Protocol (LLDP), les commutateurs, leurs voisins et les liens les connectant sont découverts et enregistrés dans les bases de données de la SAL. Le module de gestion des VPNs de CAPI s'enregistre à son tour auprès de la SAL pour recevoir toutes les notifications concernant des modifications sur les équipements contrôlés par le plugin OpenFlow.

La bibliothèque Java Universal Network/Graph (JUNG) 2.0 [75] est utilisée pour construire

la topologie réseau à partir des informations reçues du plugin OpenFlow, et pour calculer le chemin le plus court de PE-1 à PE-2 et de PE-2 à PE-1 grâce à l'algorithme de Dijkstra en utilisant la pré-congestion comme poids pour chaque lien. Les chemins calculés (l'ensemble des couples (nœuds, interfaces) de chaque chemin) sont ensuite sauvegardés dans la base de données opérationnelle via la SAL par le module de gestion des VPNs et les règles de transfert des paquets du VPN sont poussées via le plugin OpenFlow sur les commutateurs concernés.

Le module de gestion de PCN de CAPI est invoqué à son tour pour définir les règles spécifiques de surveillance du VPN que le plugin OpenFlow doit pousser comme l'ajout et la suppression de la capacité PCN aux paquets du VPN sur chaque entrée et sortie respectivement. Les fonctions de mesure de trafic et de marquage PCN sur les nœuds internes du réseau sont également installées par le module. Ensuite, le plugin OpenFlow sonde périodiquement les interfaces de sortie des deux directions du VPN pour rassembler les statistiques sur les paquets marqués et non marqués, puis CAPI reçoit les notifications de chaque mise à jour des statistiques au travers de la SAL. Le module de gestion de PCN calcule les métriques présentées dans la section 4.1.2 et les expose grâce à l'API à l'orchestrateur, et notifie également le module de gestion des VPNs en cas de dépassement d'un seuil pour un VPN surveillé.

Pour récupérer les statistiques sur la connectivité du VPN de PE-1 vers PE-2, l'orchestrateur peut simplement accéder à la base de données opérationnelle, grâce à une requête GET, avec l'URI du chemin dans CAPI : *http ://opendaylight :8181/restconf/operational/-capi :vpns/vpn/VPN12/path/DC1,DC2/*. La réponse reçue est au format JSON avec les informations suivantes :

```
{
  "path":
  [
    {
    "id": "DC1,DC2",
    "ingress-node": "PE-1",
    "egress-node": "PE-2",
    "statistics":
      {
      "inst-throughput": "100Mbps",
      "mean-throughput": "80Mbps",
```

```
12        "inst-ThM": "100%",
13        "mean-ThM": "80%",
14        "inst-ETM": "20%",
15        "mean-ETM": "0%"
16        }
17     }
18   ]
19 }
```

4.3 Plate-forme expérimentale

Dans cette section, nous allons présenter une plate-forme expérimentale destinée à valider les principes de l'architecture proposée dans la section 4.1, montrer la faisabilité de réalisation à partir de composants existants et évaluer son fonctionnement dans un environnement réel.

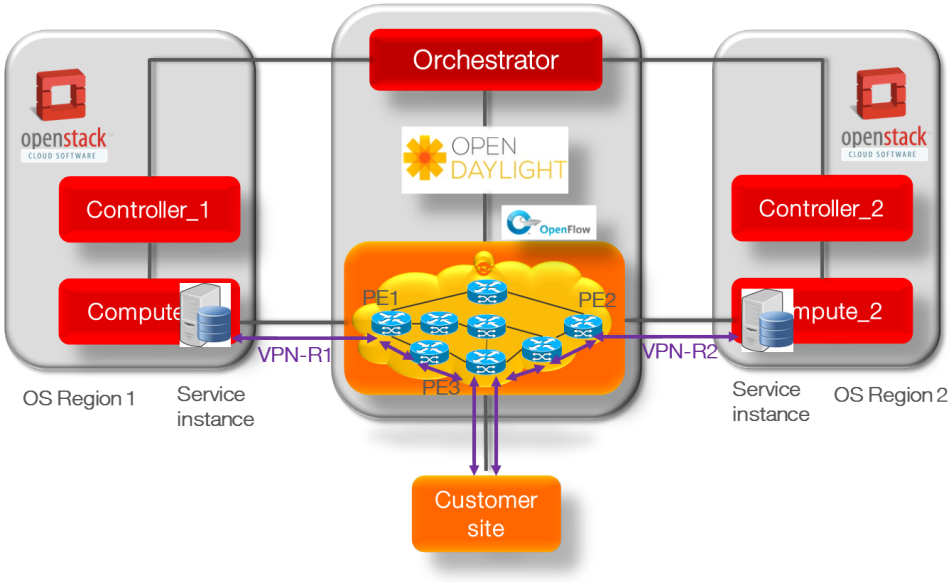

FIGURE 4.4 – Plate-forme expérimentale

La figure 4.4 présente notre plate-forme expérimentale modélisant notre architecture. Le service cloud est installé sur les nœuds 1 et 2 qui font office de sites de data centers et implante chacun une instance distincte d'OpenStack (subdivision en régions dans OpenS-

tack Kilo), tandis que le nœud 3 représente le site d'un client cloud. Enfin, le nœud 4 modélise l'orchestration et les modules du réseau qui fournissent l'interconnexion entre les deux sites du cloud et le site du client. Le réseau de transport est émulé grâce à mininet [76] qui est un logiciel qui permet de créer un réseau virtuel avec des commutateurs OpenFlow, des routeurs et des liens sur lequels les paquets sont traités par un véritable noyau Linux. L'ensemble des commutateurs émulés est contrôlé par ODL (version Lithium avec l'implantation des modules CAPI) grâce au protocole OpenFlow 1.3 (pour pouvoir utiliser la fonctionnalité de mesure de paquets). Les sites 1, 2 et 3 sont connectés aux commutateurs PE1, PE2 et PE3 respectivement.

Pour permettre la surveillance du réseau, nous avons implanté le marquage PCN lors du dépassement d'un seuil configuré sur les commutateurs OpenFlow émulés (de type ofsoftswitch13 [77]). Les commutateurs ofsoftswitch13 sont dans l'espace utilisateur au contraire des commutateurs Open vSwitch qui sont dans l'espace noyau. Certes, les premiers sont moins performants que les seconds, mais ils sont toutefois plus simples à manipuler et à modifier pour permettre le marquage PCN. De plus, ils possèdent la fonctionnalité de mesure des paquets, ce qui n'est pas le cas d'Open vSwitch malgré l'utilisation d'OpenFlow 1.3.

L'objectif de cette plate-forme est de montrer les deux niveaux de réaction présentés dans la section 4.1. Premièrement, le contrôleur du réseau utilise en interne l'information de pré-congestion pour rerouter le trafic du VPN lorsqu'il y a un risque de dégradation de la qualité de service, et opérationnellement informe l'orchestrateur via l'API. Deuxièmement, l'orchestrateur utilise l'information de pré-congestion exposée par le contrôleur du réseau grâce à CAPI pour déclencher une ré-orchestration globale des ressources : le point d'accès au service cloud du client de la première région d'OpenStack vers la seconde région et demander une mise à jour de la connectivité réseau. Nous n'utilisons ici qu'un seul seuil de marquage PCN (le seuil ETM) pour présenter l'un après l'autre les deux niveaux de réaction de l'architecture par souci de simplicité de la démonstration.

Nous allons d'abord présenter le déroulement des événements sur la plate-forme lors de la réaction du contrôleur du réseau et de l'orchestrateur, puis à titre d'illustration du comportement de la solution proposée, nous allons montrer l'influence du facteur de lissage α (qui entre dans le calcul de la fraction de paquets marqués ETM) sur la vitesse de réaction de l'orchestrateur vis-à-vis de l'évolution de la pré-congestion.

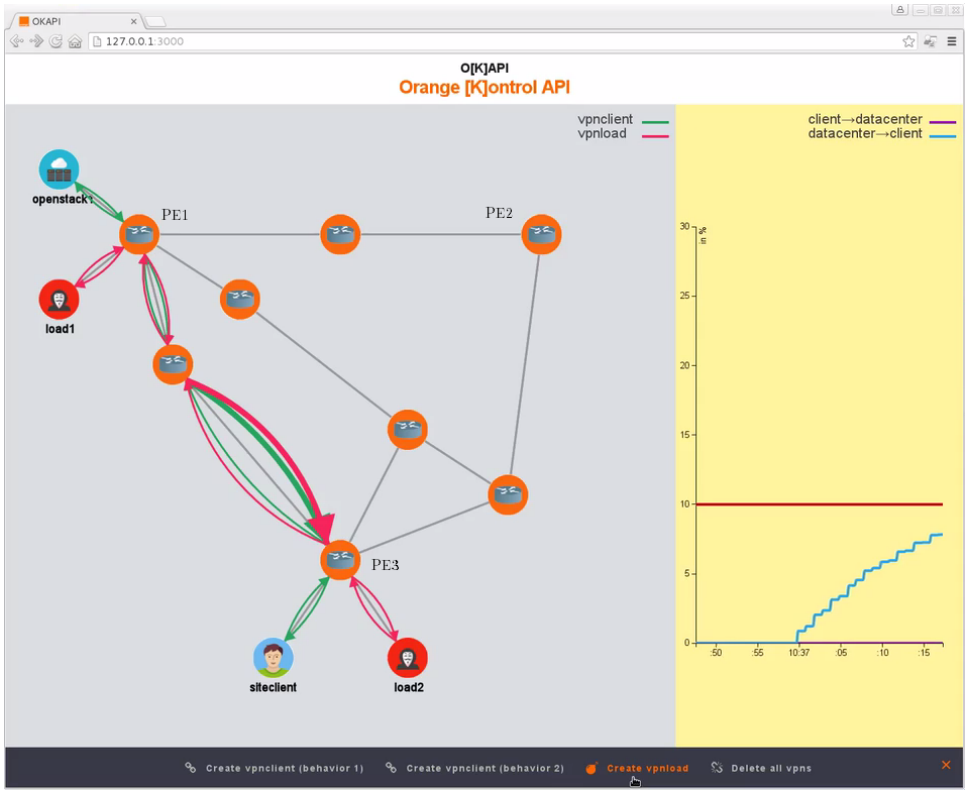

FIGURE 4.5 – Vue de l'orchestrateur avec vision détaillée du réseau

4.3.1 Réactions du contrôleur SDN et de l'orchestrateur

La figure 4.5 montre la vue de l'orchestrateur sur le système avec à gauche une vision détaillée du réseau émulé et à droite l'évolution de la pré-congestion des VPNs. Pour que le client accède à ses services cloud, l'orchestrateur démarre une instance de service (VM) sur la région 1 du cloud au travers des APIs d'OpenStack. Puis, l'orchestrateur demande au contrôleur réseau l'établissement de la connectivité VPN entre PE1 et PE3 au travers de l'interface de CAPI. Ce dernier, après l'établissement du VPN, calcule son débit et la fraction de paquets marqués PCN et les expose grâce à l'API. Les flèches vertes représentent le chemin du VPN du client dont l'intensité augmente progressivement. Nous générons ensuite un trafic de fond (représenté par les flèches rouges) qui emprunte le même chemin que le VPN du client. La pré-congestion sur l'équipement réseau interconnectant PE1 et PE3 augmente dans le sens descendant du cloud vers le client à cause de la charge de trafic sur le réseau. Le nombre de paquets marqués du VPN augmente progressivement, ce que nous pouvons constater avec l'épaisseur des flèches à la sortie de l'équipement et

avec l'évolution de la courbe bleue à droite de la figure 4.5. Lorsque cette courbe de pré-congestion du VPN atteint le seuil rouge (qui a été fixé ici à 10% de paquets marqués pour le VPN), une réaction à l'augmentation de la pré-congestion est activée, et nous avons configuré deux possibilités : dans un premier cas le contrôleur de réseau réagit, et dans un second cas, l'orchestrateur réagit.

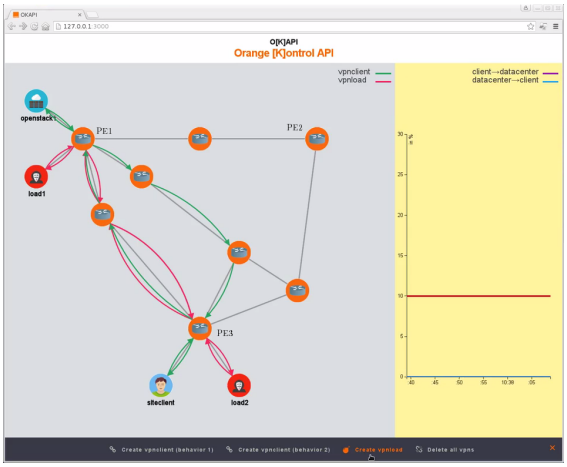

FIGURE 4.6 – Après la réaction du contrôleur du réseau

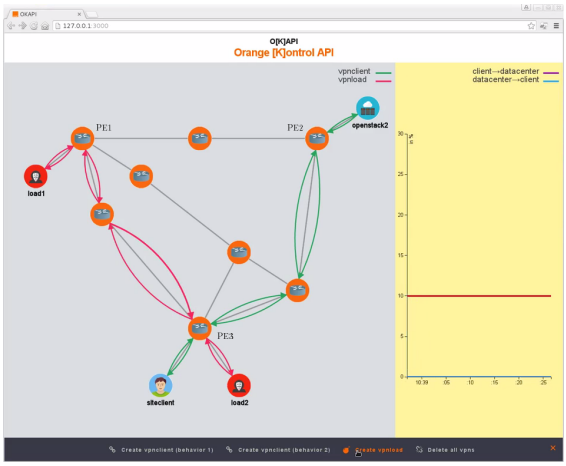

FIGURE 4.7 – Après la réaction de l'orchestrateur

Les figures 4.6 et 4.7 montrent l'état du réseau après la réaction du contrôleur et de l'orchestrateur respectivement. Dans le premier cas, le contrôleur SDN réagit en calculant la route la plus courte et qui évite en même temps l'équipement en pré-congestion. Nous pouvons voir sur la figure 4.6 que seul le chemin de PE1 vers PE3 du VPN a été rerouté, alors que le chemin de PE3 vers PE1 n'a pas été changé car ne subissant pas de pré-congestion. Le second cas, où l'orchestrateur réagit à l'augmentation de la pré-congestion, est présenté en détail sur la figure 4.8. Ici l'orchestrateur lit périodiquement l'état du VPN grâce à CAPI. Lorsqu'il remarque que le seuil de pré-congestion est dépassé, il réagit en changeant le point d'accès au service cloud de la région 1 à la région 2. Pour cela, il démarre une nouvelle instance de service (VM) sur la région 2, puis l'orchestrateur demande au contrôleur du réseau de mettre à jour le VPN afin de connecter le PE3 au PE2 au lieu du PE1. Le client accède maintenant au service cloud à partir de la région 2, l'orchestrateur libère alors les ressources de la région 1 qui ne sont plus utilisées. Nous pouvons constater la liaison entre le client et la région 2 du cloud sur la figure 4.7 avec le déplacement des flèches vertes du VPN du client.

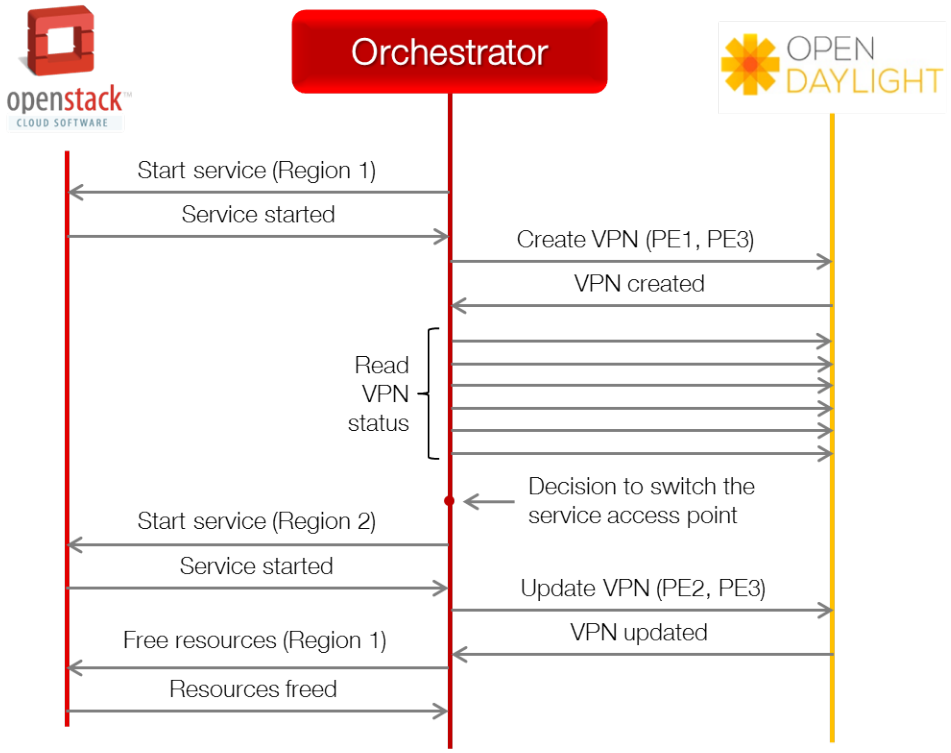

FIGURE 4.8 – Diagramme de séquence du scénario de l'orchestrateur

4.3.2 Influence de α sur le temps de réaction de l'orchestrateur

Les résultats présentés ici visent à illustrer le comportement de l'architecture proposée et montrer comment les paramètres de configuration sont susceptibles d'influer sur le niveau des performances obtenu. Une étude exhaustive des performances reste à faire. L'impact du paramètre de lissage a été plus particulièrement regardé compte tenu de son influence sur les performances de PCN.

Sur la plate-forme, les liens émulés ont un débit de 10Mbit/s, et le seuil PCN est configuré à 80% de cette capacité. Le client connecté à la région 1 du cloud génère un trafic Constant Bit Rate (CBR) de 3Mbit/s. Le trafic de fond qui partage le même chemin que le VPN du client commence à 0Mbit/s et augmente de 1Mbit/s toutes les 10 secondes jusqu'à atteindre 10Mbit/s. Quand le trafic de fond atteint 5Mbit/s, l'agrégat de trafic constitué du trafic de fond et du trafic du client sur le réseau dépasse le seuil PCN configuré et les paquets de l'agrégat sont marqués. Après 30 secondes supplémentaires, quand le trafic de fond atteint 8Mbit/s, l'agrégat de trafic dépasse la capacité des liens et le trafic du client est

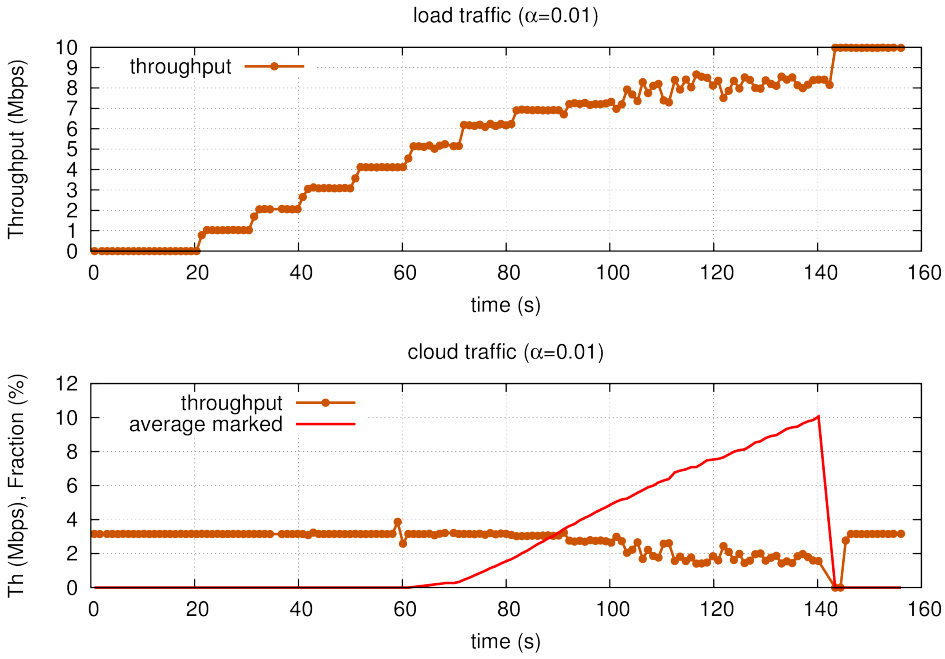

FIGURE 4.9 – Scénario pour $\alpha = 0.01$

affecté. Le but ici est d'analyser la vitesse à laquelle l'orchestrateur réagit à l'augmentation de la pré-congestion afin de changer le point d'accès au service cloud de la région 1 vers la région 2.

Chaque seconde, le contrôleur du réseau récupère les statistiques du VPN à partir des nœuds de sortie (PE3 pour le sens descendant vers le client) et calcule le débit ainsi que le taux moyen de paquets marqués avec un facteur de lissage α (cf. section 4.1.2). La valeur d'une seconde pour la récupération des statistiques est conditionnée par les commutateurs qui ne mettent à jour leurs statistiques qu'une fois par seconde. L'orchestrateur surveille l'état du VPN à la même fréquence, et lorsque le taux moyen de paquets marqués dépasse le seuil configuré de 10%, le point d'accès au service cloud passe de la région 1 à la région 2 et les métriques sont remises à zéro.

Les figures 4.9 and 4.10 montrent le débit du trafic de fond et du VPN du client, ainsi que l'évolution du taux moyen de paquets marqués du VPN du client en fonction du temps pour $\alpha = 0.01$ and $\alpha = 0.1$ respectivement. Dans les deux figures, à partir de $t = 60s$, les paquets du VPN du client sont marqués en pré-congestion, car le trafic agrégé sur les liens atteint le seuil PCN configuré à 8Mbit/s.

Avec $\alpha = 0.01$, le taux moyen de paquets marqués augmente doucement. À $t = 90s$,

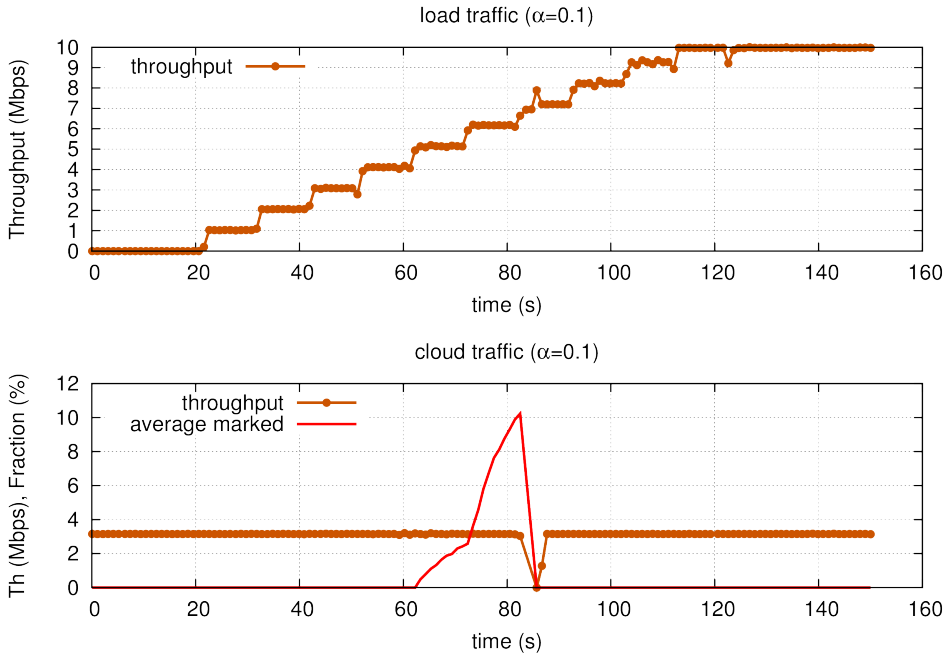

FIGURE 4.10 – Scénario pour $\alpha = 0.1$

le trafic agrégé dépasse la capacité du lien, et nous constatons une perte de débit pour le VPN du client. La réaction de l'orchestrateur survient lorsque le taux moyen de paquets marqués atteint 10% à $t = 140s$, soit près de 80 secondes après le début de l'apparition de la pré-congestion. Le point d'accès au service cloud est transféré à la seconde région et le débit du VPN reprend alors sa valeur de 3Mbit/s.

Avec $\alpha = 0.1$, le taux moyen de paquets marqués augmente bien plus vite qu'avec la précédente valeur. La réaction de l'orchestrateur vient peu après $t = 80s$, juste avant la saturation du lien. Le débit du VPN ne subit donc pas de dégradation de débit significative, et continue d'être surveillé sur le nouveau chemin. Ici l'orchestrateur a pu réagir à temps à l'apparition de la pré-congestion.

La figure 4.11 montre le temps que prend l'orchestrateur à réagir à la pré-congestion (c.-à-d. le temps entre l'apparition de la pré-congestion et la décision de faire passer le point d'accès au service de la région 1 à la région 2 pour le VPN du client) en fonction du facteur de lissage α. Quand α est inférieur à 0.04, le temps de réaction dépasse 30s (temps entre l'apparition de la pré-congestion et la saturation des liens) et augmente rapidement lorsque le facteur de lissage diminue. Quand α est supérieur à 0.04, l'orchestrateur réagit plus vite (moins de 30s) pour éviter la saturation des liens.

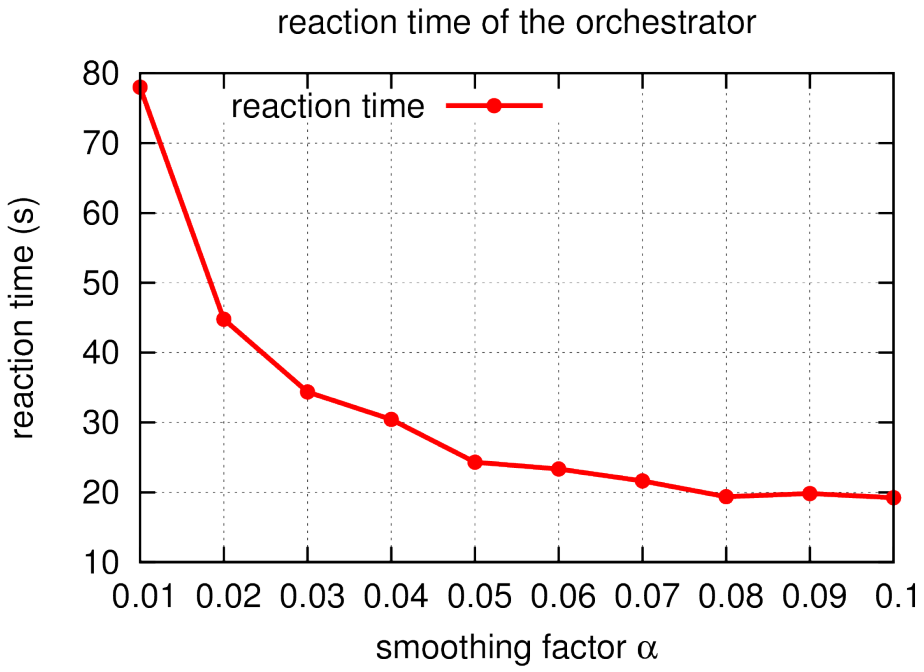

FIGURE 4.11 – Reaction time of the orchestrator versus smoothing factor α

4.4 Résumé et conclusions

Dans ce chapitre, nous avons présenté une architecture d'orchestration visant à offrir un service cloud de bout-en-bout étendu sur des sites géographiquement distribués. Afin de permettre un contrôle simultané des ressources cloud et WAN, notre proposition repose sur un orchestrateur centralisé qui contrôle les plates-formes du cloud et interagit avec le contrôleur SDN du WAN à travers une API dédiée. En s'appuyant sur les informations de l'état du réseau, l'orchestateur a la capacité d'implanter de meilleures stratégies de déploiement des services cloud sur les différentes sites.

Pour l'interaction avec le réseau de transport, nous avons proposé CAPI qui est composé de modules et d'une API dédiés à l'interconnexion de sites distants au travers de VPNs opérateur, ainsi qu'à la surveillance de l'état du réseau et des VPNs. CAPI expose une simple interface REST à l'orchestrateur pour créer les interconnexions nécessaires. L'infrastructure de surveillance du réseau et des VPNs repose sur les principes et mécanismes PCN : grâce à la mesure de charge et au marquage des paquets lors de l'apparition de pré-congestion, il est possible de calculer des métriques de QoS par VPN qui sont ensuite

utilisées par le contrôleur du réseau et par l'orchestrateur afin de réallouer dynamiquement les ressources réseau et/ou cloud afin d'éviter une dégradation de la qualité de service pour les utilisateurs du cloud.

La plate-forme expérimentale que nous avons développée nous a permis de valider les principes de fonctionnement de l'architecture proposée et de montrer ce qu'il est possible d'offrir comme services cloud grâce à l'utilisation de CAPI et de l'orchestrateur. À titre d'exemple, nous avons modélisé deux niveaux de réaction au risque de dégradation de la qualité de service, avec une première réaction venant du contrôleur du réseau qui agit sur les ressources du WAN pour maintenir la QoS offerte aux VPN du client, et une seconde réaction venant de l'orchestrateur qui peut agir également sur les ressources du cloud pour maintenir l'accès de l'utilisateur à ses services en prenant en compte l'état global du cloud et du WAN.

Grâce à la plate-forme expérimentale, nous avons pu étudier le comportement de la solution proposée, ainsi que le niveau de performance qui peut être attendu. En particulier, nous avons pu évaluer la réactivité de l'orchestrateur à l'apparition de la pré-congestion. Les premiers résultats montrent que le facteur de lissage utilisé pour estimer la pré-congestion dans le réseau affecte grandement le comportement de la solution, traduisant la sensibilité de la solution à son paramétrage. En conséquence, si le comportement général est conforme aux attentes et globalement satisfaisant, un travail supplémentaire sera nécessaire avant d'envisager un déploiement dans un réseau réel pour étudier l'impact des autres paramètres et déterminer une configuration permettant d'obtenir un niveau de performance satisfaisant.

CONCLUSION

L'utilisation intensive de l'internet dans des contextes de plus en plus variés a transformé ce réseau en la plate-forme dominante pour l'échange d'informations. Le trafic qui ne cesse d'augmenter en conséquence met en péril la qualité de service des utilisateurs qui se trouve menacée par le risque de congestion pouvant survenir dans le réseau. L'ensemble des acteurs de l'internet est alors amené à répondre à ce problème en s'adaptant aux nouveaux contextes de l'internet, en prenant en compte la diversité des applications qui possèdent chacune leurs propres exigences de qualité de service. La réponse simple qui consiste à augmenter les capacités réseau, en plus de son coût qui peut se révéler prohibitif, n'est pas toujours possible par exemple pour les réseaux radio, et ne permet de répondre que partiellement à certains défis tels que l'amélioration de la latence du réseau. Ceci explique les nombreux travaux qui sont menés aujourd'hui au sein de la communauté de l'internet, en particulier au sein de l'IETF, sur la gestion de trafic s'appuyant sur des informations de charge réseau et de congestion. L'idée de base est alors d'obtenir un comportement plus prédictif du réseau et des sources de trafic, plutôt que de réagir aux pertes d'information.

Dans cette thèse, nous avons étudié les nouveaux paradigmes de traitement de la congestion, et leur application dans différents environnements, qui répondent à la contrainte particulière d'un opérateur de réseau tel qu'Orange : offrir une qualité de service à ses utilisateurs tout en leur assurant l'équité dans l'utilisation de son réseau. À chaque fois, une attention particulière a été apportée aux aspects opérationnels, c'est-à-dire à la facilité d'implantation et de configuration, ou encore à la robustesse vis-à-vis de l'environnement des utilisateurs et du réseau afin d'obtenir un niveau de performance satisfaisant et stable, etc.

Synthèse des résultats

ConEx

Dans le cadre des approches collaboratives, l'IETF a proposé ConEx, un mécanisme qui expose la congestion rencontrée par les utilisateurs dans le réseau, ce qui permet à l'opérateur de policer chaque utilisateur en entrée de réseau proportionnellement à la congestion exposée. Notre intérêt pour ce mécanisme nous a mené à une étude approfondie de son comportement en passant par une évaluation de ses performances afin d'établir son efficacité dans l'établissement de l'équité entre les utilisateurs.

Nous avons tout d'abord évalué l'impact que pouvaient avoir différents paramètres de configuration et d'environnement sur le comportement du mécanisme. Nous avons montré que, pour un paramétrage raisonnablement complexe, ConEx peut être déployé dans un environnement réseau réel et conduire à un niveau de performance satisfaisant et stable. Afin d'améliorer les performances, nous avons pu identifier les paramètres auxquels ConEx est le plus sensible et qui devraient être pris en compte dans l'élaboration de l'algorithme de réaction du policer de congestion.

Un aspect important pour un opérateur est la facilité de déploiement du mécanisme. Nous avons pu montrer dans notre évaluation que ConEx avait des performances satisfaisantes dans la réduction de la congestion et l'établissement de l'équité entre les utilisateurs même lorsque l'on envisage un déploiement initial pour lequel très peu de modifications sont nécessaires. Le déploiement de ConEx peut être ensuite complété si de meilleures performances sont requises.

Finalement à titre d'illustration, nous avons montré comment ConEx pouvait être utilisé pour améliorer de façon significative la qualité d'expérience des utilisateurs d'une application de streaming video telle que YouTube, soit directement par le contrôle des utilisateurs, soit en incitant les utilisateurs à adopter un protocole de transport moins agressif tel que LEDBAT.

Des résultats présentés ici nous pouvons conclure que, d'un point de vue technique, le déploiement de ConEx peut raisonnablement être envisagé dans un réseau d'opérateur. La difficulté principale réside toutefois dans la possibilité d'aboutir à l'écosystème visé par l'IETF, c'est-à-dire un modèle totalement collaboratif faisant intervenir l'ensemble des acteurs de la chaîne : les fournisseurs de contenus, les opérateurs de réseau de transit et d'accès, et finalement les utilisateurs.

Intra-data center

L'évolution des entreprises vers des solutions cloud pour s'offrir des capacités de traitement de données et de stockage, en particulier pour le support de leurs services internet, a mené à une augmentation du trafic dans les environnements cloud et à un risque de congestion qui doit être pris en compte. Afin de faire face à cette situation, nous avons proposé des solutions de traitement de la congestion dans le cloud en nous focalisant d'une part sur l'environnement intra-data center et d'autre part sur l'environnement inter-data centers qui présentent des contraintes différentes.

Pour traiter la congestion dans les réseaux intra-data center, il faut d'abord prendre en compte leurs spécificités. L'utilisation des réseaux d'*overlay* au-dessus du réseau physique d'*underlay* rend le traitement de la congestion difficile en raison du manque de communication entre les deux couches. Elle oblige l'administrateur du cloud à appliquer un contrôle du trafic des *tenants* « en aveugle », sans savoir lesquels contribuent réellement à la congestion dans le réseau.

La solution de traitement de la congestion que nous avons proposée prend en compte cette particularité, afin de permettre au contrôleur du cloud et des réseaux virtuels d'identifier les *tenants* dont le trafic participe à la congestion dans le réseau du data center. Le contrôleur est alors en mesure de policer prioritairement le trafic issu des machines virtuelles de ces *tenants* afin de limiter efficacement la congestion. La solution proposée s'appuie sur le marquage ECN pour propager l'information de congestion du réseau d'*underlay* vers le réseau d'*overlay*, et alloue à chaque *tenant* un volume de congestion autorisé au-delà duquel le *tenant* fait l'objet de limitations.

Nous avons implanté notre solution dans OpenStack, un système de déploiement de clouds open source très largement utilisé aujourd'hui, plus précisément dans Neutron, le composant réseau d'OpenStack. La plate-forme expérimentale que nous avons présentée nous a permis de valider le principe de fonctionnement de notre solution, de montrer sa faisabilité technique et finalement d'illustrer les possibilités d'utilisation de la solution pour soit établir l'équité entre les *tenants* du cloud, soit à l'inverse obtenir un traitement différencié entre *tenants*, typiquement sur la base du contrat souscrit.

Inter-data centers

Le déploiement des services cloud peut s'étendre sur plusieurs data centers géographiquement distants et l'interconnexion entre les différents sites est assuré par un WAN, au travers de VPNs dont les caractéristiques de connectivité affectent directement la qualité

des services clouds. Afin de garantir la coordination entre les ressources du cloud et du WAN, nous avons proposé une architecture d'orchestration centralisée qui a une vue globale sur l'ensemble des ressources. Nous avons proposé également CAPI, une API implantée sur le contrôleur SDN du réseau WAN, qui offre à l'orchestrateur une abstraction du réseau et la possibilité de demander l'établissement de l'interconnexion entre deux sites distants. CAPI implante aussi une solution de surveillance du réseau qui repose sur les principes de PCN afin de remonter les informations sur la pré-congestion du réseau. La pré-congestion peut servir d'indicateur au contrôleur du réseau afin de rerouter les VPNs surveillés en cas de risque de dégradation de la qualité de service. Rendue disponible au travers au travers de l'API, elle peut aussi permettre à l'orchestrateur de réagir au niveau des ressources du cloud en déplaçant le point d'accès au service, par exemple dans le cas où le contrôleur du WAN n'aurait pas pu remédier à l'augmentation de la charge du réseau.

Pour valider notre architecture d'orchestration, nous avons mis en place une plate-forme expérimentale en utilisant OpenStack comme solution de gestion des sites cloud et pour le contrôle du réseau d'interconnexion ODL, au sein duquel nous avons développé les modules nécessaires à la mise en œuvre de CAPI. Grâce à la plate-forme expérimentale, nous avons montré les deux niveaux de réaction à la dégradation de la qualité de service : la réaction du contrôleur du réseau et la réaction de l'orchestrateur. Cela nous a permis une première évaluation de l'influence du paramétrage de la solution sur la vitesse de réaction des contrôleurs et de motiver notre besoin d'étudier davantage le comportement de l'architecture.

Perspectives

À l'issue de cette thèse, nous avons identifié des éléments à développer et des points clés qui méritent des extensions.

Pour ConEx, l'évaluation des performances et la connaissance que nous avons acquise du mécanisme, nous permettent d'envisager une réflexion sur un algorithme de policer de congestion qui serait adapté à nos besoins spécifiques et qui prendrait en compte toutes les contraintes de ConEx : RTT, interaction avec la perte de paquets, précision de la congestion exposée.

Comme mentionné précédemment, un autre point important dans le fonctionnement de ConEx est l'aptitude de l'auditeur à inciter les utilisateurs à déclarer le bon niveau de congestion qu'ils rencontrent. L'algorithme de l'auditeur doit idéalement être couplé avec celui du policer de congestion. En effet, nous pensons qu'un développement conjoint des

algorithmes des deux entités, policer et auditeur, serait bénéfique pour le fonctionnement de la solution globale car cela rendrait la réaction à la congestion plus précise. Avec l'évolution des réseaux vers le paradigme SDN, un contrôleur de réseau peut servir à coordonner le fonctionnement des policers localisés dans les différentes entrées du réseau et celui des auditeurs localisés dans les différentes sorties du réseau.

La solution de contrôle de congestion que nous avons développée pour l'environnement intra-data center a fait ici l'objet d'une étude exploratoire. Une étude plus poussée est nécessaire pour analyser plus en détail son comportement et évaluer plus précisément le niveau de performances qui peut en être attendu, en particulier en faisant varier la configuration des *tenants*, le nombre et le débit des flux de trafic, la taille du réseau *underlay*, etc. De même le passage à l'échelle de la solution d'implantation doit être vérifié. Fort heureusement cette solution est aisément déployable sur une plate-forme de plus grande envergure que celle que nous avons présentée. La facilité de déploiement d'OpenStack, reposant sur la virtualisation et sur la répartition des modules sur plusieurs serveurs communiquant à travers un bus partagé, rend l'évaluation de la solution à grande échelle envisageable. Cela nous donne la possibilité de continuer l'évaluation de la solution dans un environnement plus représentatif d'un data center réel, y compris en incluant des commutateurs réels pour modéliser le réseau intra-data center.

En terme d'amélioration, un objectif majeur serait de rendre automatique la configuration de plusieurs paramètres de la règle de limitation de congestion, comme le facteur de lissage α, pour simplifier la configuration du système à l'administrateur du cloud. Tandis que le seuil minimal peut être configuré très bas de manière conservatrice, d'autres paramètres comme le débit maximum de congestion sont très liés à l'environnement où la solution est déployée, par exemple au nombre de *tenants* qui se partagent l'infrastructure. Au final, l'objectif devrait être de pousser la solution de contrôle de congestion auprès de la communauté OpenStack afin qu'elle soit intégrée au système de déploiement de clouds.

Enfin, la plate-forme expérimentale d'établissement de VPNs entre des data centers distants peut encore être exploitée pour étudier plus en détails le comportement de CAPI et évaluer l'influence de l'environnement réseau et de la charge de trafic sur son comportement. CAPI, en tant que solution dans le contrôleur, peut être améliorée pour intégrer l'utilisation de protocoles de commande du réseau complémentaires, autres qu'OpenFlow. Aujourd'hui, CAPI permet d'établir uniquement une connectivité VPN point-à-point entre deux sites distants, mais une évolution envisageable serait de permettre une connectivité en multipoint entre plusieurs sites. L'étude de l'architecture d'orchestration devrait naturellement s'orienter vers l'évaluation du passage à l'échelle aussi bien en termes de taille de

réseau que de nombre de sites de data centers et de services clouds déployés. Finalement la fiabilité de la solution devrait aussi être étudiée, en s'appuyant en particulier sur la redondance des différents éléments comme l'orchestrateur et le contrôleur du réseau.

Comme nous l'avons vu, le contrôle de la charge de son réseau est, pour un opérateur, une préoccupation grandissante afin de garantir un niveau de qualité de service satisfaisant. Les travaux présentés dans cette thèse sont donc particulièrement importants pour un opérateur tel qu'Orange. Les pistes de solution proposées ici, dont les principes de fonctionnement ont été validés devront faire l'objet d'études complémentaires plus ciblées. Le contexte des data-centers, qui hébergent aujourd'hui la quasi-totalité des applications de l'internet, et qui demain devraient aussi mettre en œuvre de nombreuses fonctions réseau, nous apparaît comme particulièrement intéressant à approfondir. Cela devrait être réalisé avec un recours très important aux technologies SDN/NFV.

COMMUNICATIONS

Conférences internationales avec comité de lecture et sélection

— A. Sanhaji, P. Niger, P. Cadro, and A.-L. Beylot, “Droptail based conex applied to video streaming,” ICNS 2015, pp. 3–10, 2015.

— A. Sanhaji, P. Niger, P. Cadro, C. Ollivier, and A. L. Beylot, “Congestion-based API for cloud and WAN resource optimization,” in 2016 IEEE NetSoft Conference and Workshops (NetSoft), June 2016, pp. 141–145.

Revue internationale

— A. Sanhaji, P. Niger, P. Cadro, and A.-L. Beylot, “Conex performance evaluation and application to video streaming,” Tele15v8n34, International Journal On Advances in Telecommunications, 2015 no 3&4, vol. 8, no. 3 & 4, pp. 202–214, 2015.

Démonstrations

— Présentation de la plate-forme expérimentale ConEx, Salon de la Recherche d’Orange Labs 2013.

— Présentation de la plate-forme expérimentale Orange CAPI, Salon de la Recherche d’Orange Labs 2015.

BIBLIOGRAPHIE

[1] "Data Center Architecture Overview." [Online]. Available : http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/\DC_Infra2_5/DCInfra_1.html

[2] "Software Defined Networking." [Online]. Available : https://www.opennetworking.org/

[3] "OpenStack cloud operating system." [Online]. Available : http://www.openstack.org/

[4] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture : an Overview," RFC 1633 (Informational), Internet Engineering Task Force, Jun. 1994. [Online]. Available : http://www.ietf.org/rfc/rfc1633.txt

[5] D. Grossman, "New Terminology and Clarifications for Diffserv," RFC 3260 (Informational), Internet Engineering Task Force, Apr. 2002. [Online]. Available : http://www.ietf.org/rfc/rfc3260.txt

[6] P. Eardley, "Pre-Congestion Notification (PCN) Architecture," RFC 5559, Oct. 2015. [Online]. Available : https://rfc-editor.org/rfc/rfc5559.txt

[7] B. J. Briscoe, R. Woundy, and A. Cooper, "Congestion Exposure (ConEx) Concepts and Use Cases," RFC 6789, Oct. 2015. [Online]. Available : https://rfc-editor.org/rfc/rfc6789.txt

[8] "Définition de la congestion." [Online]. Available : http://www.larousse.fr/dictionnaires/francais/congestion/18188

[9] S. Bauer, D. D. Clark, and W. Lehr, "The evolution of internet congestion." TPRC, 2009.

[10] C. Bastian, J. Livingood, J. Mills, R. Woundy, and T. Klieber, "Comcast's Protocol-Agnostic Congestion Management System," RFC 6057, Oct. 2015. [Online]. Available : https://rfc-editor.org/rfc/rfc6057.txt

[11] Y. Zhang and N. Ansari, "On architecture design, congestion notification, tcp incast and power consumption in data centers," *IEEE Communications Surveys Tutorials*, vol. 15, no. 1, pp. 39–64, First 2013.

[12] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, Nov. 1984. [Online]. Available : http://doi.acm.org/10.1145/357401.357402

[13] V. Jacobson, "Congestion avoidance and control," in *Symposium Proceedings on Communications Architectures and Protocols*, ser. SIGCOMM '88. New York, NY, USA : ACM, 1988, pp. 314–329. [Online]. Available : http://doi.acm.org/10.1145/52324.52356

[14] M. Allman, V. Paxson, and E. Blanton, "Tcp congestion control," September 2009. [Online]. Available : https://tools.ietf.org/html/rfc5681

[15] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "Tcp selective acknowledgment options," October 1996. [Online]. Available : https://tools.ietf.org/html/rfc2018

[16] G. Texier, "Métrologie des réseaux mesure de la qualité de service," *Techniques de l'ingénieur Administration de réseaux, applications et mise en oeuvre*, vol. base documentaire : TIB481DUO., no. ref. article : te7605, 2016, fre. [Online]. Available : http://www.techniques-ingenieur.fr/base-documentaire/technologies-de-l-information-th9/administration-de-reseaux-applications-et-mise-en-oeuvre-42481210/metrologie-des-reseaux-te7605/

[17] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993. [Online]. Available : http://dx.doi.org/10.1109/90.251892

[18] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ecn) to ip," September 2001. [Online]. Available : http://www.rfc-editor.org/rfc/rfc3168.txt

[19] "Gentle RED." [Online]. Available : http://www.icir.org/floyd/red/gentle.html

[20] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive red : An algorithm for increasing the robustness of red's active queue management," Tech. Rep., 2001.

[21] K. Nichols and V. Jacobson, "Controlling queue delay," *Queue*, vol. 10, no. 5, pp. 20 :20–20 :34, May 2012. [Online]. Available : http://doi.acm.org/10.1145/2208917.2209336

[22] R. Pan, P. Natarajan, F. Baker, and G. White, "PIE : A Lightweight Control Scheme To Address the Bufferbloat Problem," Internet Engineering Task Force, Internet-Draft draft-ietf-aqm-pie-08, Jun. 2016, work in Progress. [Online]. Available : https://tools.ietf.org/html/draft-ietf-aqm-pie-08

[23] G. Karagiannis, K. H. Chan, M. Menth, P. Eardley, B. J. Briscoe, and T. Moncaster, "Overview of Pre-Congestion Notification Encoding," RFC 6627, Oct. 2015. [Online]. Available : https://rfc-editor.org/rfc/rfc6627.txt

[24] M. Kuhlewind and R. Scheffenegger, "Design and evaluation of schemes for more accurate ecn feedback," in *Communications (ICC), 2012 IEEE International Conference on*, June 2012, pp. 6937–6941.

[25] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. –, Aug. 2010. [Online]. Available : http://dl.acm.org/citation.cfm?id=2043164.1851192

[26] J. Jiang and R. Jain, "Analysis of backward congestion notification (bcn) for ethernet in datacenter applications," in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, May 2007, pp. 2456–2460.

[27] Y. Zhang and N. Ansari, "Fair quantized congestion notification in data center networks," *IEEE Transactions on Communications*, vol. 61, no. 11, pp. 4690–4699, November 2013.

[28] "Quantized Congestion Notification and Today's Fibre Channel over Ethernet Networks." [Online]. Available : http://www.cisco.com/c/en/us/products/collateral/switches/nexus-5000-series-switches/white_paper_c11-630674.html

[29] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protocol (snmp)," May 1990. [Online]. Available : https://tools.ietf.org/html/rfc1157

[30] K. McCloghrie and M. Rose, "Management information base for network management of tcp/ip-based : Mib-ii," March 1991. [Online]. Available : https://tools.ietf.org/html/rfc1213

[31] R. Enns, M. Bjorklund, A. Bierman, and J. Schönwälder, "Network Configuration Protocol (NETCONF)," RFC 6241, Oct. 2015. [Online]. Available : https://rfc-editor.org/rfc/rfc6241.txt

[32] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," RFC 6020, Oct. 2015. [Online]. Available : https://rfc-editor.org/rfc/rfc6020.txt

[33] W. Dai and S. Jordan, "Design and impact of data caps," in *2013 IEEE Global Communications Conference (GLOBECOM)*, Dec 2013, pp. 1650–1656.

[34] M. Feknous, T. Houdoin, B. Le Guyader, J. De Biasio, A. Gravey, and J. Torrijos Gijon, "Internet traffic analysis : A case study from two major european operators," in *Computers and Communication (ISCC), 2014 IEEE Symposium on*, June 2014, pp. 1–7.

[35] "Comcast customers hate data caps, but making customers hurt is all part of the plan." [Online]. Available : http://www.theverge.com/2016/4/23/11494510/comcast-att-wsj-data-cap-report

[36] "The FCC won't let Charter/Time Warner put data caps on internet plans." [Online]. Available : http://www.recode.net/2016/4/25/11586392/charter-fcc-broadband-data-caps

[37] M. Mathis and B. J. Briscoe, "Congestion Exposure (ConEx) Concepts, Abstract Mechanism, and Requirements," RFC 7713, Dec. 2015. [Online]. Available : https://rfc-editor.org/rfc/rfc7713.txt

[38] "Class-Based Weighted Fair Queueing." [Online]. Available : http://www.cisco.com/en/US/docs/ios/12_0t/12_0t5/feature/guide/cbwfq.html

[39] J.-L. Le Roux, "Mpls : applications à l'ingénierie de trafic et à la sécurisation," *Techniques de l'ingénieur Réseau Internet : protocoles, multicast, routage, MPLS et mobilité*, vol. base documentaire : TIB289DUO., no. ref. article : te7577, 2016, fre. [Online]. Available : http://www.techniques-ingenieur.fr/base-documentaire/technologies-de-l-information-th9/reseau-internet-protocoles-multicast-routage-mpls-et-mobilite-42289210/mpls-applications-a-l-ingenierie-de-trafic-et-a-la-securisation-te7577/

[40] P. P. Tang and T. Y. C. Tai, "Network traffic characterization using token bucket model," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, Mar 1999, pp. 51–62 vol.1.

[41] B. J. Briscoe, A. Jacquet, T. Moncaster, and A. Smith, "Re-ECN : Adding Accountability for Causing Congestion to TCP/IP," Internet Engineering Task Force, Internet-Draft draft-briscoe-tsvwg-re-ecn-tcp-09, Oct. 2010, work in Progress. [Online]. Available : https://tools.ietf.org/html/draft-briscoe-tsvwg-re-ecn-tcp-09

[42] S. Krishnan, C. Ucendo, M. Kühlewind, and B. Briscoe, "IPv6 Destination Option for Congestion Exposure (ConEx)," RFC 7837, May 2016. [Online]. Available : https://rfc-editor.org/rfc/rfc7837.txt

[43] B. Briscoe, M. Kühlewind, and R. Scheffenegger, "More Accurate ECN Feedback in TCP," Internet Engineering Task Force, Internet-Draft draft-ietf-tcpm-accurate-ecn-01, Jun. 2016, work in Progress. [Online]. Available : https://tools.ietf.org/html/draft-ietf-tcpm-accurate-ecn-01

[44] M. Kühlewind and R. Scheffenegger, "TCP Modifications for Congestion Exposure (ConEx)," RFC 7786, May 2016. [Online]. Available : https://rfc-editor.org/rfc/rfc7786.txt

[45] D. Wagner and M. Kühlewind, "Auditing of Congestion Exposure (ConEx) signals," Internet Engineering Task Force, Internet-Draft draft-wagner-conex-audit-02, Apr. 2016, work in Progress. [Online]. Available : https://tools.ietf.org/html/draft-wagner-conex-audit-02

[46] "The Network Simulator - ns-2." [Online]. Available : http://www.isi.edu/nsnam/ns/

[47] V. Jacobson, R. Braden, and D. Borman, "Tcp extensions for high performance," May 1992. [Online]. Available : https://www.ietf.org/rfc/rfc1323.txt

[48] A. Martin and M. Menth, "Conex-based congestion policing – first performance results," March 2012. [Online]. Available : http://www.ietf.org/proceedings/83/slides/slides-83-conex-5.pdf

[49] N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin, "An argument for increasing tcp's initial congestion window," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 3, pp. 26–33, Jun. 2010. [Online]. Available : http://doi.acm.org/10.1145/1823844.1823848

[50] S. Gebert, R. Pries, D. Schlosser, and K. Heck, "Internet access traffic measurement and analysis," in *Proceedings of the 4th International Conference on Traffic Monitoring and Analysis*, ser. TMA'12. Berlin, Heidelberg : Springer-Verlag, 2012, pp. 29–42. [Online]. Available : http://dx.doi.org/10.1007/978-3-642-28534-9_3

[51] P. Ameigeiras, J. J. Ramos-Munoz, J. Navarro-Ortiz, and J. Lopez-Soler, "Analysis and modelling of youtube traffic," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 4, pp. 360–377, 2012. [Online]. Available : http://dx.doi.org/10.1002/ett.2546

[52] J. Ramos-munoz, J. Prados-Garzon, P. Ameigeiras, J. Navarro-Ortiz, and J. Lopez-soler, "Characteristics of mobile youtube traffic," *Wireless Communications, IEEE*, vol. 21, no. 1, pp. 18–25, February 2014.

[53] R. Schatz, T. Hossfeld, and P. Casas, "Passive youtube qoe monitoring for isps," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, July 2012, pp. 358–364.

[54] T. Hoßfeld, R. Schatz, E. Biersack, and L. Plissonneau, "Internet video delivery in youtube : From traffic measurements to quality of experience," in *Data Traffic Monitoring and Analysis*, ser. Lecture Notes in Computer Science, E. Biersack, C. Callegari, and M. Matijasevic, Eds. Springer Berlin Heidelberg, 2013, vol. 7754, pp. 264–301. [Online]. Available : http://dx.doi.org/10.1007/978-3-642-36784-7_11

[55] Google, "Advanced encoding settings." [Online]. Available : https://support.google.com/youtube/answer/1722171

[56] S. S. et al., "Low extra delay background transport (ledbat)," December 2012. [Online]. Available : http://www.rfc-editor.org/rfc/rfc6817.txt

[57] D. Rossi, C. Testa, S. Valenti, and L. Muscariello, "Ledbat : The new bittorrent congestion control protocol," in *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, Aug 2010, pp. 1–6.

[58] "ConEx Working Group Charter." [Online]. Available : http://datatracker.ietf.org/wg/conex/charter/

[59] J. Dean and S. Ghemawat, "Mapreduce : Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008. [Online]. Available : http://doi.acm.org/10.1145/1327452.1327492

[60] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008. [Online]. Available : http://doi.acm.org/10.1145/1402946.1402967

[61] C. Hopps, "Analysis of an equal-cost multi-path algorithm," 2000. [Online]. Available : https://rfc-editor.org/rfc/rfc2992.txt

[62] "Amazon profit crushes estimates as cloud-service revenue soars." [Online]. Available : http://www.reuters.com/article/us-amazon-results-idUSKCN0XP2WD

[63] "Kernel-based Virtual Machine." [Online]. Available : http://www.linux-kvm.org/

[64] M. Mahalingam, T. Sridhar, M. Bursell, L. Kreeger, C. Wright, K. Duda, P. Agarwal, and D. Dutt, "Virtual eXtensible Local Area Network (VXLAN) : A Framework for

Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," RFC 7348, Oct. 2015. [Online]. Available : https://rfc-editor.org/rfc/rfc7348.txt

[65] "OpenFlow for SDN Architectures." [Online]. Available : https://www.opennetworking.org/sdn-resources/openflow

[66] B. Briscoe, "Tunnelling of explicit congestion notification," 2010. [Online]. Available : https://tools.ietf.org/html/rfc6040

[67] B. J. Briscoe and M. Sridharan, "Network Performance Isolation in Data Centres using Congestion Policing," Internet Engineering Task Force, Internet-Draft draft-briscoe-conex-data-centre-02, Aug. 2014, work in Progress. [Online]. Available : https://tools.ietf.org/html/draft-briscoe-conex-data-centre-02

[68] "Open Virtual Switch." [Online]. Available : http://www.openvswitch.org/

[69] B. Briscoe, T. Moncaster, and M. Menth, "Encoding three pre-congestion notification (pcn) states in the ip header using a single diffserv codepoint (dscp)," July 2012. [Online]. Available : https://www.rfc-editor.org/rfc/rfc6660.txt

[70] "PCN Working Group." [Online]. Available : https://datatracker.ietf.org/wg/pcn/documents/

[71] "OpenDaylight controller infrastructure for SDN deployement." [Online]. Available : https://www.opendaylight.org/

[72] "Open Services Gateway initiative." [Online]. Available : http://www.osgi.org/

[73] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol," Internet Engineering Task Force, Internet-Draft draft-ietf-netconf-restconf-15, Jul. 2016, work in Progress. [Online]. Available : https://tools.ietf.org/html/draft-ietf-netconf-restconf-15

[74] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, May 2002. [Online]. Available : http://doi.acm.org/10.1145/514183.514185

[75] "Java Universal Network/Graph Framework." [Online]. Available : http://jung.sourceforge.net/

[76] "Mininet : network emulation." [Online]. Available : http://www.mininet.org/

[77] "User space switch." [Online]. Available : https://github.com/CPqD/ofsoftswitch13

Résumé

La congestion dans les réseaux est un phénomène qui peut influer sur la qualité de service ressentie par les utilisateurs. L'augmentation continue du trafic sur l'internet fait du phénomène de congestion un problème auquel l'opérateur doit répondre pour satisfaire ses clients. Avec l'évolution de l'architecture des réseaux et l'arrivée de nouvelles applications sur l'internet, de nouveaux paradigmes de contrôle de congestion sont à envisager pour répondre aux attentes des utilisateurs du réseau de l'opérateur. Dans cette thèse, nous examinons les nouvelles approches proposées, telles que ConEx, pour le contrôle de congestion dans le réseau d'un opérateur. Nous proposons une évaluation de ces approches à travers des simulations, et nous proposons également des solutions de contrôle de congestion dans des environnements nouveaux tels que les architectures Software Defined Networking et le cloud déployé sur un ou plusieurs data centers. Nous illustrons le fonctionnement et le potentiel de nos solutions au travers de plates-formes expérimentales.

Mots-clés : ConEx, Trafic Internet, Congestion, Cloud, Data center

Abstract

Network congestion is a phenomenon that can influence the quality of service experienced by the users. The continuous increase of internet traffic makes this phenomenon an issue that should be addressed by the network operator to satisfy its clients. With the evolution of the network architecture and the emergence of new internet applications, new paradigms for congestion control have to be considered as a response to the expectations of network users. In this thesis, we examine new approaches, such as ConEx, for congestion control in an operator's network. We propose an evaluation of these approaches through simulations, and we also provide solutions for congestion control in new environments such as Software-Defined Networking architectures and cloud computing deployed over many data centers. To illustrate the feasibility and potential of our proposals, experimental platforms have been developed.

Keywords : ConEx, Internet Traffic, Congestion, Cloud, Data center