Master 2 de Science, mention Informatique,
spécialité Réseaux Informatiques et Systèmes Embarqués

Présenté par

Thibaut Ehlinger
thibaut.ehlinger@etu.unistra.fr

Supervision des réseaux IP : vers une vérification des performances interdomaines

Encadré par

Pascal Mérindol
merindol@unistra.fr

Au sein de

Equipe réseaux - UFR Mathématiques et informatique de l'université de Strasbourg

Remerciements

Je tiens tout d'abord à remercier l'UFR de mathématiques et informatique de l'université de Strasbourg pour avoir financé ce stage.

Merci aux membres de l'équipe réseaux du laboratoire ICube pour leur accueil chaleureux.

Merci plus particulièrement à Cristel Pelsser et à Fabrice Theoleyre pour leurs conseils judicieux aussi bien sur mon travail scientifique que sur mes choix futurs dans le monde académique.

Merci à Andreas Guillot pour son aide quotidienne.

Enfin un immense merci à Pascal Mérindol pour son encadrement très formateur, sincère et bienveillant.

Table des matières

Friedrisch Nietzsche, dans *Ainsi parlait Zarathoustra*, a dit : "Évaluer, c'est créer : écoutez donc, vous qui êtes créateurs ! C'est l'évaluation qui fait des trésors et des joyaux de toutes choses évaluées."

Les deux premières étapes de la réalisation d'un produit ou d'un projet sont évidentes : la conception, puis la mise en œuvre de celui-ci. Or, pour assurer la pérennité de sa réalisation et pour assimiler les enseignement de sa mise en œuvre il est indispensable de l'*évaluer*. En effet c'est l'évaluation d'un projet qui permet d'éviter de refaire les mêmes erreurs, et c'est l'évaluation d'un outil qui permet son amélioration.

Cette vérité s'applique aux réseaux informatiques : une fois un réseau déployé, son administrateur doit évaluer ses performances et mettre en place un système de détéction d'anomalies. C'est ce qu'on appelle la *supervision* des réseaux informatiques. Elle permet de réagir à court terme aux imprévus comme les pannes et les congestions. Mais elle permet également, à long terme, de s'adapter aux tendances générales du trafic circulant dans un réseau.

Les réseaux informatiques sont des sytèmes complexes pour plusieurs raisons. Premièrement, ils englobent une grande diversité de protocoles et de matériels à tous les niveaux, ce qui en fait des systèmes très hétérogènes. Deuxièmement, ils évoluent très rapidement. Enfin, afin de répartir au maximum la charge des équipements, leur gestion est très largement décentralisée. Si bien qu'il n'existe pas un système unique et simple à déployer permettant d'avoir une vision cohérente et détaillée de l'activité d'un réseau informatique, bien que ce fusse le rêve de tout administrateur. Au lieu de ça, il existe une myriade de dispositifs répondant à différents besoins.

Mon stage a lieu en collaboration avec GÉANT, le fournisseur d'accès à Internet de tous les réseaux nationaux universitaires, appelés NREN. Afin de maîtriser la qualité de service dans les réseaux interuniversitaires, des *tunnels multidomaines* ont été déployés au sein de GÉANT et des NREN. Il s'agit d'entités étanches, traversant plusieurs domaines, destinées à garantir une certaine qualité de service à une catégorie de trafic donnée.

La supervision des tunnels multidomaines en temps réel est une tâche plus complexe encore que la supervision d'un réseau local, car ces tunnels sont traversés par des débits très élevés et que leur topologique est dynamique. C'est dans ce contexte que nous avons conçu un dispositif de supervision innovant : *PathCap*.

Un stage regroupant trois acteurs

Mon stage dispose d'un statut un peu particulier, puisque, comme nous allons le voir, il s'agit d'une collaboration entre trois acteurs : le laboratoire ICube, l'UFR de mathématiques et informatique de Strasbourg et enfin les fournisseur d'accès académique GÉANT et RENATER.

Le laboratoire ICube est un des plus grands laboratoires de recherche en France. Dirigé par le professeur Michel de Mathelin, il compte 274 membres permanents. C'est un laboratoire public financé par l'université de Strasbourg et le CNRS, mais qui héberge des équipes affiliées à de nombreux partenaires dont l'INRIA (Institut national de recherche en informatique et en automatique).

Comme nous pouvons le constater en figure 1, le laboratoire est composé de quatre départements recherche nommés respectivement :

— informatique recherche ;
— imagerie, robotique, télédétection & santé ;
— électronique solide, systèmes et photonique ;
— mécanique.

Le département "informatique recherche" regroupe 8 équipes de recherche, spécialisées dans différentes disciplines de l'informatique allant de l'informatique géométrique et graphique aux réseaux informatiques , en passant par la la science des données et des connaissances.

Mon stage a lieu dans l'équipe réseaux, dirigée par le professeur Thomas Noël. Cette équipe se décline à son tour en deux thématiques, une thématique "Internet des objets" et une thématique "Réseaux de cœur".

Dans mes recherches, je suis encadré par Pascal Mérindol dont les travaux concernent la thématique des réseaux de cœur. Dans le cadre du projet GN-4, que nous allons détailler ci-après, nous avons collaboré avec plusieurs chercheurs et ingénieurs européens dans le but de déployer un service novateur de supervision des tunnels multidomaines qui traversent GÉANT et ses NREN.

Comme nous allons le voir, ce projet est innovant puisqu'il s'agirait du premier dispositif effectuant de

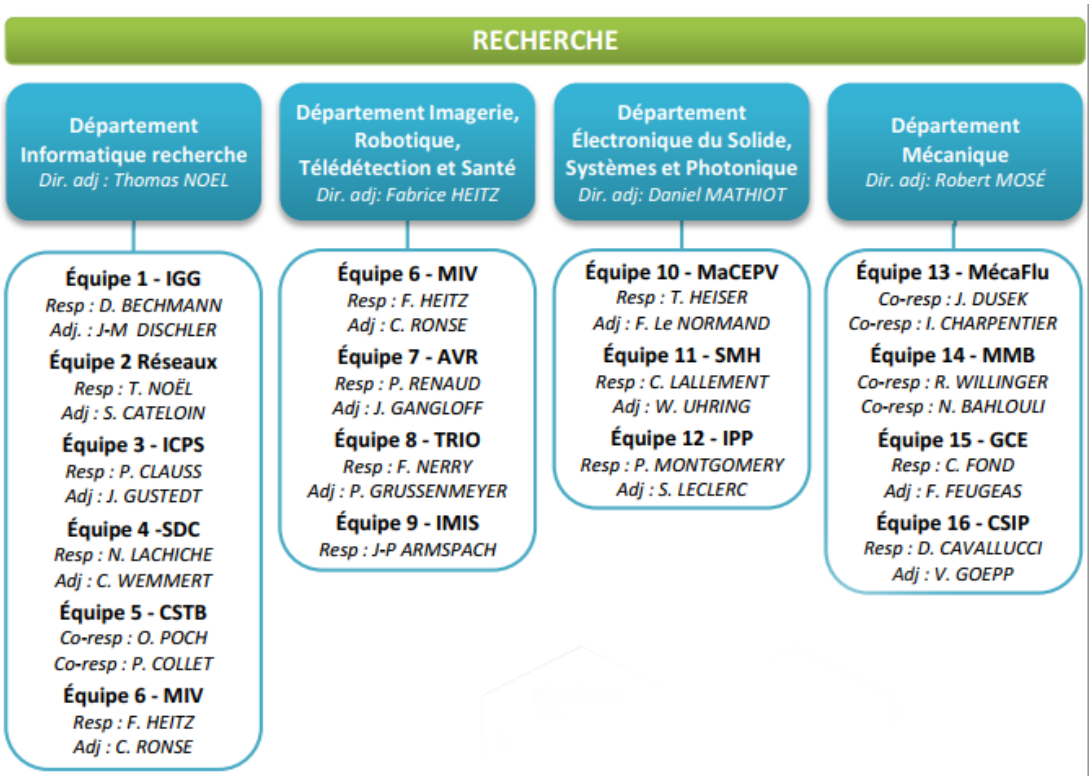

FIGURE 1: Organigramme de la section recherche du laboratoire ICube

la supervision de tunnels multidomaines, et sa conception prend en compte de nombreuses problématiques ouvertes, que ce soit la supervision des tunnels réseaux, ou bien la capture des paquets sur un lien à très haut débit.

Dans le cadre du projet de l'union européenne Horizon 2020 - un projet doté d'un budget de 80 *milliards* d'euros destiné à développer la compétitivité de l'union Européenne en terme de recherche et d'innovation sur le plan international - le fournisseur d'accès GÉANT a lancé le projet GN4-2 (GÉANT Network 4 - Phase 2).

GÉANT est le fournisseur d'accès à Internet paneuropéen, destiné aux réseaux académiques. C'est lui qui interconnecte les réseaux nationaux académiques en Europe, les NREN, et qui leur fournit une connectivité Internet.

Prenons le cas du réseau Osiris comme exemple. Il est connecté aux autres réseaux de campus universitaires de France, comme celui de l'université de Bordeaux, grâce à RENATER, le réseau universitaire national français. C'est GÉANT qui fournit un accès aux autres NREN d'Europe à RENATER ainsi qu'à Internet.

Débuté en mai 2016 pour une durée de 32 mois, le projet GN4-2 a pour but de faire du réseau de GÉANT un réseau de pointe en terme de recherche scientifique. Le projet se décompose en trois parties :

— recherche (JRA) ;
— services (SA) ;
— réseau (NA).

Pascal Mérindol et moi-même travaillons dans le groupe dédié à la recherche appliqué, et plus précisément dans l'équipe JRA2-T4.

Bien qu'il existe différentes implémentations et fonctionnalités, les tunnels ont en général le même objectif : offrir à un utilisateur une connectivité directe vers une site distant, sans avoir à prendre en compte l'existence du tunnel traversé pendant la communication.

Mais mon mémoire traite plus spécifiquement de la supervision des réseaux appliquée aux tunnels *multidomaines*, qui plus est à très haut débit. Comme nous allons le voir, cette application est source de nombreuses

problématiques, que ce mémoire traitera en détail.

Une supervision distribuée.

La supervision des réseaux regroupe l'ensemble des dispositifs et méthodes permettant d'acquérir des informations sur le réseau ciblé. Elle a deux objectifs. Sur le court terme, elle permet de détecter les événements inattendus comme les pannes ou les anomalies. Sur le long terme, elle permet de détecter les tendances globales du trafic sur un réseau afin d'anticiper les futurs besoins en terme d'infrastructures et de services.

Il existe deux catégories de mesures : les mesures passives et les mesures actives. Elles ne sont évidemment pas exclusives, et il peut-être judicieux de combiner les deux au sein d'une même plateforme.

Malgré cela, il est difficile de construire une vision cohérente de l'état d'un réseau en temps réel, surtout s'il s'agit d'un réseau de grande ampleur. Et si des solutions existent pour les réseaux locaux, la supervision des connectivités *multidomaines*, elle, en est encore au stade expérimental.

La première partie de ce mémoire consistera en un état de l'art de la supervision des réseaux, des différentes métriques entrant en jeu, à une présentation des différentes plateformes distribuées déjà présentes sur Internet. Elle nous permettra de mieux comprendre les choix de conception qui ont été faits pendant mon stage.

Les circuits dans les réseaux : une myriade d'architectures.

Les paradigmes du *best-effort* et la commutation de paquets sont des freins à deux tendances majeures dans les réseaux : l'accroissement du nombre de flux nécessitant une certaine qualité de service et l'usage de VPN (Virtual Private Network). C'est pourquoi il est parfois nécessaire de déployer des circuits dans les réseaux. Les types de circuits sont extrêmement diversifiés du fait que l'on recherche à remplir différentes fonctionnalités, que ce soit avec des tunnels point-à-point ou avec des tunnels multipoints (*overlays*). De plus cette diversité de technologies est accrue par le fait qu'il existe des tunnels et *overlays* de niveau 2 et 3.

Dans les deux types de tunnels et *overlays* étudiés, c'est l'opérateur qui déploie le tunnel sur son réseau de cœur. Le deuxième chapitre de mon mémoire se concentre sur ces tunnels, en particulier dans leur extension *multidomaine*.

Une solution novatrice de supervision basée sur la capture de paquets

Le troisième chapitre de ce mémoire présentera tout d'abord les difficultés posées par la supervision appliquée aux tunnels multidomaines haut débit. En effet pour pouvoir évaluer leur performances il faut pouvoir les sonder sans perturber leur trafic. De plus, leur topologie est dynamique, ce qui complexifie l'inférence de l'état de ces tunnels à partir de l'état des liens qu'ils traversent. D'autant que, même si tous les liens que parcourent un tunnel sont fonctionnels, il est possible que le tunnel soit défaillant pour des raisons logicielles.

C'est pourquoi nous avons imaginé le dispositif **PathCap**. Une plateforme distribuée de supervision des tunnels multidomaines. Elle est innovante car elle se base sur de la capture de trafic, une méthode de mesure passive, pour évaluer les performances de ces tunnels.

Nous conclurons ce chapitre, et par la même ce mémoire, par la présentation de l'architecture de la plateforme sur laquelle j'ai travaillé et sur ce qu'a été mon rôle exact dans sa mise en place.

CHAPITRE 1

Supervision des réseaux : de l'évaluation de performances à l'analyse

Sommaire

La supervision des réseaux, qui englobe l'évaluation des performances d'un réseau et la détection d'événements (comme des congestions et des pannes), est un maillon indispensable de la gestion des réseaux informatiques. Le but de mon stage étant de concevoir un dispositif novateur de supervision des réseaux, nous allons présenter en détails la supervision des réseaux dans ce chapitre. Dans un premier temps, nous présenterons quelques considérations générales sur les différentes composantes de la supervision. Puis nous présenterons les principes de la métrologie de réseaux informatiques : les métriques qui permettent de qualifier la qualité d'un réseau, ainsi que les outils élémentaires permettant d'effectuer les mesures. Enfin, nous présenterons les dispositifs de supervision qui existent déjà aujourd'hui sur Internet, afin de mieux positionner notre contribution.

1.1 Considérations générales

Remarque : dans ce mémoire, nous utiliserons indifféremment les termes 'dispositif', 'plateforme' et 'système' de supervision à des fins de confort de lecture. Ces mots sont interchangeables.

La supervision des réseaux fait partie intégrante du cycle de gestion d'un réseau, qui passe par :

1. La conception du réseau. On définit les dimensions du réseau, le matériel employé et les protocoles mis en place afin de répondre à des besoins.
2. La mise en œuvre du réseau : qui consiste en la mise en place du matériel, la connexion des différents éléments du réseau entre eux, et par la configuration du matériel.
3. Le **supervision** du réseau qui permet de visualiser la dynamique d'un réseau et de dégager des tendances sur le long terme.

La supervision joue donc un rôle clé dans le bon fonctionnement d'un réseau, puisque c'est la phase qui permet de d'informer les administrateurs d'un réseau, à court terme et à long terme. A court terme elle permet de réagir en cas d'événement imprévu sur le réseau. En effet si la panne d'un lien entraîne une congestion, par exemple, il est préférable que les administrateurs soient informés au plus vite. Sur le long terme, l'évaluation des performances permet l'amélioration et le redimensionnement d'un réseau. Son rôle est d'anticiper les changement de comportement des utilisateurs du réseau. Ainsi, si par exemple le nombre d'utilisateurs d'un réseau augmente, ou si leur consommation journalière moyenne de données augmente, c'est l'évaluation des performances du réseau qui permet de le déceler. Plus elle est efficace, plus elle reflète avec précision le comportement des utilisateurs d'un réseau.

Objectifs

Les objectifs d'un dispositif de supervision joue un rôle déterminant dans son architecture. Plus un système de supervision est conçu pour répondre à des objectifs ambitieux, plus il faudra mettre de moyens en œuvre et faire preuve d'ingéniosité dans sa conception.

Un système de supervision peut servir tout d'abord à faire de l'**analyse de trafic**. On peut attendre d'un dispositif de supervision qu'il permette de :

— Dresser des statistiques globales d'utilisation du réseau.
— Estimer les besoins en trafic du réseau. C'est à dire classifier les différents composants en fonction de leur taux d'usage, moyen ou de pointe...

— Classifier le trafic et détecter les *patterns* de communication. Comprendre comment le réseau est utilisé permet d'anticiper les évolutions des habitudes de ses utilisateurs.

L'analyse du trafic peut poser de nombreux problèmes. On pense premièrement à la réduction du volume de données collectées et conservées sur le long terme. Mais la compréhension des habitudes des utilisateurs pose aussi des problèmes de classification et d'agrégation des données : comment mettre en place un dispositif qui comprend de lui-même les habitudes des utilisateurs du réseau, voire qui détecte les nouvelles habitudes ?

Ensuite, un dispositif de supervision peut aussi permettre de faire de la gestion de pannes (*fault management*). Là aussi, en fonction de l'efficacité attendue du système, les défis ne sont pas les mêmes. On peut attendre d'un système qu'il permette de :

— Détecter les pannes.
— Localiser les pannes précisément sur le réseau.
— Anticiper les pannes.
— Réagir efficacement aux pannes.

La détection et la localisation des pannes pose des problèmes de gestion des différents points de surveillance, sur leur disposition optimale et sur leur éventuelle redondance. Différents systèmes employant de l'intelligence artificielle ou de la modélisation mathématique permettent d'améliorer grandement la compréhension de la panne et d'y réagir rapidement[19].

`PathCap`, la solution conçue dans mon stage, a pour objectifs de pouvoir localiser une panne dans un tunnel multidomaine avec une petite granularité, c'est à dire en détectant le domaine fautif. De plus, elle devra fournir des statistiques sur le trafic traversant les tunnels en temps réel.

Efficacité, précision et flexibilité : le compromis

Selon une classification de S. Lee *et al.*[26], tout dispositif peut être évalué selon :

— Sa *précision*. Les informations recueillies sont elles à gros grain ou plutôt détaillées ?
— Sa *flexibilité*. Ce dispositif est-il simple à installer et à reconfigurer ?
— Son *efficacité*. La quantité de calculs effectués est-elle pertinente par rapport aux résultats obtenus ?

La plupart des systèmes sont mono-critère. Et même si un dispositif est plus performant qu'un autre sur l'ensemble des critères, il est quand même nécessaire de définir le meilleur compromis.

Mesures actives et passives

Enfin il existe deux catégories de mesures dans les réseaux informatiques. Les mesures *actives* et les mesures *passives*. Une mesure est active quand elle nécessite d'injecter des paquets dans le réseau, dans le cas contraire elle ne l'est pas. Les mesures actives permettent de façon simple de collecter un grand nombre d'informations sur le réseau sondé. Depuis un seul point de mesure, par exemple, on peut mesurer simplement de délais vers d'autres points ou bien explorer les différentes routes existantes vers ce point.

En revanche, les mesures actives, puisqu'elles demandent forcément d'injecter des paquets sur le trafic, on un impact sur le réseau sondé. De plus, les mesures actives permettent assez bien de rendre compte des informations topologiques d'un réseau, en revanche il est difficile d'obtenir des informations sur le trafic circulant dans le réseau juste avec des mesures actives. Les mesures passives sont plus adaptées à ce genre d'objectifs.

La solution que nous avons développée est innovante sur ce point : elle se base sur des mesures passives pour obtenir des informations sur les délais atteints et les routes empruntées par les tunnels *ainsi* que des informations sur le trafic circulant dans les tunnels.

1.2 Métriques de performances et outils de mesure

Un réseau informatique peut être performant de différentes manières. Il peut offrir une connectivité performante à ses utilisateurs, ou bien supporter un grand nombre de connexions simultanément, par exemple. De même il peut garantir d'offrir aux flux prioritaire une connectivité quasi temps-réel. Afin d'évaluer ces différents aspects des performances d'un réseau, il est nécessaire de les quantifier. Ces quantificateurs sont

appelés *indicateurs de performances*. La quantification qui est faite grâce à ces indicateurs permet de mesurer les performances d'un réseau dans le temps, et donc d'en extraire les tendances globales. Les outils de mesure collectent des données sur ces indicateurs. Ils sont davantage destinés aux administrateurs d'un réseau qu'à ses utilisateurs. En effet, les performances d'une connectivité applicative, comme par exemple une connexion HTTP, sont aussi dépendantes d'autres facteurs que les performances du réseau qu'elle traverse. Pour HTTP par exemple, la connexion sera aussi dépendante de facteurs applicatifs comme les performances des serveurs DNS et Web consultés.

Dans cette première partie nous présenterons principalement les indicateurs de performance des réseaux informatiques, et des performances applicatives. Comme nous le verrons par la suite, ils témoignent chacun de différentes propriétés d'un réseau. Avant de les présenter en détail, nous traiterons des problématiques qui sont communes à toutes les mesures de ces indicateurs.

1.2.1 Contraintes communes à toutes les mesures

Tous les outils de mesures sont sujet à deux types de problématiques.

Interprétation des indicateurs : usage des bons outils statistiques

Des problématiques d'ordre mathématique, premièrement. En effet les données collectées et générées par les mesures peuvent être mal interprétées ce qui rendrait les mesures moins représentatives de la réalité dans un réseau.

L'interprétation des indicateurs de performances est un sujet important notamment dans le contexte des *service level agreements* (SLA). Un SLA est un contrat établi entre un fournisseur d'accès Internet et son client, dans lequel le fournisseur garantit qu'il fournira une connectivité répondant à certains critères.

En 2002, l'opérateur télécom américain Worldcomm, pour garantir une bonne connectivité à ses utilisateurs, offrait à ses clients des garanties sur la moyenne mensuelle de mesures de délai effectuées toutes les heures.

Martin *et al.* ont montré que des garanties sur de telles indicateurs protégeaient en fait peu les clients souscrivant au SLA[28]. En effet, une moyenne est un outil mathématique qui peut s'avérer trompeur. De très grands délais peuvent être compensés par de très faibles délais. Or les mesures étaient effectuées de jour comme de nuit, c'est à dire en heure de pointe et en heure creuse.

De plus, il est plus judicieux d'étudier la répartition de différentes mesures en étudiant leur répartition par intervalle en fonction de différents quantiles, et d'utiliser des outils robustes comme par exemple les intervalles de confiance. Cela refléterait mieux les performances des différents catégories de connexion.

Les SLA actuels utilisent d'ailleurs des outils statistiques plus parlants qu'une simple moyenne. *NTT Communications* par exemple offre des garanties de délai et de pertes *maximums* et sur la moyenne des 0.1% délais les plus élevés [1].

Problématiques techniques

Si ces mesures sont donc sujettes à des problématiques mathématiques, elles sont aussi dépendantes de problématiques techniques.

La synchronisation des horloges

Elle est un facteur important de la fiabilité de toute mesure unidirectionnelle. En effet ces mesures se basent sur la date de départ et d'arrivée des sondes, qui dépendent respectivement de l'horloge interne de l'agent d'émission et de celle de l'agent de réception de la sonde. Or, deux horloges peuvent être *décalées* dans le temps ou bien elles peuvent *dévier* différemment, c'est à dire que le temps ne s'écoulera pas de la même manière pour les deux. Pour synchroniser efficacement deux horloges, on peut se baser sur un dispositif GPS (connecté à une antenne fixée) mais ce procédé est relativement onéreux. De manière générale, les dispositifs réseau se basent sur *Network Time Protocole* pour synchroniser leurs horloges. Ce protocole synchronise des

1. https://www.us.ntt.net/support/sla/network.cfm

dispositifs depuis un serveur via le réseau. Il est admis que NTP peut entraîner des désynchronisations de l'ordre de 10 ms au pire, ce qu'il faut prendre en compte lors des mesures[32]. Naturellement il existe des alternatives à NTP qui sont plus récentes comme RADclock [2] ou bien Precision Time Protocol (PTP). En effet, des failles de sécurité ont été découvertes dans NTP [3]

La topologie dynamique des routes

Les outils de mesure des indicateur de performances se divisent en deux catégories. Ceux effectuant des mesures unidirectionnelles, c'est à dire des mesures nécessitant un agent à chaque bout du chemin ciblé, et ceux effectuant des mesures bidirectionnelles, c'est à dire qui ne nécessitent qu'un seul point de mesure. Ce point effectue des mesures sur le chemin en se basant sur un aller-retour.

Or, il est possible qu'un message ne parcoure pas la même route à l'aller et au retour lors d'une mesure, voire même que plusieurs paquets successifs soient acheminés sur des chemins différents lors d'une mesure, notamment en raison de la répartition de charge. En effet, il est courant pour les administrateurs d'un réseau de mettre en place des mécanismes de répartition de charge (*load-balancing*). C'est à dire répartir des paquets sur plusieurs chemins ou plusieurs liens équivalents plutôt que de tous les envoyer sur un seul lien. La répartition de charge peut se faire à différents degrés. Classés selon une granularité croissante, on parle de répartition :

— par *destination*, quand les routeurs assignent à une même route les paquets de même destination.
— par *flux*, lorsque les routeurs identifient les différentes flux, TCP ou UDP, qui les traversent et envoient les mêmes flux sur les mêmes liens.
— par *rafale*, si on répartit différentes rafales d'un flux TCP sur plusieurs chemins équivalents.
— par *paquet*, lorsque les routeurs répartissent les paquets indépendamment de leur destination ou du flux auquel ils appartiennent dans le but d'obtenir la répartition la plus équitable possible.

La répartition par destination et par flux acheminent les paquets d'une même communication sur une même route, ce qui évite des problèmes de déséquencement. La répartition par paquet n'offre aucune garantie mais permet une répartition plus équitable, car elle se fait selon une meilleure granularité.

Lorsqu'un agent de mesure effectue une mesure aller-retour, la mesure peut-être faussée si les paquets de la sonde ne parcourent pas la même route à l'aller qu'au retour, ou bien si plusieurs paquets successifs traversent différentes routes à l'aller. C'est le cas notamment de `traceroute` et `ping`, mais des alternatives permettent de pallier ces problèmes. Nous les détaillerons dans la partie suivante.

1.2.2 Des indicateurs mesurables de différentes manières

Maintenant que nous avons rapidement présenté les problématiques qu'il était important de prendre en compte lors de la mesure d'indicateurs, nous allons les étudier chacun en détail. Cinq indicateurs de performances peuvent aujourd'hui être évalués dans les réseaux IP :

— les taux de pertes ;
— le délai (uni- et bidirectionnel) ;
— les variations de délai (gigue) ;
— le débit et ses variantes ;
— les routes et leurs dynamiques.

Il est difficile d'obtenir des mesures juste et fiables de ces différents indicateurs. Pourtant, correctement mesurés et exploités, ils peuvent chacun être révélateurs de différentes dynamiques dans un réseau donné.

Délai et variations de délai

Le délai représente l'intervalle de temps écoulé entre l'émission et la réception d'un paquet donné. Il y a deux grandes familles de mesures de délai : les mesures unidirectionnelles et les mesures bidirectionnelles, communément appelé *Round Trip Time* (RTT). L'IETF a formalisé la définition du délai unidirectionnel (OWD) de la manière suivante[7] :

2. http://www.synclab.org/radclock/
3. https://www.eecis.udel.edu/~mills/security.html

Lors de la transmission d'un paquet donné entre une source `S` *et une destination* `D`*, le délai unidirectionnel* Δ_t *signifie que le premier bit du paquet a été émis à l'instant* `T` *et que le dernier bit du paquet a été reçu à l'instant* $\mathtt{T} + \Delta_t$

Mesures unidirectionneles

Elles imposent d'être capable d'effectuer des mesures du côté émetteur et du côté récepteur du paquet. L'outil le plus connu est OWAMP. *One Way* `Ping`[4] est un utilitaire destiné à être utilisé en mode teminal. C'est une implémentation du protocole OWAMP défini par la RFC 4656 [37]. Le but de `OWAMP` est d'obtenir des mesures de délai et de taux de pertes *unidirectionnelles* de haute précision. De plus, `OWAMP` se veut le plus discret possible et ce, en simplifiant le protocole au maximum et en n'utilisant que des champs dynamiques, afin que son usage ne soit pas détectable. Une communication `OWAMP` doit se faire entre deux clients `OWAMP`. Dans un premier temps, ils négocient les paramètres de leur communication lors d'une session de contrôle puis ils effectuent leurs mesures lors d'une session de test. `OWAMP` prévoit même la possibilité de chiffrer la communication et de rajouter de l'aléatoire dans la fréquence de l'envoi des tests grâce à une variable de poisson négociée. `OWAMP` a deux spécificités. Premièrement, des mesures `OWAMP` ne peuvent être effectuées que si les deux parties son consentantes. C'est important, car le nombre de sessions générées par un réseau fortement maillé de terminaux augmente de façon quadratique avec le nombre de terminaux. Par conséquent, un grand réseau fortement maillé pourrait imposer une trop grande quantité de requêtes sur des terminaux. Deuxièmement, `OWAMP` se base sur des *horloges synchronisées*. Il utilise NTP (*network time protocol*). Lors de la session de contrôle, les deux terminaux synchronisent leur horloge afin d'avoir des mesures les plus fiables que possible. En l'absence d'horloge synchronisées, `OWAMP` peut tout de même obtenir des résultats pertinents quant à la gigue de la communication. Notons par conséquent que la précision des mesures temporelles effectuées lors de la session dépend de la synchronisation des horloges.

Mesures bidirectionnelles

L'outil de mesure aller-retour par excellence est `ping`. `Ping` envoie des paquets ICMP de type `echo request`, puis mesure le temps écoulé entre l'émission et la réception du paquet `echo reply` correspondant. En 2013, Cristel Pelsser *et al.* ont néanmoins mis en évidence une faiblesse de `ping` qui est d'ailleurs aussi présente dans `traceroute`, comme nous le verrons en 1.2.2. Nous savons que `ping` envoie des paquets successifs pour mesurer le RTT entre deux points. Il est possible qu'une rafale de mesures présente une grande variance qui ne reflète pas la réalité du réseau sous-jacent. En effet, peut-être que les paquets de ces rafales ont été répartis sur différents chemins. Pour pallier ce problème, `tokyoping`[34] est un `ping` avancé qui *forge* des paquets. Les flux UDP sont identifiés par l'adresse source et destination de l'entête IP et par le port source et destination de l'entête UDP, tandis que les flux ICMP sont généralement identifiés par le type, le code et le checksum des paquets. `Tokyoping` envoie des paquets modifiés de sorte à ce qu'ils soient identifiés comme appartenant au même flux par les routeurs pratiquant le *load-balancing*. Le seul cas où `tokyoping` ne permet pas d'éviter une désagrégation des sondes est le cas où les routeurs pratiquent du *load balancing* par *paquet*.

La variation de délai

Appelée communément gigue[5], elle se base sur des mesures de délai. `OWAMP` permet aussi de mesurer la variation de délais sur une route.

Soient deux paquets p_1 *et* p_2 *émis respectivement en* $\mathtt{T_1}$ *et* $\mathtt{T_2}$ *et reçus respectivement en* $\mathtt{T_1} + \Delta_1$ *et* $\mathtt{T_2} + \Delta_2$*. Alors la variation de délai entre ces paquets* `I` *est la différence entre ces deux paquets :*

$$\mathtt{I} = \Delta_1 - \Delta_2$$

4. https://tools.ietf.org/html/rfc4656
5. le terme gigue est déprécié depuis 2002, car trop vague.[17]

On constate que la variation de délai peut-être positive ou négative. Par conséquent, le choix de l'outil statistique doit être fait avec encore plus de précautions. En effet, imaginons les mesures de gigue suivantes, en millisecondes, sur une série de 5 paquets successifs : $[400, -200, 600, -800]$. La moyenne de cette série de variations de délais est **0 ms**. On voit bien que la moyenne est un outil statistique qui ici ne reflète pas bien la réalité. En revanche, l'écart-type de cet ensemble est environ 547.7ms. C'est donc une valeur déjà bien plus représentative des forts écarts de variation mesurés.

Dans les réseaux IP, il est possible qu'un paquet P_1 arrive à destination avant un paquet P_0, bien qu'il ait été émis *après* P_0. En effet rien ne garantit que deux paquets successifs empruntent le même chemin. Dans ce cas, on assiste à un *déséquencement*. Un déséquencement est caractérisé par une variation de délai négative.

Les pertes

Les pertes témoignent de la **fiabilité** d'un réseau. Toute connexion, qu'il s'agisse d'un téléchargement de fichier volumineux ou bien d'une communication audio temps-réel, est dépendant du taux de pertes sur le réseau qu'il traverse. Le groupe *IP performance metrics* (IPPM) de l'IETF propose trois types de mesures qui donnent des informations pertinentes sur les pertes dans un réseau :

— le taux de perte, premièrement. Il s'agit du nombre de paquets perdus divisés par le nombre total de paquets échangés lors d'une communication ;
— la *distance* de pertes, qui représente le nombre de paquets reçus entre deux pertes ;
— la *période* de pertes, qui représente le nombre de paquets perdus d'affilée.

Ces trois mesures permettent de refléter différentes caractéristiques des pertes dans un réseau, comme la longueur moyenne des périodes de pertes et la fréquence moyenne des pertes. `ping` et `OWAMP` permettent d'effectuer des mesures de taux de perte.

Débit

Le débit représente la quantité de données qui peut être transportée en un temps imparti. En qualité de service, il joue un rôle majeur. En effet, même si on ne constate que très peu de pertes sur un réseau, il est important de savoir si il est capable de transporter une grande quantité de données assez rapidement.

Quand on effectue des mesures de débit, on peut s'intéresser soit :

— à la capacité des liens ;
— au débit *résiduel* ;
— à la bande passante *effective* ;
— à la bande passante *applicative*.

La bande passante effective représente la quantité d'information de niveau 3 transportable en un temps donné. Elle diffère de la capacité du matériel de niveau 2 sous-jacent. En effet, un lien de niveau 2 doit encapsuler des données pour les acheminer jusqu'au prochain saut dans un entête matériel. En Ethernet, pour un débit de transmission de 10 Mb/s, la capacité de niveau 3 de ce lien est de 7,24 Mb/s avec des paquets IP de 100 octets contre 9,75 Mb/s avec des paquets de 1500 octets[35].

Le débit résiduel, lui, représente la capacité *restante* sur un lien.

Enfin, la bande passante applicative TCP, appelée en anglais *bulk transfer capacity* (BTC) représente le débit *maximal* que peut atteindre une connexion TCP sur un chemin donné. Cette métrique est spécifique à TCP, et c'est pourquoi elle diffère de la bande passante effective .

Bien sûr, on peut toujours mesurer une bande passant soit sur un lien, on parle de *per-hop capacity* ou bien on peut mesurer la capacité de bout en bout. La capacité de bout en bout est limitée par le saut ayant la capacité la plus faible. En 2007, Dischinger *et al.* ont démontré par exemple que, dans les réseaux d'opérateur destinés aux particuliers, le lien reliant le particulier au réseau d'accès, souvent appelé *last mile link* est le goulet d'étranglement dans la quasi-totalité des cas[18]. Nous allons maintenant présenter différents outils adaptés pour mesurer ces différentes facettes de la bande passante.

Pour la capacité des liens

Il existe `bing`, nommé ainsi en référence à `ping`. Soit une route $A \to R1 \to R2$. Pour obtenir la capacité des liens successifs, `bing` envoie deux paquets ICMP de tailles différentes vers $R1$. Pour estimer la capacité de $A \to R1$, on se base sur la *différence* des RTT. En supposant que les temps de propagation et d'attente dans les files des routeur est le même pour deux paquets successifs, on peut connaître la capacité d'un lien en mesurant combien de temps *supplémentaire* un paquet d'une quantité connue de bits supplémentaire met à faire l'aller-retour. Une fois la capacité de $A \to R1$ trouvée, on envoie la même paire de paquets que précédemment vers $R2$ pour en déduire la capacité de $R1 \to R2$, et ainsi de suite. Les auteurs de `bing` sont conscients des limites de leur outil[1]. Dans certains cas de figure, `bing` ne permet pas d'obtenir des mesures fiables :

— Lorsqu'on essaie de mesurer la capacité d'un lien qui est plus de quinze fois supérieure au lien de plus faible capacité d'une même route.
— Dans le cas d'un routage asymétrique, c'est à dire si le paquet ne parcourt pas le même chemin à l'aller qu'au retour.
— En la présence de tunnels, MPLS ou autre, qui occultent la présence de plusieurs sauts.

Pour la bande passante disponible de bout en bout

Goldoni *et al.* [21] ont réalisé une comparaison en 2010 de neuf outils de mesure de bande passante disponible qui se basent sur différentes techniques. `Pathload`[24], et son dérivé `Yaz` semblent être les outils les plus précis. Ils ne nécessitent pas d'horloge synchronisée mais requièrent de pouvoir prendre des mesures au point d'émission *et* de réception des paquets. Pour mesurer la bande passante disponible, ils envoient un train de paquets de même taille et à fréquence constante. Si, à la réception des paquets, le temps écoulé entre l'arrivée de chaque paquet n'est pas constant mais *croissant*, alors il y a eu congestion quelque part sur le chemin parcouru. Les deux outils se basent sur une recherche binaire pour le choix du débit du train de paquets suivant. L'outil `spruce`[39] présentait lui des résultats précis si le réseau testé était parcouru par peu de trafic tandis que `IGI`[23] était plus performant lorsque le réseau était plus occupé. Contrairement à `pathload`, ces deux outils n'essaient pas de déclencher une congestion dans le réseau. Ils se basent juste sur l'intervalle entre des paires de sondes et déduisent la bande passante correspondante. Ces techniques sont donc moins coûteuses en bande passante émise.

Pour la bande passante effective

`Iperf`[4], un utilitaire américain conçu par des ingénieurs et des chercheurs de ESnet et Lawrence Berkeley, est largement utilisé. Pour tester la bande passante effective, il est recommandé de d'abord tester la capacité des liens ainsi que la bande passante disponible de la route en question[14]. Ensuite, `iperf` permet de réaliser des mesures TCP et UDP sur différents paramètres spécifiques aux protocoles : taux de pertes et gigue pour UDP, BTC pour TCP. On peut aussi mentionner `Nuttcp`. `Nuttcp` permet de faire des mesures actives de bande passante en UDP et en TCP, et d'obtenir des informations comme l'usage CPU du destinataire et le *wall-clock time*, donnée indiquant le temps total du transfert, tel qu'il serait perçu par l'utilisateur. C'est à dire, depuis le moment où l'utilisateur ordonne l'envoi des données jusqu'au moment où on lui indique que le transfert est terminé.

La découverte des routes

Enfin, la troisième grande catégorie de mesures que l'on pourrait vouloir effectuer dans un réseau concerne *l'intégrité* des communications, c'est à dire l'impact que la topologie du réseau et les évènements peuvent avoir sur les paquets véhiculés.

La découverte des routes est une composante importante de la métrologie d'Internet. Elle permet, par exemple, de faire du déboggage et peut aider à mieux localiser une défaillance dans un réseau, mais elle est aussi une pierre angulaire de la cartographie d'Internet. L'outil de découverte de route le plus répandu est l'utilitaire `traceroute`. Pour découvrir les différents routeurs que traverse une route, `traceroute` joue sur le champ *time to leave* (TTL) des paquets IP. Pour rappel, le TTL d'un paquet est un entier qui est décrémenté

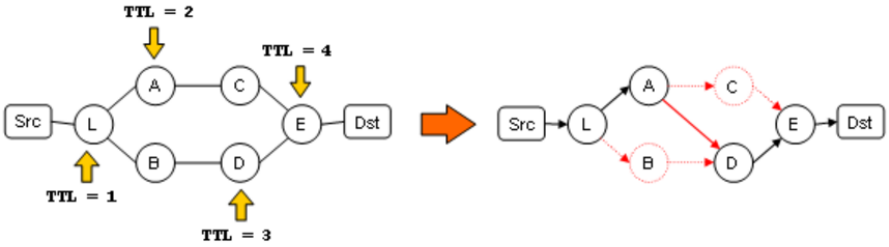

FIGURE 1.1: Cas où `traceroute` pourrait annoncer l'existence d'un lien, à tort[8]. `traceroute` détecte la présence de la mauvaise route

à chaque fois qu'un paquet traverse un routeur. Si un routeur décrémente le TTL à 0, ce paquet est jeté et l'émetteur du paquet est prévenu grâce à un message ICMP `ICMP Time exceeded`. Ce mécanisme permet de débarrasser le réseau des paquets bouclant indéfiniment. `Traceroute`, lui, détourne l'usage du TTL : il envoie une successions de paquets en *incrémentant* leur TTL. Ainsi, `traceroute` recevra un `ICMP Time exceeded` de la part de chacun des routeurs présents sur la route, et pourra ainsi les identifier.

Néanmoins, il a été montré que `traceroute` pouvait souffrir de certaines faiblesses. En effet `traceroute` n'est pas adapté au mécanismes de répartition de charge qu'utilisent les routeurs aujourd'hui. Comme nous l'avions déjà mentionné avec `tokyoping` en 1.2.2, il est possibe qu'avec les mécanismes de *load-balancing*, une rafale de paquets soit répartie sur plusieurs routes parallèles. Par conséquent il arrive que `traceroute` croient que deux routeurs se succèdent, alors qu'ils sont situés sur deux routes parallèles . Cette situation est illustrée figure 1.1. C'est pourquoi Brice Augustin *et al.* ont conçu `paris-traceroute` [9]. Pour éviter qu'une rafale de paquets à TTL incrémental soit distribuée sur deux routes équivalents, `paris-traceroute` forge des paquets qui sont plus prompt à être aiguillés sur le même chemin, en les faisant appartenir au même flux. C'est le même procédé qui est utilisé dans `tokyo ping`. `Paris-traceroute` est donc efficace dans le cas d'une répartition de charge, tant qu'elle reste par flux ou destination et non pas par paquet, ce qui est vrai dans 98% des cas d'après les concepteurs de `paris-traceroute`[10].

Le tableau 1.1 présente succinctement les indicateurs de performance des réseau IP. Chacun peut-être représenté par des fonctions objectives au propriétés mathématiques différents. Certaines sont multiplicatives, comme les pertes, tandis que la capacité de bout en bout sur un lien peut-être modélisé par un minimum de la capacité des liens successifs. La dernière colonne du tableau représente l'objet analysé par l'indicateur de performances, ainsi que certains commentaires.

Il était indispensable d'avoir une vision globale des différents indicateurs de performance et leurs outils dans le cadre de mon stage. En effet, nous avons déployé une *plateforme* de supervision, qui utilise ces différents outils.

1.3 Plateformes de supervision

1.3.1 Évaluation des performances depuis les feuilles

Il existe de nombreux moyen de superviser les réseaux locaux et d'évaluer leurs performances. Très utilisés dans les réseaux d'entreprises, des solutions comme *Nagios* sont des dispositifs qui permettent de détecter les anomalies et les incidents en temps réel. Dans le cadre de mon stage, nous n'étudierons pas ces solutions. Nous allons plutôt nous intéresser aux dispositifs d'évaluation de performances et de supervision de réseaux de grande envergure, comme les réseaux de cœur. En effet c'est à cette catégorie de réseaux qu'est destinée notre plateforme de supervision.

Pour faire de l'évaluation de performances, il faut tout d'abord considérer le point de vue duquel on souhaite observer le réseau. Pour Internet par exemple, on peut collecter une quantité d'information très conséquente en ne collectant que les données du point de vue utilisateurs. C'est ce que nous verrons dans cette section. En annexe A, nous présentons Netradar, une plateforme d'évaluation de performances destinée

Indicateur	Métrique	Outil(s) de mesure	Objects analysés
Délai	$+, \mathbb{R}_{\geq 0}$	`(tokyo-)ping, owamp`	chemins ECMP aller/retour chemins dirigés (synchro ?) vs. liens/segments dirigés
Gigue Déséquencement	$+, \mathbb{R}$	`QoSMet`	variations délais (n,n+1) (dé)composition bout en bout approximative / dégradé si pas de numérotation interne
Taux de pertes/succès	*	Idem délai/gigue	Même calcul que Délai et gigue
Capacités	min	`pathload`, `iperf`, `nuttcp, bing`	brut/résiduel vs. utile/applicatif tcp vs udp saut par saut aller/retour ?
Topologie	Composition	`(paris-) traceroute`	Changement de chemins

TABLE 1.1: Récapitulatif des métriques des réseaux informatiques et de leurs outils de mesure

aux réseaux mobiles.

SamKnows : un boîtier entre le client et le fournisseur d'accès

Fondée en 2009 par Sam Crawford, SamKnows [15] est une entreprise travaillant étroitement avec le monde de la recherche pour établir des statistiques sur Internet. Son principe de fonctionnement est simple : des utilisateurs répartis géographiquement dans le monde se font livrer des boîtiers prêts à l'emploi - les *whiteboxes* - qu'ils connectent en Ethernet au boîtier électronique de terminaison de ligne de leur fournisseur d'accès à internet (FAI), et ce *entre* leurs ordinateurs et autres terminaux et le boîtier du FAI. Aujourd'hui, SamKnows a déployé plusieurs centaines de milliers de *Whiteboxes* à travers le monde, et peut ainsi surveiller les réseaux de 36 FAI.

SamKnows effectue du sondage **actif** : les *whiteboxes*, dont le *firmware* est distribué sous licence GPL, effectuent le plus régulièrement que possible des mesures *de bout en bout*. Cela signifie qu'elles envoient des données à des serveurs dédiés qui leur permettent de tester un grand panel de métriques. Elles collectent bien sûr des informations générales sur le réseau : délai de bout en bout, délai en heure de pointe, débit de téléchargement et de téléversement, gigue. Mais aussi des informations applicatives, puisqu'elles effectuent des tests sur la voix sur IP, le *Peer to peer*, les résolutions DNS, le *streaming* vidéo et encore d'autres types de protocoles. Pour ne pas gêner le trafic des utilisateurs avec ses sondages actifs, la *whitebox* effectue de la surveillance *passive*. Elle effectue des sondages quand le réseau local est peu actif.

SamKnows a permis de nombreuses découvertes sur différents aspects d'Internet. Par exemple, les travaux de Zachary S. Bischof *et al.* ont montré que les les données générées par les réseaux de P2P BitTorrent pouvaient permettre de déduire des informations pertinentes sur la bande passante et le délais des réseaux IP sous-jacents [12].

BISmark : du matériel plus conséquent

BISmark (Broadband Internet Service Internet Benchmark) est une initiative de l'université de Georgia Tech. Il s'agit de plus d'une centaine routeurs Netgear avec un OS modifié, au code source libre. Contrairement à SamKnows, ces routeurs permettent de réaliser une plus grande quantité de tests, puisqu'ils font aussi de l'évaluation *passive*. Alors que SamKnows se basait sur des `ping` et des `traceroute` pour réaliser ces mesures, BISmark se sert pour le sondage des outils suivants :

— `paris-traceroute`.

— `shaperprobe`, qui détecte si l'opérateur limite le débit maximum offert à l'utilisateur ;

— `mirage` dédié au test de requêtes HTTP ;
— `iperf` ;

Les ingénieurs de BISmark effectuent du sondage passif, car ils observent les paquets transitant dans leurs équipements. Cela peut poser des problème concernant l'atteinte à la vie privée des utilisateurs. Comme l'ont d'ailleurs mentionné les auteurs de l'étude empirique des habitudes des particuliers sur Internet [22], cela a été laborieux. Il n'ont pu le faire que sur les sondes posées sur le sol américain, avec l'accord écrit explicite des utilisateurs et l'accord de l'*institutional review board*, après avoir pris de très nombreuses précautions en anonymisant les données et en ne prélevant qu'une partie d'entre elles. Néanmoins, seules des données générées par les utilisateurs auraient permis par exemple de connaitre les sites les plus visités sur Internet dans chaque foyer.

Il est prévu que le dispositif étudié dans le cadre de mon stage effectue aussi de la capture de trafic.

1.3.2 Les solutions présentes *à l'intérieur* des réseaux de cœur

RIPE Atlas : des sondes *harwares* pour une surveillance de grande envergure

RIPE Atlas fournit des sondes matérielles qui sont déployées par les opérateurs réseaux dans le monde entier, mais aussi chez des particuliers. RIPE Atlas compte environ *12 mille* sondes réparties sur les cinq continents du globe. 6 % des AS dans le monde sont couverts en IPv4 et 11% en IPv6 [3]. Il s'agit là de valeurs relatives.

Bien qu'il ait vu le jour en 2014, RIPE Atlas est en réalité le descendant d'un projet plus ancien, RIPE TTM, qui lui a vu le jour en 1997. Par conséquent, les modèles de sondes sont divers et dépendent de leur date de mise en place. RIPE Atlas se base sur deux types de dispositifs : les *sondes* RIPE Atlas, des petits dispositifs sur lesquels sont installés une distribution Linux dédiée aux systèmes embarqué - OpenWRT - et les *points d'ancrage* RIPE Atlas, des serveurs dédiés qui sont les cibles des mesures actives effectuées par les sondes.

Les sondes RIPE Atlas ne font que des mesures actives. Elles effectuent des mesures de RTT, `traceroute` et des requêtes HTTP et produisent tous leurs résultats au format JSON. Elles se basent sur planificateur de tâches `eperd`, qui est très proche de `cron` pour effectuer leurs mesures de manière périodique. `Eperd` dispose de fonctionnalités supplémentaires par rapport à `cron`. Il peut ajouter un délai aléatoire avant d'effectuer un événement, ce qui évite les congestions dues à une trop grande quantité de sondes effectuant leur mesures *exacement* en même temps.

RIPE Atlas est un pilier de la recherche sur Internet en général. Il a été utilisé pour une très grande quantité de recherches au cours du temps. Néanmoins, bien que cette plateforme permette d'effectuer des mesures depuis le cœur des réseaux, elle est uniquement dédiée à l'évaluation de performances.

perfSONAR : la solution d'évaluation logicielle en interne

PerfSONAR permet d'effectuer de la **supervision** de réseau. En plus de faire de l'évaluation de performance, cette plateforme a pour but de détecter les défaillances réseaux en temps réel. PerfSONAR est une initiative émanant de plusieurs FAI académiques : GÉANT, Internet2, Energy Sciences Network, et le "*Brazil's National Education and Research Network*" (RNP). Son but est de déployer un dispositif d'évaluation de performances mondial destiné à la surveillance des réseaux dédiés à l'échange de données scientifiques. PerfSONAR est une solution logicielle qui se décline désormais en un grand nombre de variantes. Toutes néanmoins sont basées sur le même principe : pour la déployer, il faut installer un OS dédié sur la machine destinée à prendre des mesures. PerfSONAR est déployé sur CentOS, une distribution Linux. Aujourd'hui, PerfSONAR compte plus de 14 000 services web distribués sur plus de 2000 machines réparties à travers le monde[6] comme le montre la carte figure 1.2. En fonction des besoins, les différentes variantes de perfSONAR peuvent ou non permettre de :

— lancer des sondages périodiquement ;
— contrôler la machine à distance via un service web ;
— stocker les données localement ou bien les envoyer à un point central périodiquement [5].

FIGURE 1.2: Carte localisant les sondes perfSONAR déployées en janvier 2017.

Depuis, d'autres projets basés sur perfSONAR ont vu le jour : perfSONAR *Performance Toolkit* (Perfsonar-PS) et PerfSONAR *Multi-Domain Monitoring* (PerfSONAR- MDM). PerfSONAR PS était historiquement utilisé depuis 2008 pour la surveillance d'une partie de la *Worldwide LHC Computing Grind* (WLCG), une grille dédiée au stockage et à l'étude des données générées par les expériences réalisées sur le *Large Hadron Collider* (LHC). En effet les accélérateurs de particules ont la spécificité de générer une quantité conséquente de données critiques. Le LHC a généré entre 2012 et 2015 environ 75 *petaoctets* (10^3 To). PerfSONAR-MDM, de son côté, est utilisé par GÉANT et regroupe 60 stations de sondages disséminées majoritairement dans GÉANT mais aussi dans les réseaux ESnet et Internet2. Depuis 2010, PerfSONAR PS et MDM sont interopérables.

PerfSONAR effectue aussi bien de l'évaluation active que passive. PerfSONAR est un dispositif conséquent, et PerfSONAR PS et MDM utilisent des outils différents.

PerfSONAR PS se base sur :

— *Bandwidth test controller* (`bwctl`), qui est un utilitaire offrant des commandes terminal ainsi que la possibilité de planifier des tâches et de gérer des démons. `Bwctl` regroupe Iperf, Nuttcp, `ping`, `OWAMP` et `traceroute`. Comme tous les outils permettant d'effectuer des mesures de délai unidirectionnel et de bande passante effective, `bwctl` nécessite des serveurs dédiés auxquels les clients se connectent.

— `PingER`[30], pour les mesures de délai et de gigue. `PingER` est une infrastructure déployée en 2002 dont le but est de réaliser des mesurer de délais et de pertes périodiques. La particularité de `pingER` réside dans sa manière de faire des mesures : `PingER` envoie 11 pings de paquets de 100 octets espacés de une seconde, suivi de 10 pings de paquets de 1Ko eux aussi espacés de 1 seconde. Il envoie 11 paquets dans le premier train afin d'ignorer le premier, dans le cas où des mécanismes de mise en cache le ralentiraient. `PingER` envoie des paires de trains de pings à une série de nœuds enregistrés préalablement dans un fichier en dur.

PerfSONAR MDM, quant à lui, utilise les outils :

— Bwctl MP, un service utilisant `bwctl` et qui stocke les résultats des mesures dans une base de données SQL.

— Hades Active Delay Evaluation System (HADES), un dispositif responsable de la prise et du stockage de mesures de délai unidirectionnel, gigue et de découverte de routes sur le réseau GÉANT. HADES est un système dont les mesures temporelles atteignent une précision record de $4\ \mu s$. Pour cela, les sondes effectuant les mesures sont équipés de deux dispositifs matériels, l'un connecté à un système GPS et l'autre connecté à DCF77. [6]

Pour faire de la supervision de réseaux, PerfSONAR collecte en plus des information sur les routeurs grâce à SNMP. PerfSONAR réalise donc de la supervision grâce à un accès privilegié aux équipements réseaux.

6. DCF77 est un dispositif mis en place par la *Physikalisch-Technische Bundesanstalt*, une agence nationale de métrologie allemande, afin de diffuser l'heure sur les ondes radio. DCF77 se base sur une horloge atomique au césium, type horloge dont les modèle les plus performants dérivent d'une seconde toutes les 160 millions d'années.

C'est la raison pour laquelle aucun des autres dispositifs présentés dans ce mémoire ne fait de supervision, au sens où il se contentent d'évaluer les performances du réseau sondé, sans pour autant chercher à détecter de défaillance.

Dans ce chapitre, nous avons étudié l'évaluation de performances et la supervision des réseaux de grande envergure. Plusieurs problèmes surviennent lors de la conception d'un tel dispositif. Premièrement, il existe un grand nombre d'indicateurs de performances dans les réseaux, qui reflètent différentes aspects du réseau. Il faut en plus faire attention au traitement statistiques des données générées par ces mesure, car nous avons vu qu'un mauvais traitement peut donner une vision faussée de la dynamique du réseau.

Pour effectuer les mesures, il existe une grande panoplie d'outils. Afin d'obtenir les mesures les plus pertinentes sur un réseau, il est intéressant de démultiplier les points de mesure ainsi que les types de mesures effectuées. L'agrégation des ces différents outils donne naissance à des plateforme de supervision des réseaux.

Les plateformes présentées dans ce chapitre permettent jusqu'à la détection de défaillances dans un domaines grâce à des mesures passives, *via* SNMP. Dans le cadre de mon stage, nous avons été amené à concevoir une solution allant encore plus loin. L'objectif serait d'obtenir des statistiques sur le trafic ayant réellement circulé dans des tunnels multidomaines. Pour ce faire, notre solution effectuera de la capture d'un grand volume de trafic en temps réel. Comme nous le verrons dans le dernier chapitre de ce rapport, c'est une première, qui a mené à l'identification de nombreux verrous scientifiques.

CHAPITRE 2

Services réseaux : circuits et *overlays*

Sommaire

Dans le premier chapitre, nous avons étudié les problématiques que posaient la supervision et l'évaluation de performances des réseaux, qui sont *inhérentes* à la structure des réseaux. Comme un réseau informatique est distribué, qu'il est composé d'équipements hétérogènes et que le type de trafic qui y circule varie énormément en fonction du réseau, les dispositifs de supervision varient en fonction des résultats attendus et des moyens à disposition. Néanmoins il y a désormais un consensus dans la littérature scientifique sur les différentes métriques que l'on peut prendre en compte lorsqu'on effectue des mesures sur Internet ou dans d'autres réseaux de plus petite échelle[33]. De plus, de nombreux outils libres permettent d'effectuer ces mesures. Mais comme nous l'avons vu par exemple avec `traceroute` et `ping`, il est difficile de prouver leur précision et leur efficacité. À l'échelle d'Internet, les plateformes d'évaluation de performances les plus abouties ont demandé des années de déploiement et malgré cela, il est extrêmement difficile d'avoir une appréciation globale du trafic circulant dans les réseaux de cœur d'Internet.

Une difficulté supplémentaire s'ajoute dans le cadre de mon stage : non seulement je travaille sur la mise en place d'un système de supervision d'un réseau dont certains liens ont des débits de l'ordre de la *centaine* de gigabits par seconde, mais en plus je vais travailler plus précisément sur la supervision de *tunnels* à large échelle sur le réseau de GÉANT et ses NREN.

Dans ce chapitre, nous présenterons les besoins existants en services réseaux et les différents moyens d'y répondre. Puis, nous verrons le fonctionnement des solutions existantes de mise en œuvre de tunnels et de réseaux privés virtuels. Enfin, nous conclurons par le service offert par GÉANT et ses NREN : GÉANT MD-VPN.

2.1 Définitions et objectifs

Dans cette partie, nous verrons dans un premier temps les raison qui pourraient pousser les opérateurs d'un réseau à mettre en place des tunnels et des réseaux privés virtuels. Puis, nous détaillerons les mécanismes qui sont communs aux différents types de tunnels et de Virtual Private Network (VPN). Cela nous permettra de mieux appréhender les différences et les subtilités des différentes technologies permettant la mise en place de tunnels et de VPN, que nous détaillerons plus en profondeur dans les sections suivantes.

2.1.1 Grandeur et décadence du *best-effort*

Comme nous allons le voir, les réseaux IP ne permettent pas de répondre à certains besoins des utilisateurs. Premièrement, le principe du *best-effort* rend difficile la mise en place de garanties en terme de qualité de service. Deuxièmement, il n'est pas possible de concevoir un réseau étanche distribué dans différentes parties d'un même réseau en se basant simplement sur les mécanismes de routage IP.

Augmenter la résilience des réseaux, au détriment de la qualité de service

Les communications sur Internet, et plus généralement sur les réseaux IP, fonctionnent sur le principe de la *commutation de paquets* : les données échangées sont divisées en fragments - les paquets - et ces paquets sont envoyés sur le réseau sans qu'il y ait eu besoin en amont d'établir de route. Au lieu de cela, à la réception d'un paquet, les routeurs décident dynamiquement de la route qu'empruntera le paquet en fonction de sa destination. D'autres réseaux fonctionnent sur le principe de *la commutation de circuits*. C'est par exemple le mode de fonctionnement qu'utilisaient les réseaux téléphoniques pendant longtemps. Lorsqu'un réseau se base sur *la commutation de circuits*, un circuit est établi en amont de la communication. Les données envoyées suivent le circuit préétabli lors de leur acheminement.

Les deux modes de communication présentent chacun leurs avantages. La commutation de circuits demande moins de temps de traitement lors de l'acheminement des paquets. Les réseaux ferrés sont un bon exemple : lorsqu'un train se déplace entre deux destinations, la route qu'il emprunte est déterminée avant son déplacement. Ainsi, il n'est pas nécessaire d'arrêter le train à chaque intersection du réseau ferré pour décider de la meilleure route à emprunter. Dans les réseaux IP, en revanche, les opérateurs ne peuvent pas anticiper avec précision la quantité d'information qui circulera sur leur réseau, ni les caractéristiques des différentes communications. C'est pourquoi les paquets sont acheminés selon un *démultiplexage probabiliste*. Le processus du choix de la route est rééffectué pour chaque paquet reçu pour chaque routeur : la charge imposée est certes un peu plus conséquente, ce qui d'ailleurs est de moins en moins problématique avec l'amélioration des performances globale des équipements réseaux, mais le gain en fluidité est important. De plus, les réseaux IP ont une topologie très fluctuante. Les liens tombant en panne fréquemment [27], il est important que les chemins empruntés par les communications soient adaptés dynamiquement aux changements topologiques. D'ailleurs, les réseaux à commutation de paquets avaient été historiquement inventés par Paul Baran [11] afin de concevoir un réseau efficace malgré l'éventualité de la destruction d'une partie du réseau suite à une offensive militaire.

Malheureusement, la commutation de paquets pose deux problèmes majeurs, qui sont inhérents au paradigme. Premièrement, dans les réseaux IP, les routeurs effectuent de l'acheminement "*best-effort*". Cela signifie qu'il n'est pas garanti qu'un paquet atteigne sa destination. Certes la couche transport, et en particulier TCP, et d'autres corrections applicatives pallient ce manque, en détectant en aval quels paquets ne sont pas arrivés à destination et en les réémettant, mais cela reste moins efficace qu'un réseau fiable. Deuxièmement, dans un contexte "*best-effort*", il est impossible de garantir à un utilisateur que les débits, ou la bande passante, auxquels il aimerait prétendre seront toujours disponibles. En effet, pour garantir l'accès à une fraction de la capacité d'un chemin, il faut pouvoir mettre en place des circuits.

Or le trafic sur Internet nécessite aujourd'hui plus de bande-passante utile que dans les débuts d'Internet : d'après l'opérateur télécom canadien Sandvine [2],**70%** du trafic internet généré par les utilisateurs particuliers appartient désormais à la catégorie dite du "divertissement temps réel", c'est à dire du *streaming* de vidéo. Que ce soit pour ce trafic, ou bien pour les communications temps réel, le *best-effort* pose de réelle limitations puisque retransmettre des paquets perdus est parfois inutile, voire impossible, dans ces communications.

Le besoin en étanchéité

Pour des raisons de sécurité et de simplicité de gestion, les entités administratives, qu'il s'agisse d'entreprises, de campus universitaires ou d'administrations publiques peuvent avoir besoin d'*étanchéifier* leurs réseaux. Cela signifie que, même si ces entités disposent de différents sites distants géographiquement reliés entre eux par un réseau de cœur sur lequel elles n'ont pas la main, elles veulent mettre en place un *réseau privé virtuel* : un VPN.

Les VPN, afin de masquer la présence d'un réseau intermédiaire à deux agents de deux sous-réseaux distants, se basent sur la mise en place de *tunnels*. Un tunnel est une entité logique qui peut-être implémentée de différentes façons mais dont le fonctionnement est toujours le même : ils sont définis par un point d'entrée et un point de sortie et leur présence est transparente à l'utilisateur. Cela signifie qu'un agent souhaitant communiquer avec un autre agent situé de l'autre côté du tunnel ne connaît pas les détails de de ce tunnel. Les tunnels peuvent être définis sur différentes couches du modèle OSI, en fonction des besoins. Les tunnels applicatifs sont aujourd'hui très répandus. On pense par exemple aux tunnels SSH, qui permettent à un utilisateur de chiffrer une communication entre deux agents. Ces tunnels ne rentreront pas dans le cadre de ce rapport, puisqu'ils font complètement abstraction de la couche réseau. Nous traiterons des tunnels qui permettent la mise en place de VPN ou qui ont une incidence sur l'acheminement des données.

Dans le cadre de mon stage, je participe à la mise en place de dispositifs de *monitoring* de tunnels sur le réseau de GÉANT et sur les réseaux de ses utilisateurs. C'est pourquoi il est important de présenter d'abord en détail les différents types de tunnels présents dans GÉANT et leur fonctionnement.

2.1.2 Alternatives au *best-effort*

DiffServ : de la qualité de service contre une inégalité de traitement

Nous avons donc vu qu'il est extrêmement compliqué de garantir une certaine qualité de service à un utilisateur dans un réseau IP et aussi qu'il est compliqué de montrer que la qualité de service promise est effectivement respectée. Or, les utilisateur d'Internet sont exigeants sur la qualité des services à laquelle ils aspirent. En 2013, Krishnan et Sitaraman ont étudié les données fournies par le Content Delivery Network (CDN) Akamai, un des plus gros hébergeur de contenu Internet. Il est responsable d'environ 15 à 30% du trafic web [25] mondial. Ils se sont plus particulièrement penchés sur 23 millions de visionnages de vidéos aux USA par plus de 6 millions d'utilisateurs et ont déterminé que les utilisateurs commençaient à abandonner le visionnage d'une vidéo si elle mettait plus de 2 secondes à démarrer et que chaque seconde la probabilité d'abandonner augmentait de 5,8%.

Étant donné l'importance d'une bonne qualité de service, S. Blake *et al.* ont proposé de modifier le comportement des réseaux IP et de ne plus imposer un acheminement homogène à toutes les communications du circuit. Ils ont conçu une nouvelle architecture pour des services différenciés : *DiffServ*[13]. Le but de cette architecture est de mettre en place une classification de flux en modifiant le champ `DSCP` des entêtes des paquets. Les flux sont classifiés dans différentes catégories plus ou moins prioritaires. Puis les routeurs traitent ces catégories de manières différentes. Cette différenciation est paramétrables par les administrateur du réseau.

Cette architecture a l'avantage de prendre en compte les différences propres au besoins des flux. Une communication temps réel devrait, dans l'idéal, toujours passer devant un transfert de fichier dans les files des routeurs. *DiffServ* offre simplement un mécanisme de classification de flux et des règles de conception de réseau l'implémentant. En effet, les auteurs recommandent de laisser les routeurs de bordure s'occuper de la classification des paquets, et les routeurs de cœur doivent se contenter d'acheminer en fonction de l'entête `DSCP`.

Vers le court-circuitage du routage intradomaine

Les réseaux informatiques sont régis par des mécanismes de routage pour assurer la commutation de paquets. Grâce à différents protocoles de routage, qui sont présentés en détail en annexe B, les routeurs déterminent un plan de *contrôle* qui leur sert à déterminer le meilleur chemin pour chaque paquet en fonction de la topologie du réseau à un instant donné.

DiffServ permet de répondre à un des besoins en qualité de service existants : la différenciation de services. Grâce à *DiffServ*, un administrateur peut influer sur les mécanismes décisionnels de routeurs en faisant de la *classification de flux*. Néanmoins, *DiffServ* ne change pas le paradigme de la commutation de paquets. Afin d'obtenir la plus haute fiabilité possible et de diminuer la charge imposée sur les routeurs par le processus décisionnel enclenché lors de la réception d'un paquet, il existe différents mécanismes permettant de mettre en place des circuits sur un réseau. Ces circuits permettent de répondre aux besoins en bande passante et aux

besoins en étanchéité qu'ont certains utilisateurs dispersés sur un réseau. Nous allons maintenant détailler les différents mécanismes entrant en jeu dans la mise en place de ces tunnels.

L'exécution des circuits

Elle s'effectue aujourd'hui généralement grâce au protocole Multi Protocol Label Switching (MPLS). Par exécution, on entend l'acheminement des paquets en suivant les circuits qui ont été établis en amont. Le protocole MPLS est un protocole très simple : il offre la possibilité d'insérer des étiquettes de 32 bits *entre* les entêtes de niveau 2 et 3 d'un paquet. Ces étiquettes peuvent s'empiler. Elles comportent un identifiant de 20 bits, un TTL, un champ réservé à la qualité de service et un booléen appelé *bottom of stack* indiquant si cette étiquette est la dernière avant l'entête IP ou non.

Comme MPLS se place dans un contexte de commutation de circuits, les opérateurs connaissent les différents circuits du réseau. Ils disposent de tables de commutation MPLS, dont le processus de consultation de déroule en 3 étapes. D'abord, le routeur détermine à quelle Forwarding Equivalence Class (FEC) le paquet appartient. Une FEC est une entité logique qui décrit tous les paquets empruntant un même circuit, généralement vers une même destination. Une fois la FEC du paquet déterminée, le routeur effectue l'action correspondante inscrite dans la table qui consiste soit en l'apposition d'une nouvelle étiquette, un *push*, soit au remplacement d'une étiquette, un *swap*, soit à la suppression de l'étiquette lue, un *pop*. Enfin, après avoir effectué les actions nécessaires au niveau de l'étiquetage, le routeur achemine le paquet sur l'interface de sortie correspondant à la FEC déterminée.

Trois types d'agents interviennent dans l'exécution d'un chemin MPLS :

— Le routeur d'entrée du domaine, le routeur *ingress*. C'est lui qui applique la toute première étiquette MPLS au paquet. Pour ce faire, il détermine à quel circuit le paquet est destiné, grâce aux FEC de sa table de routage, puis il pose l'étiquette correspondante. Enfin, il pousse le paquet sur l'interface de sortie appartenant au circuit.

— Les routeurs intermédiaires, les *label switch router* (LSR), situés au cœur du domaine. Lorsqu'ils reçoivent un paquet étiqueté, eux se contentent de lire l'étiquette MPLS du paquet, et ne réalisent pas d'opération IP. Généralement, ils remplacent l'étiquette consultée par l'étiquette correspondant au saut suivant sur le circuit, puis ils envoient le paquet sur l'interface correspondante.

— Le routeur de sortie, appelé routeur *egress* enlève la dernière étiquette de la pile et achemine le paquet selon son adresse IP de destination. Il ne fait plus d'acheminement par circuit.

Il est possible que le routeur situé juste avant le routeur *egress* soit chargé d'enlever toutes les étiquettes MPLS avant de transmettre le paquet au routeur de sortie. C'est ce qu'on appelle du *penultimate hop popping* (PHP). Le PHP permet de mieux répartir la charge entre le routeur de sortie est les routeurs situés juste avant.

Un chemin MPLS est appelé Label-switched Path (LSP). Il s'agit donc d'un circuit défini par l'étiquette d'entrée, appliquée par le routeur *ingress*.

La mise en place des circuits

Elle est complètement indépendante de leur exécution. Elle peut se faire de manière statique, les administrateurs doivent dans ce cas configurer chaque routeur, ou bien automatisée. Il existe deux protocoles différents destinés à cette fin : Label Distribution Protocol (LDP) et Ressource reSerVation Protocol (RSVP-TE). Les deux protocoles visent à déployer des chemins d'étiquettes MPLS, des LSP.

LDP est très simple à mettre en place. Une fois lancé , chaque routeur annonce son existence et l'étiquette permettant de le rejoindre à ses voisins, qui eux-mêmes ajoutent cette étiquette à leur table de routage MPLS et repropagent la destination qu'ils viennent d'apprendre avec l'étiquette MPLS qui mène jusqu'à eux. Les routes se propagent ainsi à la manière d'un protocole à vecteur de chemins, où les chemins sont remplacés par des étiquettes. Il est aussi possible d'établir des sessions LDP entre deux routeurs distants, ce qui est peut être nécessaire par exemple dans de la mise en place de certains tunnels.

LDP permet une meilleure extensiblité des tables de routage en interne. En effet, avec LDP, le nombre d'entrées dans le tables de routage est fonction du nombre de points de *sortie* du domaine, et non du nombre de destinations dans Internet.

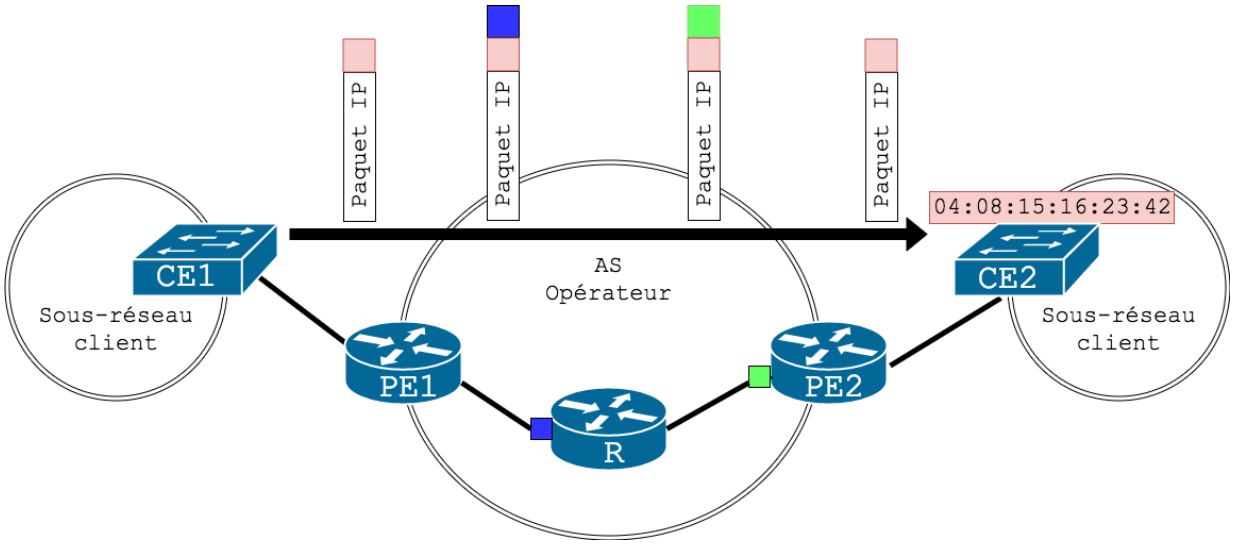

FIGURE 2.1: Schéma des différents acteurs dans le déploiement d'un tunnel. Et d'un tunnel VPWS entre CE1 et CE2. L'encapsulation MPLS se fait avant l'entête IP.

Comme nous le verrons par la suite, RSVP-TE est plus intéressant que LDP si on souhaite mettre en place de la qualité de service car il permet de mettre en place des tunnels qui garantissent une certaine bande passante.

2.2 Du tunnel point à point niveau 2 au VPN niveau 3

Afin d'offrir des services réseaux d'acheminement spécifique, il faut donc d'une manière ou d'une autre mettre en place des tunnels. Ces tunnels servent à deux choses. Premièrement, ils permettent aux opérateurs d'un réseau d'avoir un meilleur contrôle des services qu'ils offrent à leurs utilisateurs en terme de qualité de service. Deuxièmement, ces tunnels permettent le déploiement de VPN. Nous avons vu précédemment les mécanismes qui intervenaient dans la mise en place de ces tunnels, et dans cette section nous allons présenter les différents types de tunnels actuellement utilisés.

Ces tunnels ont comme caractéristique d'offrir aux utilisateurs finaux la possibilité de joindre l'autre bout du tunnel sans avoir besoin d'en connaître l'existence. Pour arriver à cette fin, il existe deux catégories de technologies de *tunneling* : celles offrant un service point à point, c'est à dire un tunnel joignant deux sous-réseaux et celles offrant un service multipoints. Ces deux mode de fonctionnement se font en coopération avec l'opérateur du ou des domaines par lesquels passent ces tunnels. En effet, pour déployer de tels tunnels, il faut intervenir sur l'intégralité du chemin par lequel passe le circuit.

Afin de pouvoir présenter avec précision la topologie d'un tunnel, il est important de bien différencier les différents acteurs qui interviennent dans sa mise en place. Il y a deux catégories d'équipements qui jouent un rôle important : les équipements Customers Edge (CE) et les équipements Providers Edge (PE). Comme nous pouvons le voir figure 2.1, les CE sont les équipements de bordure du sous-réseau d'un client, tandis que les équipements PE sont situés à la bordure du réseau de cœur, par lequel dans notre cas les tunnels transitent.

2.2.1 Le point à point

Un tunnel point à point est un circuit *unidirectionnel*, généralement MPLS, entre un routeur *ingress* et un routeur *egress*. Ils sont déployés dans les réseaux de cœur d'Internet pour deux raison majeures. Premièrement les routeurs acheminant des paquets sur des circuits ne doivent plus faire de routage IP, ce qui accélère le processus décisionnel puisque seul un routeur, le routeur *ingress*, se charge de sélectionner la destination du paquet. Les tunnels point à point entraînent donc un gain en performances mais ils permettent aussi un gain

en *extensibilité*. En effet, comme nous l'avons vu avec LDP, un des protocoles de distribution d'étiquettes automatisée, plusieurs tunnels empruntant un même tronçon utilisent la même étiquette. Par conséquent, les tables de routage MPLS des routeurs intermédiaires sont fonction du nombre de points de sortie du réseau de cœur et non du nombre de destinations IP.

Nous allons maintenant rapidement présenter deux types de tunnels point à point couramment utilisés dans les réseaux de cœur : MPLS Traffic Engineering LSP (MPLS TE LSP) et Virtual Private Wire Service (VPWS).

MPLS TE LSP

L'ingénierie de trafic, en anglais Traffic Engineering (TE), correspond aux mécanismes permettant d'influer sur l'acheminement des paquets dans le but de fournir une certaine qualité de service. MPLS TE LSP fait référence à la mise en place de tunnels grâce à des chemins étiquetés avec MPLS, appelés LSP, à des fins de qualité de service.

Bien qu'il soit plus complexe, RSVP-TE permet à l'émetteur de la demande de tunnel d'ajouter des contraintes à son tunnel. Il peut expliciter le chemin qu'il souhaite que son circuit emprunte, et il peut demander à réserver une partie de la *capacité* du circuit. En contrepartie, le destinataire de la demande de réservation de tunnel, le routeur *ingress* a le droit de refuser la réservation.

Une fois que l'administrateur a spécifié les détails de la réservation, RSVP effectue du routage *Constrained shortest path* (CSPF), grâce à des extensions de protocoles de routages usuels comme OSPF-TE ou IS-IS-TE, pour déterminer quelle route emprunter pour un tunnel donné : avant de calculer le chemin le plus court, il va d'abord éliminer tous les tronçons ne correspondant pas aux critères requis. Grâce à sa traffic engineering database (TED), les routeurs connaissent les bandes passantes disponibles sur chaque lien. Il vont donc éliminer tous les liens ne disposant pas d'une bande passante réservable suffisante.

Les circuits dans RSVP sont simples. Pour déployer le circuit, le routeur d'entrée du domaine envoie un message `RSVP_PATH` au routeur de sortie ciblé. Sur le chemin, chaque routeur stocke des informations concernant le réservation, s'il l'accepte, puis la transmet au prochain routeur du circuit. Le tunnel est ouvert lorsque le routeur reçoit le message `RSVP`, et est maintenu grâce à un échange de message périodique entre les routeurs d'entrée et de sortie.

Dans MPLS TE LSP, l'encapsulation des données se fait avant l'entête IP. Grâce à ce mécanisme, les administrateurs peuvent mettre en place des garanties sur la qualité de service dans leur domaine pour un tunnel donné.

VPWS

VPWS se base aussi sur MPLS pour déployer des tunnels point à point. En revanche, contrairement à MPLS TE LSP, ces tunnels sont déployés au niveau matériel et non plus réseau. Cela signifie qu'un tunnel VPWS offre une connectivité de niveau 2 entre deux points, c'est pourquoi cette technologie est souvent appelée *Ethernet over MPLS*.

Dans la figure 2.1, un VPWS est déployé entre les switchs `CE1` et `CE2` par dessus un réseau de cœur. `CE1` peut ainsi envoyer des paquets destinés à l'interface entrante de `CE2` en se basant sur son adresse MAC. Les routeurs `PE1` et `PE2` encapsulent les paquets dans un tunnel MPLS et, lorsque le paquet atteint `PE2`, le paquet est décapsulé et envoyé à `CE2`. On constate que les deux interlocuteurs ne connaissent pas l'existence du réseau de cœur. Plusieurs méthodes existent pour encapsuler de l'Ethernet dans du MPLS qui sont présentées en détail dans la RFC 4448[29]. Encapsuler de l'Ethernet dans du MPLS revient à mettre en place un "câble Ethernet virtuel", appelé *pseudowire*. Le but d'un *pseudowire* est d'émuler le comportement d'une connectivité Ethernet par dessus un réseau à commutation de paquets, même multisaut. Le résultat est un pont Ethernet logique.

Pour ce faire, le routeur d'entrée du tunnel enlève l'entête IP du paquet reçu et n'en extrait que les informations clé (port source, port destination, EtherType [1]) et s'en sert pour reconstituer un nouvel entête

1. Le champ EtherType est un champ de 8 octets qui, en fonction de sa valeur, indique soit la taille de la charge utile soit le protocole encapsulé

Ethernet destiné à l'émulation. Une fois cet entête produit, le routeur en déduit un identifiant de *pseudowire*, qui sera rajouté par dessus. Enfin ce paquet est transmis à l'entité du routeur responsable de l'aiguillage MPLS, qui applique un, ou plusieurs, *labels* MPLS.

2.2.2 Le multipoint

Les tunnels que nous avons présentés jusque là permettent de mettre en place des tunnels entre deux points de bordure d'un réseau. Les routeurs effectuent ainsi de la commutation de circuits ce qui court-circuite le routage IP. Comme nous l'avions vu en introduction de ce chapitre, cela améliore les performances du réseau en limitant la charge imposée sur les routeurs de cœur du réseau et cela réduit grandement la taille des tables de routage.

Dans de nombreux cas cependant, les opérateurs souhaitent relier entre eux plusieurs routeurs de bordure à l'aide de circuits afin de déployer des réseaux privés virtuels multipoints. Une première solution à ce problème serait de déployer autant de tunnels qu'il y a de combinaisons de paires de routeurs de bordure connecté à une partie de ce sous-réseau. Mais le nombre de combinaison de paires dans un ensemble est fonction quadratique du nombre d'éléments de l'ensemble, ce qui peut sur le long terme poser des problèmes de maintenabilité.

Au lieu de ça, diverses solutions existent pour déployer des tunnels multipoints de niveau 2 et 3, que nous allons maintenant traiter : Virtual Private LAN Service (VPLS), VPN et Border Gateway Protocol (BGP)/MPLS.

VPLS

Les tunnels présentés ici sont établis entre les équipements PE d'un réseau. Les équipements de sortie des utilisateurs du réseau de cœur, une fois le tunnel déployé, peuvent ainsi contacter l'équipement CE d'un autre sous-réseau distant uniquement grâce à leur adresse physique, en faisant abstraction du réseau qui les sépare.

Un VPLS est aussi un mécanisme de *tunneling* de niveau 2, mais, contrairement à VPWS, offrant une connectivité multipoint. Cela permet donc de mettre en place un VPN de niveau 2 déployé sur plusieurs sites séparés par un réseau de cœur. Le mécanisme d'encapsulation des paquets est le même que pour les VPWS. La seule différence entre VPLS et VPWS est que VPLS permet la mise en place de tunnels multipoints. Ces réseaux sont dits privés, car deux équipements CE n'appartenant pas au même VPLS ne peuvent pas se joindre via leurs adresses intraVPLS respectives.

Ces VPN se basent sur une topologie fortement maillée : les routeurs PE peuvent joindre tous les autres routeurs PE du VPN dans le réseau de coeur avec un tunnel VPWS. Ainsi, comme tous les points d'accès sont reliés par un tunnel direct, un routeur d'entrée ne doit jamais *relayer* un message. Il est soit le point d'entrée, soit de sortie du VPN.

Deux versions de VPLS existent. Elles diffèrent dans les technologies employées dans le réseau de cœur pour la mise ne place du plan de contrôle : VPLS peut être basé sur du LDP ou sur du BGP.

Dans le premier cas, les routeurs PE établissent des sessions LDP entre eux sur lesquelles ils échangent les étiquettes d'entrée des différents tunnels et peuplent ainsi une table de routage de niveau 2. Comme pour un commutateur, un routeur PE qui reçoit un paquet dont il ne connaît pas la destination émet ce paquet vers tous les autres commutateurs PE.

L'autre solution emploie BGP. BGP est un protocole originellement destiné à l'acheminement des paquets *entre* différents domaines (plus de détails en annexe B). Afin de garantir la cohérence des plans de contrôle des routeurs d'un même domaine et de pouvoir faire transiter des paquets au sein d'un domaine, tous les routeurs de bordure d'un même domaine communiquent entre eux au sein de sessions iBGP d'un même domaine. C'est au sein de ces sessions iBGP que les routeurs PE peuvent aussi s'échanger des informations pour le plan de contrôle des différents VPLS grâce à une extension du protocle BGP : Network Layer Reachability Information (NLRI). Cette extension avait originellement été conçue pour préciser si les informations échangées concernaient des adresses IPv4 ou IPv6. Depuis, NLRI est souvent employé à d'autres fins, notamment la mise en place de tunnels et de VPN dans un réseau de cœur.

BGP intervient dans trois étapes de la mise en place des plans de contrôle des routeurs de bordure impliqués dans le VPLS : la découverte, la mise en place et la destruction des *pseudowires*. Les routeurs BGP s'échangent des messages BGP `UPDATE` avec un champ NLRI spécifique contenant toutes les informations nécessaires à la mise en place du VPLS.

Les auteurs de la RFC présentant VPWS et VPLS précisent que les performances offertes par un réseau local virtuel de niveau 2 ne peuvent être équivalentes à celles offertes par un réseau local physique.

En 2015, des ingénieurs de Alcatel-Lucent et Juniper ont conçu E-VPN [2]. E-VPN est voué à supplanter VPLS. E-VPN est utilisé en combinaison avec BGP est permet l'usage d'un *route-reflector* BGP, ce qui rend la solution plus extensible. De plus, E-VPN se sert aussi de MP-BGP pour l'apprentissage des adresse MAC, ce qui rend cet apprentissage finement paramétrable.

Le point-à-multipoint Ethernet

Au niveau 2 du modèle OSI, de nombreuses extensions d'Ethernet ont été conçues afin de diviser un seul sous-réseau en plusieurs sous-réseaux étanches. C'est ce qu'on appelle un Virtual Local Area Network (VLAN). Comme nous allons le voir, la mise en place de VLANs offre de nombreux avantages.

Les VLANs : des tunnels *à l'intérieur* des réseaux locaux

Des VLANs peuvent être mis en place sur plusieurs couches du modèle OSI. Il existe des VLANs pyhisiques, matériels et IP. Peu importe la couche sur lequel un VLAN est déployé, des paquets marqués d'un identifiant de VLAN A ne peuvent pas circuler dans un VLAN B.

Étanchéifier ainsi son réseau offre de nombreux avantages. D'un point de vue sécuritaire, cela permet de limiter les communications entre deux sous-parties du réseau, et donc de diminuer le nombre de points d'accès à un réseau, ce qui rend la supervision des accès plus simple. Mais les VLANs permettent aussi de limiter la charge imposée sur un sous-réseau puisque les communications `broadcast` ne se propageront plus qu'au membres d'un même VLAN et non plus du même sous-réseau physique. Les VLANs offrent donc une *segmentation* du réseau, source de sécurité, et une diminution de charge.

En revanche, un VLAN n'a qu'une portée locale, ce qui est contraignant. En effet il est impossible pour un administrateur d'un réseau d'entreprise, par exemple, de regrouper des entités fonctionnellement similaires bien que séparées géographiquement dans un même VLAN.

C'est pourquoi le groupe IEEE 802.1, groupe travaillant sur les protocoles destinés aux réseaux locaux, a conçu différentes normes permettant de propager des VLAN sur des VPNs. Ces normes étant intensivement utilisées dans GÉANT, il est important de les mentionner. Elles sont présentées plus en détail en annexe C.

VPN BGP MPLS

Pour l'instant nous avons étudié la mise en place de tunnels multipoints de niveau 2. Avec les VPN BGP MPLS, il est possible de déployer un réseau IP virtuellement connexe, mais physiquement divisé en plusieurs sites séparés par un réseau de cœur, de niveau 3. Ainsi, deux utilisateurs appartenant à deux sites distants mais appartenant au même VPN peuvent appartenir au même réseau IP.

Principe

Le but des VPN BGP MPLS, est de laisser la responsabilité du déploiement et du maintien des tunnels à l'opérateur du réseau de cœur. Ainsi, l'utilisateur du VPN est responsable du routage IP au sein de son VPN, tandis que l'opérateur du réseau de cœur doit mettre des tunnels en place et acheminer les paquets à bon port. Cette répartition des rôles est communément appelée *peer-model*, puisque c'est le routeur CE de l'utilisateur qui travaille de paire avec le routeur PE du réseau de cœur auquel il est connecté. Les routeurs CE de l'utilisateur ne communiquent plus directement entre eux.

Au lieu de cela :

2. https://tools.ietf.org/rfc/rfc7432.txt

NOM	DATE	Type	Protocoles	Objectif
VLAN	1998	Sous-réseau	VLAN - 802.1Q	Découpage LAN
VPN BGP MPLS	1999	Overlay IP	BGP, MPLS, LDP	Réseau privé IP distribué
MPLS TE LSP	2001	Tunnel IP	MPLS, RSVP TE	Qualité de service
VPWS	2005	Tunnel ethernet	VPWS	Tunnel couche 2
Q-in-Q	2005	Overlay VLAN	802. 1ad	Transport VLAN
VPLS	2007	Overlay ethernet	VPLS	Overlay niveau 2
Mac-inMac	2008	Overlay VLAN	802.1ah	Transport VLAN ciblé
E-VPN	2015	Overlay VLAN	BGP, EVPN	Overlay niveau 2

TABLE 2.1: Types de tunnels et d'*overlays* réseaux couramment utilisés

— le routeur CE du client transmet à son routeur PE toutes les informations de routage IP à l'intérieur de son VPN ;
— le routeur PE traite ces informations et les exporte aux autres routeurs PE concernés ;
— les routeurs PE transmettent les informations à leurs routeurs CE respectifs ;
— chaque routeur CE intègre l'information dans son plan de contrôle pour le routage IP en interne et la propage aux autres membres de sa sous-partie du VPN.

On voit ainsi qu'il y a une démarcation claire des rôles du client et de l'opérateur dans la mise en place et la gestion du VPN BGP MPLS. Cette solution est très extensible, puisque pour ajouter un site au VPN il suffit que le routeur CE du nouveau site transmette ses informations de routage à son routeur PE, et non à tous les autres membres du VPN.

Les routeurs PE s'échangent les informations sur le sous-préfixe auquel ils sont respectivement connectés via des sessions iBGP. Cela pose certains problèmes que les VPN BGP MPLS doivent pallier.

Problématiques

Premièrement, les membres de deux VPN différents ne doivent pas s'échanger d'informations sur les adresses dont ils disposent. Pour éviter cela, les routeurs PE utilisent des tables Virtual Routing and Forwarding (VRF). Une table VRF est associée à un seul et unique VPN, en revanche un routeur PE dispose d'autant de tables VRF que de VPN auxquels il est connecté. Lorsqu'un routeur BGP utilisant le VRF reçoit une route vers un VPN, pour identifier à quelle table VRF cette route appartient, il se base sur la *communauté BGP* à laquelle appartient cette route.

Deuxièmement, deux VPN peuvent utiliser le même préfixe en interne, c'est ce qu'on appelle un *chevauchement* de préfixes. C'est particulièrement intéressant lorsque l'on souhaite utiliser des plages d'adresses IP privées. Or les routeurs PE doivent être en mesure de différencier les deux routes et de les associer au bon VPN. Pour ce faire, les routeurs PE utilisent des adresses VPN-IP. Ce sont des adresses composées du préfixe en question et d'un marqueur de route (*route distinguisher*). Afin d'éviter les collisions, il est déterminé à partir du numéro d'Autonomous Systems (AS) *et* d'un identifiant de VPN choisi par l'opérateur.

Enfin, un troisième problème majeur subsiste : pour l'instant, les routeurs de cœur de l'AS doivent acheminer les paquets en fonction de leur adresse VPN-IP. Cela veut dire que la taille de leurs tables de routages sont fonction du nombre de sites par VPN et du nombre de VPN. Pour cloisonner le routage intraVPN et le routage des tunnels, les VPN BGP MPLS utilsent MPLS.

Ainsi, chaque routeur PE établit un LSP vers tous les autres routeurs PE du domaine. Comme nous l'avons vu avec LDP, grâce à la fusion d'étiquettes, la taille des tables de routage MPLS est indépendante du nombre de destinations. Grâce à BGP, les routeurs PE s'échangent des routes VPN-IP associées à une étiquette de VPN (*VPN label*), qui est une étiquette MPLS destinée à identifier un VPN. Lorsqu'un routeur PE reçoit une route associée à un VPN *label*, il rajoute cette information à sa table VRF.

Parcours de bout en bout : exécution

Lorsqu'il doit envoyer un paquet vers un membre distant du VPN, un routeur PE insère dans l'entête du paquet au moins deux étiquettes MPLS. Tout d'abord l'étiquette de LSP, qui permet aux routeurs de cœur

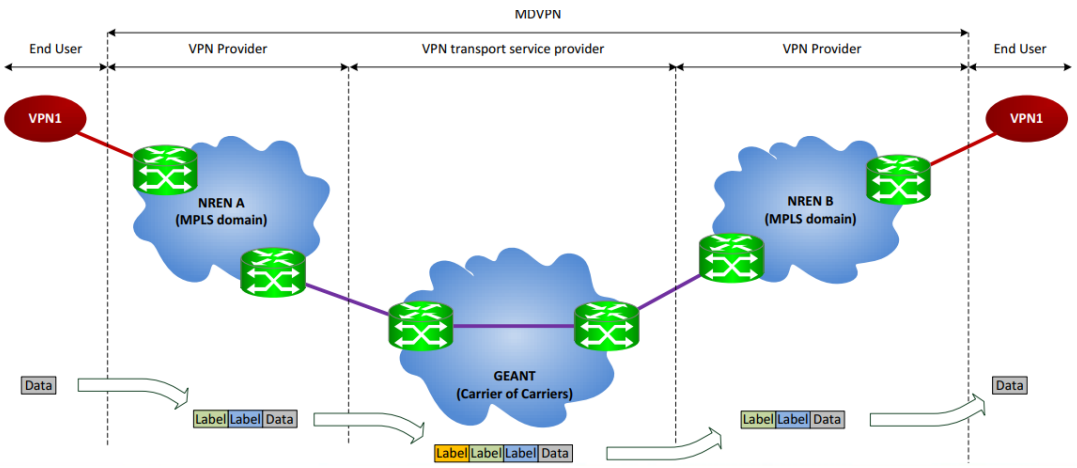

FIGURE 2.2: Suivi des étiquettes d'un paquet au sein de GÉANT MD-VPN

de savoir sur quel circuit MPLS le paquet doit être acheminé et vers quel routeur PE de sortie. La deuxième étiquette est le VPN *label*. Lorsque le paquet arrive au routeur PE de destination, le routeur enlève l'étiquette MPLS qui lui était destinée, et, grâce au VPN *label*, il sait à quel VPN ce paquet est destiné localement. On rappelle en effet qu'un routeur PE peut-être relié à plusieurs VPN.

2.3 GÉANT MD-VPN : transport de tunnels multidomaines

Les tunnels que nous avons décrits précédemment avaient différents degrés de complexité. Dans le cas le plus simple, les VLAN, il s'agit simplement de découper un réseau local en différents sous-réseaux locaux étanches. Mais il est possible de déployer des solutions plus complexes, comme les VPN BGP MPLS qui permettent de déployer un VPN facilement extensible par delà l'AS du fournisseur d'accès à Internet. Il existe une telle diversité de tunnels et *overlays* pour deux raisons. Premièrement pour des raisons historiques : les premiers VLAN ont été définis en 1998 tandis que Ethernet VPN (EVPN) a été défini dans la RFC 7432, qui date de 2015. Tout cela est résumé table **??**.

Ces tunnels n'offrent pas la possibilité d'être étendus à travers plusieurs domaines successifs. Certes les solutions les plus avancées permettent de traverser l'AS du fournisseur, mais ces solutions n'intègrent pas de méthode facilement paramétrable pour déployer un tunnel qui traverserait plusieurs AS afin de relier deux réseaux distants. En pratique, le déploiement de tunnels multidomaines avec les méthodes classiques est laborieux.

Un service novateur

Les tunnels multidomaines sont encore peu usités, puisqu'il n'est pas dans les habitudes de deux opérateurs concurrents de collaborer. Or GÉANT et ses NREN sont des réseaux à vocation purement éducationnelle et académique. C'est pourquoi il est plus aisé pour les administrateurs de cet ensemble de domaines de collaborer.

Dans ce contexte, ils ont déployé GÉANT MD-VPN, qui est un service destiné à déployer facilement des tunnels multidomaines de tous types à travers une succession d'AS. Dans cette partie, nous allons étudier en détail son fonctionnement et ses caractéristiques, car GÉANT MD-VPN une des cibles principales du dispositif imaginé au cours de mon stage.

L'objectif de GÉANT MD-VPN est de pouvoir fournir à un utilisateur un moyen simple de déployer un tunnel ou un VPN à travers plusieurs domaines.

Architecture

Dans l'architecture logique de GÉANT MD-VPN, figure 2.2, deux points de démarcation logiques sont à noter :

— Les *Service Demarcation Point* (SDP) ;
— Les *Service Stitching Point* (SSP).

Il ne s'agit bien sûr pas d'entité physiques. Ces points marquent simplement deux frontières. Le SDP démarque le commencement d'un VPN. C'est à partir de là que commence l'encapsulation VPN classique qui peut se faire selon n'importe quel protocole de *tunneling*. On le rappelle, GÉANT MD-VPN est compatible avec n'importe quel protocole de tunnel ou de VPN de niveau 2 ou 3.

La deuxième frontière, le SSP, détermine la frontière à partir de laquelle on réencapsule les paquets. Cette fois-ci, on le fait indépendamment du protocole VPN utilisé. C'est une méta-encapsulation qui permet de livrer les paquets au domaine visé peu importe le protocole initialement utilisé. C'est pourquoi GÉANT MD-VPN est aussi appelé "transporteur de VPN".

Au sein d'un NREN, le déploiement d'un tunnel s'effectue de manière classique. C'est entre les SDP qu'il s'effectue différemment.

Les routeurs de bordure des NREN, les ASBR, sont chacun identifiés par une étiquette MPLS. C'est pourquoi les ASBR de GÉANT sont tous reliés par des sessions iBGP, et qu'ils transmettent aux ASBR des NREN leur étiquette MPLS grâce à BGP LU.

En parallèle, les routeurs ASBR s'échangent des étiquettes MPLS pour chaque VPN qu'ils acheminent. Ils peuvent le faire grâce à différents protocoles, qu'il s'agisse de BGP ou de LDP.

Pour résumer, un routeur de bordure d'AS est identifié par une étiquette MPLS `A`. Un VPN est identifié par une étiquette de VPN `V`. Ainsi, le site d'un VPN est identifié par `<A;V>`, et c'est cette paire d'étiquettes qui est apposée sur les paquets transitant dans le réseau de GÉANT.

Déploiement d'un tunnel

Pour déployer un tunnel, un utilisateur autorisé sur un campus transmet une demande contenant la description complète du tunnel aux administrateurs des SDP concernés. Par exemple si un administrateur souhaitait monter un service entre iCube et un laboratoire de l'Université de Normandie, seuls les réseaux régionaux OSIRIS et SYVIK auraient besoin de se mettre d'accord, RENATER et GÉANT ne seraient pas informés. Pour l'échange de routes VPN de nouvelles sessions BGP multihop sont montées entre les route-reflectors MD-VPN de chaque entité (GÉANT, NREN, réseaux régionaux, etc).

Une fois que le NREN initiateur du tunnel a reçu le feu vert des SDP concernés, il effectue des tests et, si ils sont concluants, dispose du tunnel commandé. La figure 2.2 illustre l'apposition des étiquettes sur un paquet empruntant un tunnel multidomaine.

CHAPITRE 3

Contribution : PathCap

Sommaire

Mon stage consiste en la mise en place d'une solution de vérification des performances des tunnels *multidomaines* déployés au travers de GÉANT et de ses NREN. C'est pourquoi nous avons dans un premier temps présenté les problématiques liées à la vérification des performances des réseaux et les dispositifs de supervision déjà existants sur Internet. Puis, dans un second temps, nous avons présenté les structures de tunnels et de VPN existants et utilisés. Il était en effet nécessaire de comprendre le fonctionnement des tunnels et *overlays* car ils seront la cible du dispositif de supervision présenté dans ce stage.

Dans ce chapitre, nous présenterons l'objectif du stage, et les problématiques posées par la supervision des tunnels multidomaines à haut débit. Puis, nous allons présenter en détail la solution PathCap et ma contribution.

3.1 Difficulté de superviser les tunnels multidomaines et solution

Le but de l'équipe dans laquelle nous sommes impliqués est de vérifier que les tunnels MD-VPN ne fassent pas l'objet de défaillances. Étant donné que ses tunnels sont déployés avec des contraintes de type SLA, un tunnel sera bien sûr considéré comme défaillant s'il est en panne, mais aussi si ses performances ne respectent pas les SLA préétablis (en termes de taux de pertes, variations de délais, etc). Ainsi un tunnel ayant des problèmes de performances sera considéré comme un tunnel défaillant.

Dans le contexte des tunnels multidomaines, notre plateforme a plusieurs objectifs qui sont :

— la localisation des défaillances et des problèmes de performance en général dans un tunnel [1] ;
— la collecte de statistiques d'utilisation des tunnels en quasi temps-réel ;
— la mise en place d'un tableau de bord de l'état des tunnels parcourant GÉANT et ses NREN.

Les indicateurs qui témoigneraient le mieux de l'état d'un tunnel sont des mesures de pertes, gigue, délai et dans une moindre mesure de bande passante. Bien que ces objectifs semblent raisonnables à première vue, nous allons voir qu'ils posent de nombreux défis.

3.1.1 Verrous scientifiques et techniques

Des tunnels ne dépendant pas que des performances des réseaux qu'il traversent

Il n'y a pas de dépendance directe entre l'état d'un réseau et l'état d'un tunnel le traversant. Autrement dit, la panne d'un lien emprunté par un tunnel n'est pas une condition suffisante à la panne du tunnel l'empruntant. Les protocoles de VPN et tunnels disposent en effet de mécanismes permettant de s'adapter à une défaillance du réseau qu'ils traversent. De même, la panne d'un tunnel n'est pas nécessairement due

1. Pour des raisons de coûts, nous nous contenterons d'effectuer la localisation de défaillances à l'échelle d'un AS.

à la panne d'un lien sous-jacent. Les tunnels sont régis par des entités logicielles au niveau des routeurs, ce qui peut être une autre source de la défaillance.

Cette indépendance relative nous force à effectuer des mesures **à l'intérieur** des tunnels. On ne peut se baser sur des mesures de l'état des liens sous-jacents. Or, il y a deux manières d'effectuer des mesures à l'intérieur d'un tunnel.

On peut tout d'abord détecter le chemin qu'emprunte un tunnel et effectuer des mesures sur chaque tronçon de ce chemin, en y injectant des paquets forgés à la volée - paquets marqués comme appartenant au tunnel. Cette méthode est complexe, car les tunnels ont des topologie dynamiques, ce qui demande de recalculer les chemins qu'emprunte chaque tunnel pour savoir quels liens sonder. Il faudrait donc avoir accès aux configuration de chaque routeur traversé potentiellement par le tunnel, et avoir accès à chaque modification de la configuration afin de connaître la topologie du tunnel en temps réel. Une autre solution consisterait à écouter tous les messages de contrôle circulant sur un réseau et reconstruire le plan de contrôle de chaque routeur à partir de ces messages pour en déduire le chemin qu'emprunte un tunnel. Solution peu fiable, car elle repose sur la garantie que tous les messages de contrôle ont été reçus depuis le début de la mise en service du réseau. Cette solution n'est pas envisageable pour notre plateforme, car il faudrait reconstituer le plan de contrôle de chaque routeur potentiellement traversé par au moins un tunnel dans GÉANT et ses NREN. De plus, ces deux solutions peuvent être très coûteuses en ressources.

Il est aussi nécessaire de rappeler que les tunnels supervisés sont multidomaines. Cela rend la tâche encore plus ardue puisque non seulement les tunnels peuvent changer de route dans un domaine, mais il peuvent même emprunter des domaines différents en cas de panne. Les tunnels MD-VPN, par exemple (*cf.* 2.3) peuvent traverser une suite d'AS complètement différente de la route précédente en cas de changement topologique. Cela accroît considérablement les ressources nécessaires au calcul des routes empruntées par un tunnel, et le nombre de routeurs qu'il faudrait surveiller, puisque le nombre d'AS à surveiller augmenterait par la même.

De plus, notamment pour des raisons de sécurité, certains tunnels sont trop étanches pour pouvoir injecter des paquets à l'intérieur du tunnel sur n'importe quel tronçon. C'est surtout vrai pour les tunnels de niveau 2, et cela rend encore plus difficile la supervision d'un tunnel à partir de mesures effectuées sur chaque tronçon de celui-ci.

On pourrait aussi tout simplement se contenter d'effectuer des mesures de bout en bout. Cela nécessiterait d'avoir un dispositif à l'entrée du tunnel et un à la sortie du tunnel, qui effectueraient des mesures actives de pertes, gigue et de délai avec des outils comme OWAMP et PathLoad que nous avons présentés en 1.2. Une telle solution est réalisable, puisque les dispositifs pourraient être des solutions matérielles ou logicielles légères et simples à mettre en place. Tout administrateur souhaitant profiter de sont MD-VPN devrait en contrepartie placer chez lui un dispositif de mesures actives à l'entrée de son tunnel.

Malheureusement, cette solution ne permet pas de répondre à l'un des objectifs de notre plateforme : la localisation de défaillances à l'origine de violations des SLA. Il est même possible que les mesures actives ne soient pas perturbé alors que le reste du trafic circulant dans le tunnel le soit.

De plus, un des objectifs de notre plateforme est de pouvoir constituer un historique du trafic qui a réellement traversé les différents tunnels. Ce qui est impossible avec les mesures actives.

La dépendance fort relative entre un tunnel et le réseau qu'il traverse est l'obstacle majeur à surmonter dans la conception d'une plateforme permettant de localiser les pannes sur un tunnel multidomaine. Néanmoins notre solution utilise une technique de mesures innovante afin de répondre à cette problématique, en se basant sur la capture de paquets. C'est ce que nous verrons en 3.1.2.

L'impossibilité des mesures actives temps-réel sans impact

La plateforme de supervision doit offrir aux administrateurs de GÉANT et des NREN la possibilité de consulter les mesures de performance des différentes tunnels multidomaines transitant dans leurs réseaux respectifs *en quasi temps réel*.

Là encore, le choix de la méthode d'acquisition des mesures est déterminant. En effet, inférer les performances d'un tunnel à partir de mesures actives est problématique.

Premièrement, parce que pour effectuer des mesures actives, il faut injecter des paquets dans ce tunnel, ce qui a forcément un impact sur les performances du tunnel. En d'autres mots, l'acquisition de données sur un tunnel par sondage actif nécessite d'influer sur la métrique mesurée. Même si l'impact peut-être minime en fonction de la mesure effectuée, la mesure ne pourra pas refléter la stricte réalité. Certaines mesures comme les mesures de bande passante présentées en 1.2.2 vont jusqu'à déclencher une congestion sur un chemin afin d'en déduire la bande passante maximale de bout en bout.

Deuxièmement, des mesures actives, même dans l'hypothèse qu'elles eussent un impact minime sur le trafic d'un réseau, ne peuvent pas refléter les performances offertes par un tunnel de manière assez précise pour nos objectifs. En effet, le trafic généré par les utilisateurs d'un tunnel influe directement sur les performances du tunnel, et il est nécessaire de reproduire au mieux ce trafic lors d'une mesure active des performances d'un tunnel. Certains phénomènes sont provoqués par des interactions entre différents flux au niveau de la couche transport. Prenons l'exemple de la synchronisation globale TCP. Il arrive que plusieurs flux TCP indépendants se comportent exactement de la même manière en même temps [36].

1. Ils détectent une congestion au même moment.
2. Ils réduisent leur fenêtre d'émission en même temps ;
3. Ils accroissent la taille de leur fenêtre à la même vitesse ;
4. Ce qui reprovoquera une congestion qui sera détectée par les mêmes flux plus ou moins en même temps.

Ce phénomène est provoqué par une synchronisation de mécanismes de contrôle de congestion au niveau 4, dans un contexte particulier. Il est difficile de reproduire ce phénomène à l'aide de mesures actives et il est impossible de simuler toutes les mauvaises interactions possibles sur toutes les couches du modèle OSI. En somme, on ne peut pas refléter exactement le trafic réel des utilisateurs à l'aide de mesures actives.

Il est extrêmement difficile de reproduire le comportement des utilisateurs d'un réseau pour l'étudier à l'aide de mesures actives, et encore moins sans avoir d'influence sur le réseau mesuré. De manière générale, on ne peut pas non plus garantir que l'on pourra obtenir une vision réelle du réseau observé, puisqu'il est tout à fait possible que, les paquets générés pas des mesures actives ne soient pas traités de la même manière que le trafic réel transitant dans le réseau.

Par conséquent, le meilleur moyen d'évaluer les performances d'un réseau et d'obtenir les vraies statistiques sur son utilisation, tout particulièrement en présence de tunnels, est de prélever des données sur ce réseau sans générer de trafic. Il existe des protocoles dédiés à cela comme SNMP ou NetFlow [2]. Mais ces dispositifs imposeraient une charge supplémentaire sur tous les équipements potentiellement traversés par au moins un tunnel multidomaine sur GÉANT et les NREN. Du fait que l'on souhaite obtenir des informations sur tous les paquets traversant le routeur, au moins pour savoir si oui ou non ils appartiennent à un tunnel, il faudrait que tous les équipements de GÉANT et ses NREN analysent tous les flux les traversant. Cette solution serait extrêmement contraignante pour l'ensemble des réseaux supervisés, ce qui risque d'ailleurs de décourager les administrateurs des différentes domaines de participer. Nous souhaiterions donc éviter d'avoir à manipuler directement les routeurs.

Pour avoir des statistiques réelles sur les performances des tunnels multidomaines traversant GÉANT et ses NREN, nous allons effectuer des mesures **passives**, car ce sont les seules qui reflètent le comportement et la dynamique du trafic *réel* traversant le tunnel, et qui n'imposent pas une charge supplémentaire aux routeurs de GÉANT et ses NREN.

La capture de trafic très haut débit

Au vu des contraintes imposées par la localisation de défaillances et la nécessité d'effectuer des mesures passives, nous avons établi que la solution devra :

— capturer le trafic des tunnels traversant les liens de GÉANT et des NREN en certains points ;
— réduire au maximum le volume des données générées ;

2. http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html

— rassembler toutes les données collectées et, en les agrégeant, en déduire les performances offertes par les tunnels traversant les liens ;
— puis, pour finir, sur base de ces statistiques, s'assurer que l'ensemble des SLA soient respectés, sinon contacter les opérateurs du ou des domaines défaillants.

La capture du trafic traversant GÉANT est une première, car ces liens peuvent atteindre une capacité de **100 Gb/s**. Comme nous le verrons en 3.1.2, de telles performances ne peuvent être atteintes qu'en utilisant du matériel dédié et des composants logiciels optimisés à toutes les couches du système d'exploitation effectuant les captures.

De plus, la solution devra réduire au maximum le volume de données générées par les captures, en ne retenant que l'information absolument indispensable aux mesures de performances des tunnels.

3.1.2 PathCap : la supervision par capture de paquets en plusieurs points

La supervision des tunnels multidomaines à très haut débit est donc complexe. Premièrement, car on souhaite évaluer les performances offertes par des *tunnels* et non par des réseaux. Deuxièmement, car ces tunnels ont une topologie dynamique qui rend leur localisation difficile, difficulté qui est grandement amplifiée par le caractère multidomaines de ces tunnels. Troisièmement, car les fonctionnalités demandées - la localisation des défaillances et la vérification des performances réelles en temps réel - posent de nombreuses contraintes sur les méthodes de mesure.

La conjonction de ces trois contraintes est un problème ouvert, car aucune plateforme de supervision existante n'est actuellement en mesure d'offrir une vision aussi détaillée et réaliste d'un système aussi vaste. Pour y arriver, nous sommes en train de concevoir la plateforme provisoirement nommée `PathCap`.

Comme nous allons le voir dans la première partie ce chapitre, `PathCap` est un dispositif conçu de manière incrémentale afin de pouvoir répondre au plus grand nombre de besoins que possible. En deuxième partie, nous présenterons le détail de l'architecture de `PathCap` ainsi que la preuve de concept que nous avons déployée et mon implication dans le projet. Enfin, nous conclurons ce chapitre par une discussions sur le limitations de `PathCap`.

Le projet final étant extrêmement ambitieux, il a été décidé qu'il offrirait trois fonctionnalités, appelés **modes** de fonctionnement. À la fin de son déploiement, les trois modes cohabiteront et répondront à différents besoins. Le projet est ainsi doté d'une architecture très modulaire qui pourra être déployé et amélioré incrémentalement tout en offrant rapidement une partie des résultats attendus.

Trois modes de fonctionnement

Les trois modes de fonctionnement dans `PathCap` sont :
— le mode 3 : le mode le plus ambitieux, la capture de trafic réel ;
— le mode 2 : basé sur la capture de trafic artificiel généré par des mesures actives ;
— le mode 1 : n'utilisant que des mesures actives de bout en bout.

Le mode 1 nous sera utile car c'est un mode qui sera rapidement fonctionnel et qui pourra ainsi nous fournir des résultats assez tôt, à confronter aux résultats des deux autres modes.

Mode 3 : trafic réel et localisation des défaillances

Le mode 3 répond en théorie à tous les objectifs du projet. C'est à dire :

1. les statistiques **réelles** d'utilisation et de performances du tunnel, disponibles en quasi temps réel ;
2. la localisation des défaillances à l'échelle d'un domaine ;
3. la vérification des SLA de bout en bout et par domaine.

Le meilleur moyen de répondre en même temps à ces objectifs est de concevoir une plateforme qui capture tout le trafic transitant dans GÉANT et ses NREN, et qui analyse ce trafic afin de compter les paquets de chaque tunnel et d'horodater finement - au moins à la nanoseconde - leur passage au niveau de chaque nœud du réseau.

En effet, si l'on dispose de la liste des entêtes des paquets transitant par chaque point du réseau et de leur date de passage, on sait quel volume de trafic a transité dans chaque tunnel à chaque instant. Pour obtenir un tableau de bord de l'état des tunnels et un système de localisation de défaillances, il faudra centraliser ces données et les agréger, voire les corréler. Nous verrons les détails architecturaux de `PathCap` en 3.1.2.

Mode 2 : prélèvement de paquets forgés

Le mode 2 a pour objectif d'offrir :

1. des statistiques sur les performances du tunnel basées sur du trafic généré par la plateforme ;
2. la localisation des défaillances à l'échelle d'un domaine.

La différence majeure avec le mode 3 est que le mode 2 ne se base pas sur de la capture du trafic réel. Cette différence offre deux avantages. Premièrement, le trafic forgé aura un débit contrôlable, ce qui résout indirectement les problèmes de capture de paquets à très haut débit. Deuxièmement, il n'y a plus de problème de confidentialité, puisqu'on ne prélève que du trafic factice. Cela présente des avantages "politiques", puisqu'il sera probablement plus aisé pour les administrateurs des NREN d'installer des dispositifs de supervision dans leur domaine dans ce cadre.

En contrepartie, on perd la possibilité d'avoir des statistiques parfaitement réalistes sur les performances des tunnels multidomaines de GÉANT et ses NREN. Pour vérifier les métriques offertes par un tunnel, il faudra le tester en le sondant activement. Ce pose le problème de la supervision par le sondage actif présenté en 3.1.1. En revanche il sera toujours possible de localiser les défaillances avec le mode 2 puisqu'on prélèvera le trafic forgé sur tous les points de capture.

Mode 1 : sondage de bout en bout

Bien que limité, ce mode sera la première étape de notre projet et permettra d'obtenir des mesures actives de bout en bout des tunnels. Des dispositifs placés à chaque bout des différents tunnels effectueront des mesures actives de délai, pertes, gigue, déséquencement et bande passante. On aura donc :

1. une vérification partielle des performances du tunnel ;
2. un état du tunnel à chaque sondage.

Si `PathCap` ne dispose que de ces fonctionnalités, alors la plateforme sera limitée. En revanche, les mesures actives de bout en bout pourraient être complémentaires au mesures passives de bout en bout sur du trafic forgé. De plus, étant donné que nous développons notre plateforme incrémentalement, le mode 1 est une première étape satisfaisante, notamment à des fins de comparaison.

Modules de `PathCap`

`PathCap` est une plateforme distribuée basée essentiellement sur trois modules logiques, illustrés sur un exemple figure 3.5.

- le *monitoring correlator* (MCr), en jaune, responsable de la collecte et de la corrélation des données ;
- les *packet capturers* (PC), en vert, qui capturent les paquets et envoient les informations nécessaire au corrélateur ; et
- les *monitoring agents* (MA), en orange, qui seront les dispositifs chargé du sondage actif à l'entrée des tunnels.

Les *packet capturers* : une capture de paquets optimisée

Concrètement, les *packet capturers* utiliseront des TAP réseaux. Un TAP réseau est un boîtier qui dispose d'un port d'entrée et de deux ports de sortie. Son seul but est de dédoubler le trafic sur un port destiné au *monitoring* tout en restant invisible à l'utilisateur. Afin de pouvoir analyser tout le trafic, la capture sera optimisée à chaque étape. Les TAP utilisé seront finement paramétrables, performants, et donc très coûteux.

Le dispositif qui collectera les paquets prélevés par le TAP se basera sur un système de capture de paquets (*packet capture engine*) optimisé, comme par exemple WireCap[40]. Il s'agit d'un pilote logiciel dédié à la

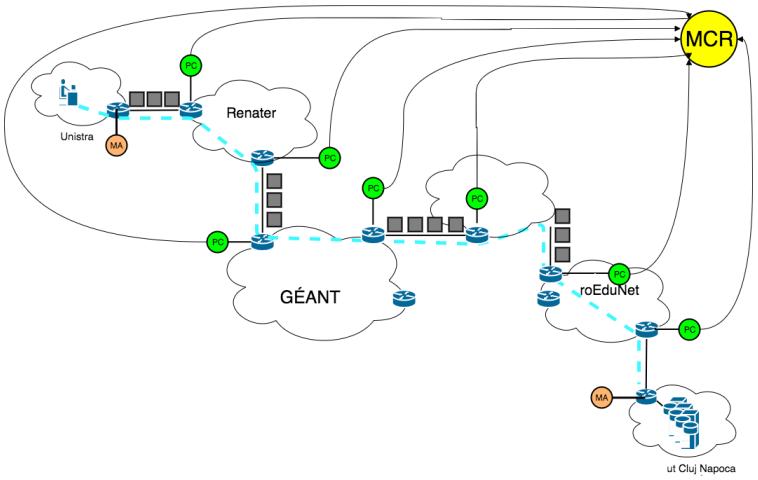

FIGURE 3.1: Schéma de l'architecture de `PathCap` : un tunnel reliant Strasbourg à Cluj Napoca est *monitoré*. Le trafic est capturé par tout les *packets capturers* puis transmis au *monitoring correlator*. Les *monitoring agents*, font du sondage actif.

capture de paquets. Afin d'atteindre des hautes performances, ces pilotes réimplémentent la pile réseau du système. Le but de la nouvelle implémentation est de limiter au maximum les copies de données depuis la carte réseau vers l'espace utilisateur et de paralléliser la remontée des paquets dans la pile réseau.

Les PC devront, le plus possible, réduire le volume des données qui sera remonté vers le corrélateur. Pour ce faire, il y a deux possibilités.

D'un côté, on peut effectuer une réduction *par paquet* seulement. Cela signifie ne conserver que les données pertinentes pour chaque paquet. La date de passage dans le PC, l'identifiant du service emprunté par le paquet, l'adresse IP du PC, les adresses IP source et destination, les ports sources et destination, et éventuellement des données sur la taille du paquet ainsi qu'un *hash* du paquet permettant d'avoir une empreinte de celui-ci, pour l'identifier de manière unique.

Une autre solution consisterait à effectuer une réduction des données *par flux*. En effet les entêtes des mêmes paquets de flux TCP et UDP ont les mêmes adresses IP et port source et destination. Pour réduire au maximum le volume de données transmises au corrélateur, il suffirait de ne lui transmettre que l'entête du premier paquet puis la date de passage et un numéro de séquence pour les paquets suivants. Néanmoins, pour distinguer la retransmission (TCP, par exemple), une empreinte est certainement toujours utile pour le trafic réel.

Le *monitoring correlator (MCr)* : le cœur de la plateforme

Le *monitoring correlator* est une entité logicielle dont le but est de construire une vision cohérente de l'état des tunnels et des flux qui les traversent à partir des données prélevées par les PC.

Les données reçues seront réduites au maximum. Néanmoins il est indispensable que le corrélateur puisse identifier un paquet de manière unique dans les données qu'il a reçues des PC ayant prélevé le paquet.

C'est pourquoi il est très important d'assigner le même identifiant à un paquet sur différents PC, ce qui pourrait se faire à l'aide d'une fonction de hachage du paquet entier. Il suffirait de cet identifiant, de l'identifiant du flux auquel il appartient, et de l'horodatage d'un même paquet à chaque étape de son trajet pour dresser toutes les statistiques désirées initialement.

La localisation des défaillances serait de même relativement simple. En effet on pourrait effectivement déduire où les pertes ont lieu dans un tunnel en cherchant le dernier PC ayant détecté les paquets.

Les *monitoring agents* (MA)

Les MA seront des dispositifs légers placés à l'entrée et à la sortie de chaque tunnel à des fins de sondage actif. Ils présentent l'avantage d'être simple à installer et n'ont pas besoin d'être extrêmement performants.

Bien que peu adapté aux statistiques d'utilisation réelles du tunnel, le sondage actif pourra être utilisé pour les extensions du projet. Un des objectifs sur le long terme serait d'effectuer des tests de bande passante maximale de bout en bout lors du déploiement d'un tunnel afin de vérifier qu'il offre bien les performances promises à l'utilisateur.

De plus, le mode 1 sera le premier mode fonctionnel de `PathCap`. En attendant d'avoir toutes les fonctionnalités attendues, la plateforme offrira rapidement des données de mesures actives périodiques de délai, gigue, pertes et déséquencement, ce qui peut permettre de détecter des défaillances, notamment en cas de panne du tunnel, mais seulement de bout en bout.

Hypothèse

Pour être certain que les performances offertes par un tunnel soient correctement évaluées avec la capture de trafic, `PathCap` s'appuie toutefois sur une hypothèse forte. En effet, soit un tunnel traversant les points de capture `A` et `B`. Si le point `A` capture un paquet `X`, mais que ce même paquet n'est jamais capturé par `B`, alors soit :

— il s'agit d'une perte ;

— la topologie du tunnel a changé, et le paquet `X` a contourné `B`.

Le seul moyen de différencier une perte d'un contournement est de s'assurer que tous les paquets traversant un tunnel passent par deux points situés le plus près possibles des deux *Service Termination Points* (STP) du tunnel, qui font office de point de capture de référence. Ces points de référence doivent obligatoirement être traversés par le tunnel supervisé en toutes circonstances. Cet aspect est encore plus fondamental à l'échelle multidomaine.

3.2 Déploiement de la preuve de concept

Pour l'instant, nous avons vu l'architecture de `PathCap` tel qu'il sera déployé dans GÉANT et les NREN en fin de projet. Afin de vérifier la faisabilité de celui-ci, nous avons dans un premier temps déployé une *preuve de concept* de `PathCap` sur une plateforme dédiée aux essais réseaux : *GÉANT Testbeds Service* (GTS). Mon stage est essentiellement dédié au déploiement de cette preuve de concept, dans lequel j'ai eu un rôle clé : le développement du *monitoring correlator*.

Dans cette section, nous présenterons d'abord rapidement GTS. Puis nous verrons l'architecture de la preuve de concept. Enfin, nous étudierons en détail le *monitoring correlator*. Son code source est libre d'accès (https://github.com/tehlinger/aligregator).

3.2.1 GTS : un *testbed* hybride

GTS est un outil offert par GÉANT qui fournit des environnement virtuels dédiés à la recherche en réseaux informatiques [3]. Pour ce faire, GTS dispose de *points of distribution (POD)* dans ses PoPS. Chaque POD dispose d'un réseau virtuel réalisé grâce à des *switchs* virtuels Openflow[31] et serveurs virtuels. Les POD sont tous reliés entre eux par le réseau de production. Ainsi, leur distance géographique permet d'émuler de vrais réseaux étendus sur un continent.

Dans ce contexte, notre équipe a commandé un banc d'essai permettant :

1. d'effectuer des mesures actives au travers de plusieurs tunnels.
2. de capturer le trafic généré par des mesures en **plusieurs points**.

Le résultat de notre commande est présenté figure 3.2. Notre *testbed* est localisé sur un seul POD, composé de plusieurs équipements de deux switchs virtuels (`OVS` sur la figure) et de trois routeurs classique : deux Juniper aux extrémités et un Cisco au cœur. Les différents points - aussi bien les `MA` que les `CE` - sont des machines virtuelles ubuntu sur lesquels nous avons les droits d'administration.

3. https://www.geant.org/Services/Connectivity_and_network/GTS/Documents/GTS%20v1.1%20Engineering,%20Installation,%20and%20Configuration%20Guide.pdf

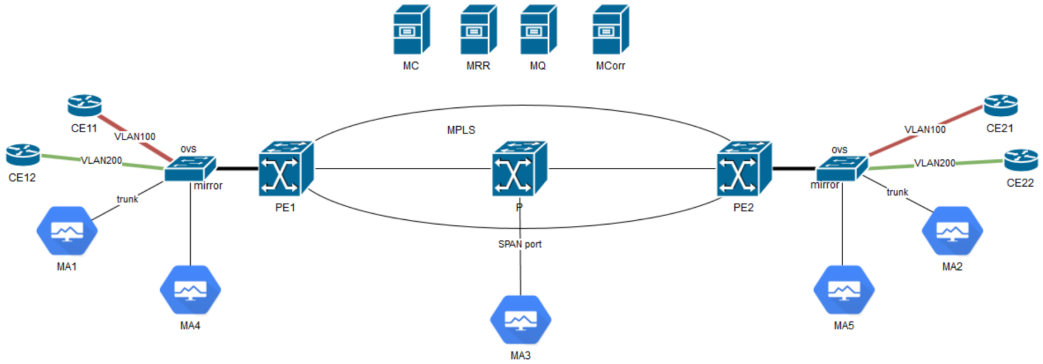

FIGURE 3.2: Notre banc d'essai pour le déploiement d'une preuve de concept : `CE11` et `CE12` appartiennent à deux vlan différents. Ils communiquent, dans leurs vlan respectifs, avec `CE21` et `CE22`. Ces VLAN sont interconnectés avec MPLS. Le trafic est capturé par les différents `MA`

.

Les deux tunnels supervisés ont une architecture hétérogène. Sur les tronçons les plus proches des émetteurs/récepteurs, il s'agit de deux VLAN. Au centre de la maquette, un tunnel MPLS relie les deux VLAN distants. Les étiquettes MPLS changent entre les différents segments. Bien sûr, chaque VLAN est encapsulé avec des étiquettes MPLS différentes.

Grâce à du *port-mirroring*, le trafic généré sur les vlan 100 et 200 est récolté par les différents `MA`. Les quatre serveurs représentés en haut sont :

— Le *monitoring controller* (MC), est un serveur dédié à la configuration des captures dans `PathCap`. Sur le long terme, il permettra de paramétrer la supervision de tunnels de manière centralisée et quasi automatique.
— Le `MRR`, le serveur dédié à la présentation et au stockage des données.
— Le serveur `MQ`, qui sert à l'échange des messages entre les modules.
— Le serveur `MCor`, qui est le serveur agrégeant les données et calculant les statistiques de trafic.

3.2.2 Architecture de la preuve de concept

Pour cette preuve de concept, notre équipe souhaite répondre à une partie des besoins et objectifs du projet. La condition nécessaire est tout d'abord que la solution soit **distribuée**, ce qui nous permettrait de mettre en évidence les difficultés posées par un déploiement à grande échelle. De plus, la preuve de concept doit impérativement :

1. Effectuer des mesures périodiques de pertes, délai et de gigue sur les différents tunnels (Mode 1).
2. Capturer le trafic en différents points des tunnels et établir des statistiques d'utilisation (Mode 2).
3. Offrir une interface permettant de consulter les statistiques en temps réel.

Nous avons donc déployé une plateforme, fonctionnelle depuis début juillet, offrant toutes ces fonctionnalités. Dans cette partie, nous allons détailler l'architecture globale de la preuve de concept. Nous conclurons enfin ce chapitre par la présentation de ma contribution à cette preuve de concept.

Implémentation des différents modules

En pratique, nous avons implémenté les modules de `PathCap` selon le schéma figure 3.3.

Communication intermodule

La preuve de concept est distribuée sur plusieurs terminaux. Ceux-ci vont devoir s'échanger un grand nombre de données :

— Les PC envoient tous les paquets capturés au MCor ;

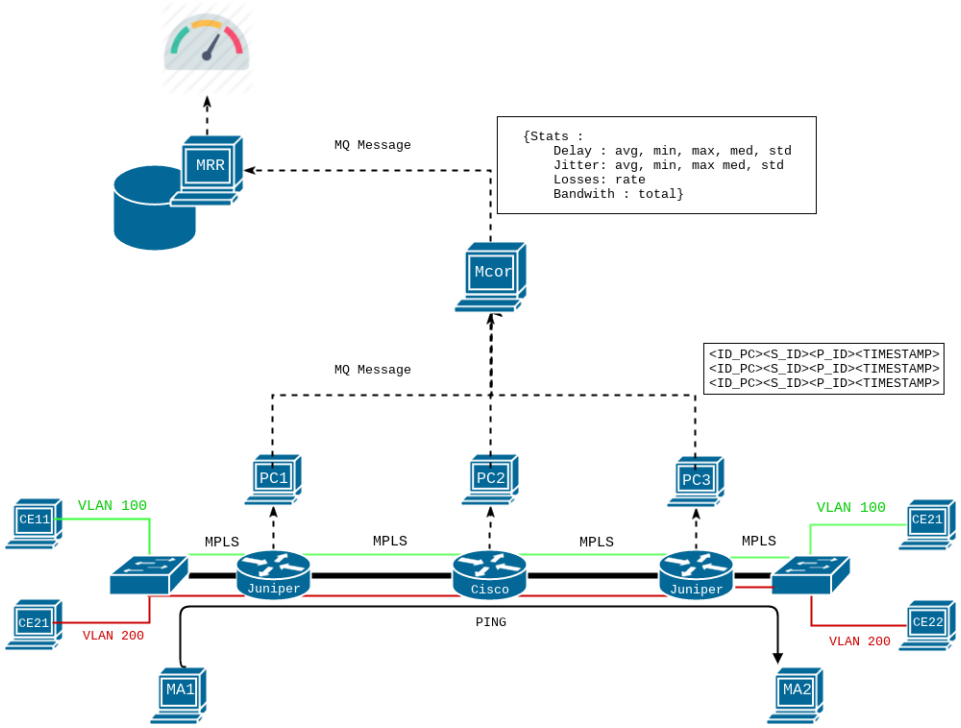

FIGURE 3.3: Schéma de l'implémentation de la preuve de concept

— les MA envoient leurs mesures directement au MRR ;
— le MCor envoie les statistiques calculées au MRR.

En pratique, ces communications sont centralisées dans la preuve de concept, mais rien n'empêche de faire autrement dans la perspective d'un déploiement réel. Elles passent toutes par le serveur `MQ`. Ce serveur utilise `rabbitmq`[4]. RabbitMQ est un gestionnaire de message libre robuste et simple d'utilisation, compatible avec de nombreux langages de programmation.

Le serveur `MQ` instancie des files `rabbitMQ`. Ces files sont remplies par les messages émis et peuvent être consommées de manière asynchrone. La gestion des files et de leur consommation est entièrement configurable, avec par exemple la possibilité d'exiger ou non l'acquittement des messages, de choisir différents algorithmes de répartition de messages entre les consommateurs...

Mode 1 : sondage actif

Le sondage actif est effectué de `MA1` vers `MA2`. Il a été implémenté par Pavle Vuletic qui travaille pour le NREN serbe AMRES. Ces mesures de pertes et délais ont lieu de manière périodique et leurs résultats sont directement émis vers le `MRR`.

Mode 2 : mesures et génération de trafic

David Schmitz, de l'université LMU à Munich, est responsable de la génération de trafic **et** des mesures actives pour le mode 2. `CE12` et `CE11` émettent des mesures unidirectionnelles avec `owampd` vers dans leurs tunnels respectifs dont les interdéparts suivent une loi exponentielle d'espérance 0.1 seconde. Ce trafic est constitué de paquets contenant un **magic cookie**, une valeur statique prédéfinie. Quand les `PC1, PC2` et `PC3` lisent les paquets et détectent le *magic cookie*, ils émettent un message vers l'agrégateur (MCor).

Ce message contient les données suivantes :

```
<id_PC> ; <Service id>l; <ip_dst:ip_src> ; <pck_id> ; <timestamp>
```

4. http://www.rabbitmq.com

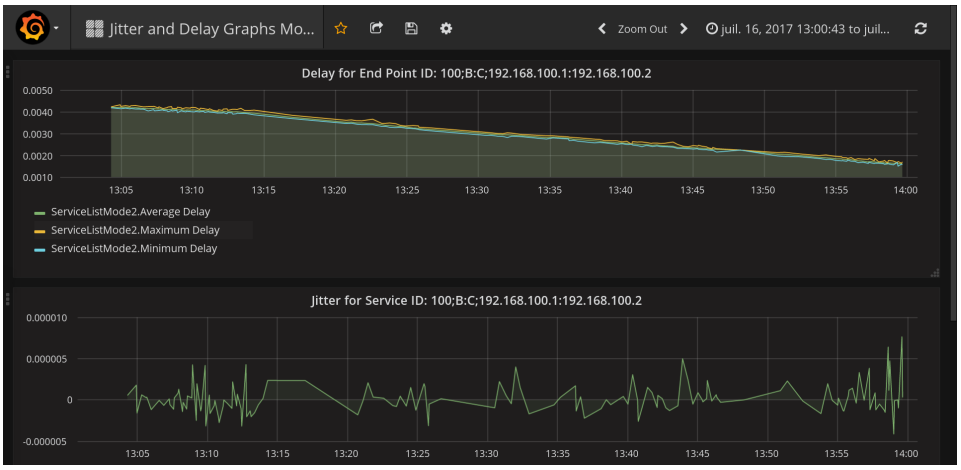

FIGURE 3.4: Tableau de bord Grafana représentant les délais moyens sur un tronçon d'un tunnel.

Le MCor reçoit tous les messages des différents points de capture et, de là, en déduit les différentes métriques de chaque paquet ayant traversé les différents tunnels. C'est cela qui permet d'évaluer les performances des tunnels MPLS.

Stockage et présentation des statistiques

En fin de chaîne de traitement se trouve donc le *monitoring repository* (MRR). Développé par Henrik Wessing, assistant professeur à l'université du Danemark, le MRR est en charge de la réception des statistiques calculées par l'agrégateur et des mesures effectuées par les *monitoring agents* ainsi que de leur présentation.

Pour ce faire, le MRR utilise deux outils. Premièrement, pour le stockage des données, l'application se sert du gestionnaire de base de données libre **influxDB** [5], car celui-ci est spécialisé dans le stockage de séries temporelles. De plus, InfluxDB donne accès aux données nouvellement stockées moins de **100ms** après leur réception. Cela permet de réduire au maximum le délai entre la capture du paquet et la présentation des statistiques sur les mesures.

Pour la présentation, c'est Grafana [6] qui est utilisé. Grafana est une plateforme permettant de déployer des tableaux de bords web pour le *monitoring* de systèmes. La figure 3.4 montre le tableau de bord de notre dispositif.

Nous avons passé en revu presque tous les composants majeur de la preuve de concept.Avec Pascal Mérindol, nous avons développé l'agrégateur de données, qui va donc être présenté en détail dans la section suivante.

3.3 Contribution : agrégateur de captures multipoints temps-réel

L'agrégateur de données doit :

1. réceptionner les données prélevées par tous les *packets capturers* ;
2. en déduire la topologie des routes empruntées par les différents tunnels ;
3. calculer les statistiques sur différentes métriques déduites de l'horodatage des paquets successifs.

Cette application étant le centre de la preuve de concept, une des contraintes du projet est que nous étions extrêmement dépendants de l'avancée du développement des *packet capturers* et du MRR. En effet nous ne maîtrisions pas totalement le format de données attendues en sortie ni le format des données reçues par l'agrégateur en entrée.

5. https://github.com/influxdata/influxdb
6. https://grafana.com

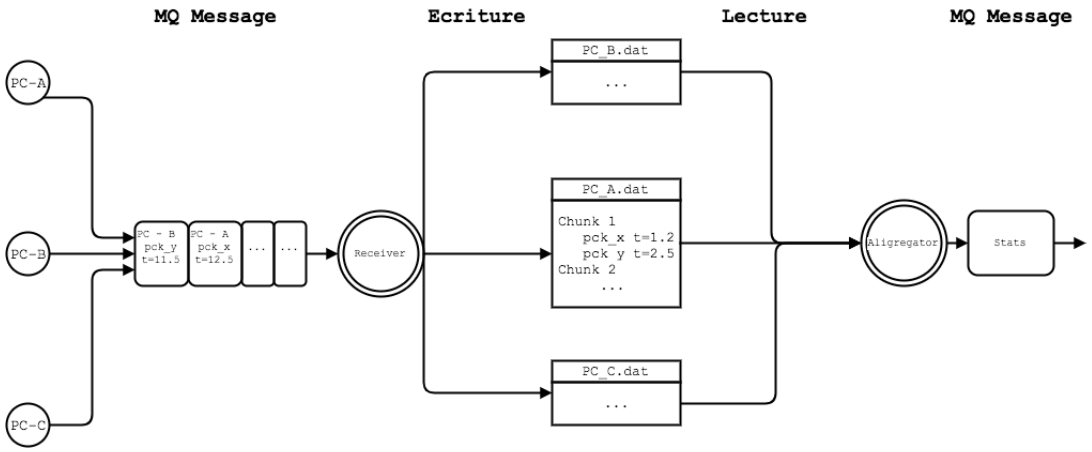

FIGURE 3.5: Architecture d'`aligregator`

Pour se libérer de cette contrainte, nous avons divisé le logiciels en plusieurs modules. Dans cette partie nous allons présenter `aligregator` (`Aligregator` : a listening agregator), le fruit de la collaboration entre Pascal Mérindol et moi-même.

3.3.1 `Aligregator` : *the listening agregator*

Remarque : dans cette section, du ***pseudocode*** *est régulièrement utilisé, avec une syntaxe proche de celle du python. Il n'est naturellement pas exécutable. De plus, des morceaux du code sont passés sous silence car ils présentent un intérêt algorithmique moindre.*

`Aligregator` est composé de deux applications indépendantes : un récepteur de données et un agrégateur.

L'architecture d'`aligregator` est représentée en figure 3.5. À gauche, les quatre *packet capturers* envoient les métadonnées collectées à une file commune. Le récepteur, une application python, est en charge de rassembler ces métadonnées par *chunk*, ou bloc. Un *chunk* rassemble l'ensemble des informations collectées sur n secondes. Cette notion de blocs de paquets est omniprésente dans `aligregator`. Lorsque le récepteur reçoit tous les paquets correspondant à un bloc, cette information est renseignée dans un fichier temporaire correspondant au PC émetteur. L'agrégateur effectuera des calculs (typiquement moyenne, médiane, min, max, variance) par bloc de paquets reçus. Ainsi, on réduit considérablement le volume de données puisque seules les statistiques globales d'un bloc de n paquets sont conservées sur le long terme, et non les *timestamps* de chaque paquet.

L'agrégateur, à droite, est donc en charge du calcul des statistiques à partir des fichier générés par le récepteur, ainsi que de l'émission de ces statistiques. L'algorithme suivant décrit l'ensemble de ses fonctionnalités à haut niveau :

Algorithme 3.1: Boucle principale d'`aligregator`

```
while True:
    i = next_chunk_id_to_load( )
    raw_data = load_chunk(i)
    sort(raw_data)
    stats = get_stats(raw_data)
    send(stats)
```

On y voit que les étapes de l'algorithme sont, successivement :

1. la lecture des fichiers et calcul du prochain bloc à analyser ;
2. la lecture de ces traces et le chargement en mémoire par blocs de paquets ;
3. le tri des *timestamps* afin de reconstituer la topologie des tunnels ;

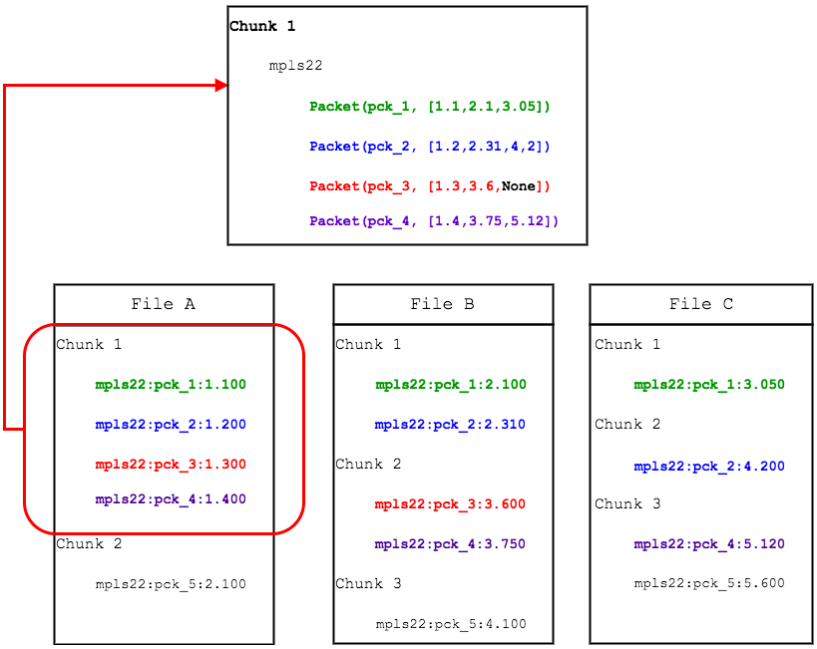

FIGURE 3.6: Exemple du chargement en RAM du 'Chunk' numéro 1. Le paquet 3 a été perdu entre le PC B et le PC C.

4. le calcul de statistiques de trafic ;
5. l'envoi des données produites.

Réception des données et génération de trace

La preuve de concept devait être fonctionnelle fin juin. Malheureusement, le *testbed* n'est devenu opérationnel que fin mai. Afin de commencer l'implémentation plus tôt, nous avons donc décidé que le cœur d'`aligregator`, la partie la plus complexe, se baserait sur la lecture de traces réseaux. Un fichier dont chaque ligne décrirait les métadonnées d'un paquet.

C'est le consommateur qui s'occuperait de la transformation de toutes ces traces selon le format défini pour le mode 2. Ainsi le consommateur pourrait-être codé à partir du moment où le *testbed* serait fonctionnel, tandis que le cœur serait déjà prêt.

Tous les *packet capturers* émettent leurs messages dans une seule file RabbitMQ, une file multiproducteur/simple consommateur. Le consommateur lit les messages, et inscrit les métadonnées dans le fichier correspondant à l'émetteur.

Pour ce faire, le consommateur conserve en RAM la liste des paquets reçus durant n secondes pour chaque émetteur et, au passage à la seconde suivante, inscrit les lignes correspondantes dans le bon fichier, avec une ligne contenant les métadonnées de ce bloc de paquets.

Un extrait de trace est disponible en annexe D. Il faut en retenir que chaque paquet est identifiable par un identifiant unique. Les autres informations figurant dans une ligne sont les identifiants du tunnel auquel il appartient, l'ip source et destination du paquet, et bien sûr le *timestamp* de son passage.

Chargement en RAM des métadonnées de paquets

Comme nous l'avons déjà mentionné, `aligregator` charge les données par blocs, ou 'Chunk' depuis des fichiers remplis par le récepteur. Pour l'instant, un fichier correspond à un *packet capturer*, et tout est défini statiquement. Un bloc correspond à n secondes de paquets, n défini dans le récepteur de paquets. Comme les paquets traversent les différents *capturers* à des instants différents, il est possible qu'un paquet appartienne au bloc $\mathcal{X}$ pour le PC1 mais qu'il appartienne au bloc $\mathcal{X}+1$ voire $\mathcal{X}+2$ ou $\mathcal{X}+3$ pour un autre PC, c'est le cas par exemple du paquet `pck_2` dans la figure 3.6. En effet l'écriture des paquets par bloc n'est pas synchronisable entre tous les PC au niveau du récepteur.

Cette désynchronisation peut donner lieu à des chevauchements au niveau des blocs. Par exemple, toujours dans la figure 3.6, le *timestamp* du paquet `pck_4` est situé dans le bloc 2 du fichier B et dans le bloc 3 du fichier C. Donc, lorsque l'agrégateur charge les *timestamps* du bloc 1 du fichier A, il doit vérifier dans les blocs 2 et 3 des fichiers B et C pour trouver les *timestamps* correspondants. Le nombre de blocs suivants parcourus est paramétrable.

Compréhension de la topologie

Ce point est un des points délicats du code. En effet, le projet a pour objectif de déduire la topologie des tunnels à partir des *timestamps* des paquets, ou des TTL IP. Comme notre *testbed* est composé de deux tunnels à la topologie statique, qui de plus passent par les deux mêmes liens successifs, nous pensions que nous ne pourrions pas travailler cet aspect de l'agrégateur dans cette phase du projet.

Or, lors de nos premiers essais, les délais calculés par l'application étaient négatifs. Cela était du à une mauvaise synchronisation des horloges des différents points de capture et à de très faibles délais, d'autant que les équipement utilisés dans notre banc d'essais sont tous situés au même endroit. Cela rendait à première vue les résultats de nos mesures inexploitables. `Aligregator` ne pourraient que fonctionner avec une synchronisation très forte des horloges de points de capture. En revanche, nous avons pu tirer parti de l'incohérence de ces résultats pour implémenter une nouvelle fonctionnalité dans notre application.

En effet, en admettant une synchronisation temporelle forte entre les points de capture, nous pouvons reconstituer la topologie d'un tunnel à partir du classement chronologique de ses paquets. Comme les horloges du *testbed* étaient mal synchronisées au début de notre projet, nous avons décidé que notre application devait trier les *timestamps* de tous les paquets d'un même bloc par ordre croissant de sorte à ce que les délais mesurés soient positifs. Certes les mesures seraient fausses avec cette méthode, mais nous avons pu grâce à cela implémenter la reconstitution de topologie de tunnels sans connaissance préalable, qui est fonctionnelle sous réserve d'une synchronisation d'horloges forte. Entretemps les horloges ont été synchronisées plus fortement, ce qui nous garantit des mesures d'une plus grande précision.

Une fois que tous les paquets d'un bloc et leurs *timestamps* ont été chargés en RAM, c'est à dire ligne 4 de l'algorithme 3.3.1, `aligregator` trie tous les *timestamps* de chaque paquet et retient les opérations de tri à effectuer. Puis il détecte l'ordre majoritaire dans un bloc, et en déduit qu'il s'agit de la topologie du tunnel comme illustré dans algorithme suivant :

Algorithme 3.2: Détection de la topologie des tunnels à partir de l'ordre chronologique des *timestamps*

```
def sort(raw_data):
    most_common_order = get_most_common_ts_order(raw_data)
    apply_order(most_common_order, raw_data)

#Detection de l'ordre des timestamps le plus frequent
def get_most_common_ts_order(raw_data):
    sort_indexes = dict( )
    for packet in raw_data.packets:
        indexes = get_sort_instructions(packet)
        sort_indexes[str(indexes)] += 1
    return biggest_value(sort_indexes)

def apply_order(most_common_order, raw_data):
    for packet in raw_data.packets:
        packet.timestamps = [packet.timestamps[i] \
                for i in most_common_order]

def get_sort_instructions(packet):
    l = packet.timestamps
    return [l.index(x) for x in sorted(l)]
```

Cette méthode permet d'être insensible aux événements isolés comme les pertes ou les changements de route temporaires. Naturellement ce processus est paramétrable de deux façons.

Premièremenent, il suffit de se baser sur un échantillon des paquets pour calculer la topologie. La taille de cet échantillon et le processus de sélection est paramétrable. Deuxièmement, si au sein d'un bloc de paquets deux topologies successives sont présentes, alors il peut s'agir par exemple d'un changement de route. Il faut fixer des seuils à partir desquels `aligregator` considère qu'une variation d'ordre des *timestamps* n'est pas un événement isolé mais le témoin d'un changement dans le plan de données du réseau.

Le *testbed* ne nous a pas permis de jouer sur les seuils, c'est donc un problème qui reste ouvert pour la prochaine implémentation de PathCap à plus grand échelle.

Calcul des statistiques

À partir des *timestamps* d'un bloc de paquets, on peut calculer des statistiques sur les différents indicateurs de performance du tunnel. Il est surtout possible de calculer ces statistiques pour chaque *tronçon* du tunnel.

Le calcul de délai est trivial : comme on connaît, pour chaque paquet, l'instant exacte auquel ont franchi les différents points du tunnel, à la nanoseconde près, il suffit de faire calculer la différence entre les différents horodatage pour connaître le délai sur les différents tronçons. De même, pour les taux de pertes, il suffit de compter le taux de paquets pour lesquels il manque des *timestamps* sur le nombre total de paquets du bloc.

Le calcul des variations de délai nécessite une étape supplémentaire puisque nous nous basons sur les variations de délai entre deux paquets successifs. Nous ne calculons que les variations de délai pour des paquets qui se suivent et qui n'ont pas été perdus. Soient les paquets p_1, p_2, p_3 et p_4. Si p_2 est perdu, alors la aucune variation de délai ne sera calculée. Cela nous permet de faire des statistiques sur des valeurs comparables.

Enfin `aligregator` offre même la possibilité de faire des mesures de bande-passante. En effet la trace initiale peut contenir une colonne indiquant la taille du paquet. Si c'est le cas, alors `aligregator` calcule aussi le débit généré par un bloc de paquets.

Tous ces calculs sont effectués en une passe par souci de performances comme décrit ci-dessous :

Algorithme 3.3: Calcul des statistiques de performance pour un bloc de données.

```
def get_stats(raw_data):
    losses = previous_delay = None
    d_max = d_min = max_j = min_j = 0
    all_d = all_j = [ ]

    for p in raw_data.packets:
        total += 1
        if has_loss(p):
            losses += 1
            previous_delay = None
        else:
            d = delay(p)
            if previous_d is not None:
                j = d - previous_delay
            previous_delay = d

            #On met a jour toutes les variables
            update_all(d,j)

    #Toutes les stats sont renvoyees
    return all_stats(total,losses,all_d,all_j,j_min,j_max,d_min,d_max)
```

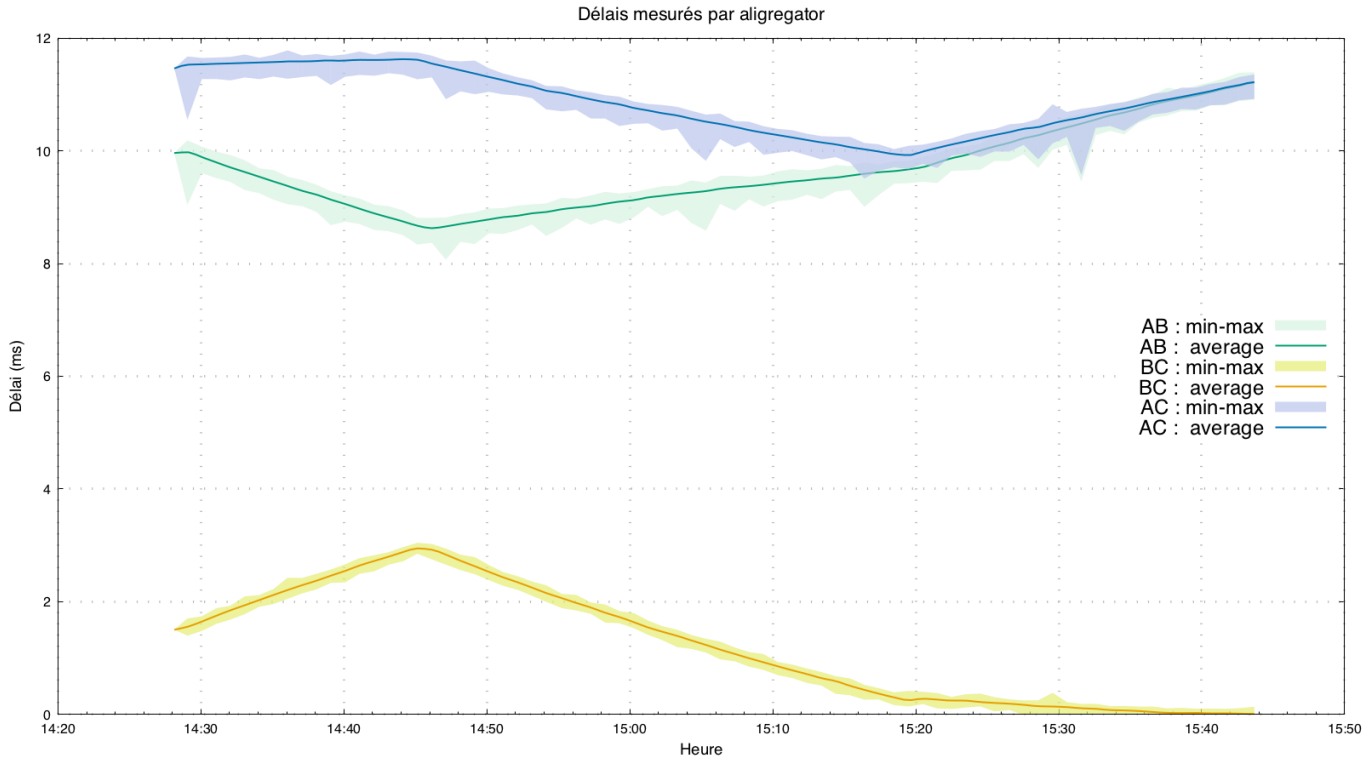

FIGURE 3.7: Moyennes, minimas et maximas de lots de mesures sur une minute de délais effectuées par `aligregator`. En haut, les mesures de bout en bout, en dessous sur chaque tronçon.

Méthodologie de développement

`Aligregator` est une application qui est vouée à être mise en production. C'est pourquoi nous avons fait preuve de rigueur et de méthodologie durant son développement.

Premièrement, le code a été produit en *test driven development*. À chaque étape, les tests unitaire ont été écrits *avant* l'implémentation d'un module. Ainsi, il est possible de vérifier si le module développé est fonctionnel à tout moment. Et même plus tard, si une modification est nécessaire, on peut facilement vérifier si cette modification a une incidence sur le fonctionnement du module. De plus, écrire les tests du code avant son développement oblige à penser la structure du code et son comportement avant sa rédaction. Cela formalise le problème, et oblige à chercher la cas difficiles (*corner-case*) en amont du développement. D'après la littérature scientifique, le *test driven development* est plus efficace que le développement en cascade[20] sur la durée.

3.3.2 Résultats préliminaires

`Aligregator` et désormais fonctionnel et actif dans le *testbed* présenté en 3.2.1.

Le code d'`aligregator` a été rédigé en *test driven development*, ce qui nous offre certaines garanties sur l'exactitude des résultats produits, quand les données en entrée sont correctes. Si les mesures en entrée sont incohérentes, en raison du tri des *timestamps* qui est réalisé pour chaque paquet afin de reconstituer la topologie des tunnels sans hypothèse prélablr, l'application calculera des délais positifs mais non-représentatifs de la réalité. Les résultats produits par `aligregator` ne seront vrai que si les horloges sont synchronisées de manière très forte.

La figure 3.7 représente les délais mesurés par `aligregator` pendant environ une heure sur trois points de capture A, B et C. Les segments sur lesquelles des mesures ont été effectuées sont :

A-C , qui est en fait la mesure de délai de bout en bout ;

A-B , le premier segment de la route empruntée par les tunnels ;

B-C , le deuxième segment.

Sur cette figure, on constate que les résultats sont cohérents *entre eux*. La somme des courbes bleues et rouges est égale à la courbe violette. On remarque même que les points qui se détachent de la courbe bleue se détachent aussi de la courbe violette.

Tout ce que l'on peut en conclure, c'est qu'`aligregator` semble produire de résultats cohérents. Néanmoins il est impossible pour l'instant de savoir s'ils s'approchent de la réalité ou pas.

3.3.3 Discussions et perspectives

À l'heure actuelle, PathCap calcule donc des statistiques sur le trafic transitant dans deux tunnels distincts selon le mode 2, c'est à dire en prélevant du trafic généré par `owampd` qui contient un *magic cookie*. Cette méthode permet de mesurer les délais, taux de perte et la bande passante sur chaque tronçon de ces tunnels au fil du temps et d'en déduire la topologie des tunnels.

Prochains développements

GÉANT *testbed service* va déployer dès novembre un nouveau banc d'essai dédié à la preuve de concept 2.0 de PathCap. Dans celle-ci, l'équipe JRA2-T4 pourra implémenter et tester deux fonctionnalités essentielles de PathCap.

Premièrement, cette nouvelle maquette disposera de plusieurs routes pour relier les deux sites des différents VLAN. L'équipe pourra manipuler le plan de contrôle pendant la collecte de mesures. Le changement de route devra être détecté par `aligregator`. Nous pensons qu'en cas de changement de route pour un tunnel, un ou deux blocs contiendront des paquets ayant traversé différentes routes, et les blocs suivants contiendront uniquement des paquets ayant traversé la nouvelle route. Un des futurs défis de JA2-T4 sera justement d'utiliser des outils statistiques pour pouvoir classifier les blocs de paquets entre ceux contenant un changement de route et ceux contenant des paquets perdus dans une boucle de routage, ou bien temporairement déviés.

Deuxièmement, la preuve de concept 2.0 devra implémenter le Mode 3 de PathCap. En d'autres termes, il faudra cette fois émuler du trafic proche de ce que connait un réseau de cœur comme celui de GÉANT dans ses tunnels. Cela non seulement pour tester les *performances* des différentes composantes de PathCap, mais surtout cela permettrait de comparer les caractéristiques du trafic généré avec les statistiques calculées par PathCap. Il sera ainsi possible de vérifier si la capture de trafic à des fins d'évaluation de performances est une méthode efficace.

Verrous futurs

Comme nous allons le voir, l'architecture repose sur trois hypothèses fortes. Premièrement, il doit toujours être possible d'identifier à quel tunnel un paquet appartient.

Pour identifier le tunnel encapsulant, il faut chercher un identifiant de tunnel ou de VPN dans l'entête en fonction du protocole d'encapsulation. D'ores et déjà, on remarque que les tunnels utilisant du chiffrement d'entête ne pourront pas être évalués car ils sont justement conçus de sorte à ce qu'on ne puisse pas savoir à quel tunnel un paquet appartient.

Pour les entêtes non chiffrés, il faut être sûr qu'on a une **sémantique de bout en bout** pour chaque tunnel. En d'autres termes, il faut que les paquets empruntant les tunnels observés dispose d'un identifiant de tunnel ou de VPN qui est conservé dans l'intégralité du tunnel. Heureusement, nous avons vu que l'étiquetage des paquets dans les tunnels de GÉANT-MDVPN est transparent. Mieux, la pile MPLS des entêtes est composée de trois étiquettes :

— une étiquette pour le transport entre les deux SSP de GÉANT ;
— une étiquette identifiant le tunnel entre les deux NREN ;
— une étiquette identifiant le tunnel de bout en bout.

Pour un paquet donné, il sera donc toujours possible non seulement d'identifier le tunnel auquel il appartient, mais aussi entre quels NREN ce tunnel est déployé.

Deuxièmement, la synchronisation des horloges des points de capture doit être très forte. Non seulement pour avoir des mesures précises, mais aussi pour garantir une reconstitution correcte de la topologie des tunnels.

Dans notre preuve de concept, nous avions synchronisé les horloges avec un serveur NTP qui était situé dans le réseau local de test. Et pourtant nous avons constaté des décalages de l'ordre de la dizaine de miliseconde, ce qui est considérable. Pour pallier ce problème, on peut :

— utiliser un protocole plus efficace de synchronisation d'horloge, comme `RADclock` ou `ntimed` ;
— utiliser de la synchronisation d'horloges par GPS ou ondes radio, qui est plus coûteuse mais plus efficace.

S'il est impossible d'avoir une synchronisation d'horloges assez forte et que le TTL IP des paquets n'est pas disponible, il faudra rajouter des hypothèses fortes. Par exemple, si le premier et le dernier point de capture d'un tunnel sont connus en amont, ce qui est le cas dans GÉANT MD-VPN avec les STP, alors il est possible de comparer les résultats obtenus avec les modes 2 et 1 (mesures actives et passives) pour déterminer si le dispositif est en mesure de reconstituer ou non la topologie des chemins.

Le troisième défi à surmonter lors du déploiement de PathCap à grande échelle sera d'ordre social. Concrètement, il faut tenir compte du respect de la vie privée des utilisateurs de GÉANT et ses NREN. En effet une grande partie du trafic transitant dans les tunnels sera copiée au niveau des TAP, puis traitée par les *packet capturers*. Il est donc indispensable d'*anonymiser* au maximum les données prélevées. Pour cette raison, le contenu des paquets ne sera naturellement jamais conservé. Ensuite, une fois le paquet identifié par son flux et son numéro de séquence dans le flux, voire une empreinte, il n'est pas nécessaire de conserver les adresse et les numéros de port en mémoire. Il suffira de remplacer ses informations par des identifiants, tout en faisant attention aux collisions.

Il faudra convaincre les administrateurs de NREN d'accepter que le trafic de leur réseau soit copié à des fins de supervision. Une solution pour cela est de leur laisser l'initiative de la transmission des données vers `aligregator`. En d'autres termes, les captureurs enverraient les données prélevées à un serveur contrôlé par l'administrateur du domaine auquel il appartient. Libre ensuite à l'administrateur de vérifier que les données sont correctement anonymisées avant de les transmettre au corrélateur.

Cette solution empêcherait en revanche de pouvoir dresser des statistiques en temps réel, c'est pourquoi elle est vue comme une solution de secours en cas de réticence forte.

Conclusion

L'objectif de ce stage est ambitieux, car il pose des défis à de nombreux niveaux.

La **supervision** des réseau à commutation de paquet, premièrement, est un domaine complexe de par l'hétérogénéité des réseaux et leur aspect fortement distribué. De plus le choix des mesures prises et leur traitement statistique est délicat. Mal opérée (mauvais outils, indicateurs ou traitement statistique), cette phase peut donner une vision biaisée des performances du réseau ciblé.

Afin de répondre aux nouveaux besoins des utilisateurs en terme de bande-passante, les opérateurs utilisent différentes technologies afin de déployer des circuits point-à-point et multipoints. Ces tunnels et *overlays* offrent une meilleure étanchéité et, une fois déployés, simplifient l'acheminement des paquets. De plus, il permettent de faire de la réservation de bande passante. Pour autant la supervision de ces **tunnels** ajoute un nouveau niveau de complexité. En effet, évaluer les performances d'un service demande de pouvoir identifier le trafic qui lui appartient et de plus, nous avons vu que les performances d'un tunnel ne sont pas exclusivement dépendantes des performances du réseau qu'il traverse, mais aussi d'autres facteurs comme par exemples les composantes logicielles en charge de la mise en place et de la gestion des services.

Enfin, dans les réseaux de l'éducation et de la recherche, il existe désormais des tunnels **multidomaines**. Leur topologie est dynamique aux échelles intra et interdomaines, et, qui plus est, il peuvent acheminer du trafic à des débits extrêmement élevés.

Le défi est donc le suivant : est-il possible d'évaluer, voire de vérifier, les performances offertes par des tunnels multidomaines en temps-réel ?

Nous pensons que oui, sous certaines conditions bien évidemment.

Dans le cadre d'un projet européen de grande envergure, nous avons déployé une preuve de concept opérationnelle. Notre architecture utilise une nouvelle méthode d'évaluation de performances alliant méthodes actives et passives de supervision : la mesure par la capture multipoint de trafic. Pour connaître les performances réelles de services réseau, nous capturons le trafic qui y circule en plusieurs points et en déterminons les différents indicateurs de performance. Cela a été mis en œuvre dans un banc d'essai, où trois points de capture récoltent les métadonnées du trafic circulant dans les tunnels ciblés. Elles envoient les données récoltées à un agrégateur qui calcule les statistiques sur des indicateurs de performances pertinents : le délai, le taux de perte et les variations de délai. Enfin, cet agrégateur envoie les données à une application en charge de la présentation et du stockage des données.

Notre preuve de concept évalue le trafic des deux tunnels parallèles avec précision, sur plusieurs tronçons. Ainsi, quand une perte a lieu par exemple, nous pouvons déterminer entre quels points de capture elle a eu lieu. En exploitant une incohérence dans les mesures, nous avons même réussi à montrer qu'il est possible de reconstituer la topologie des différents tunnels à partir de simple captures. J'étais en charge de l'implémentation de l'agrégateur des données, le cœur de l'application. Son développement a été un succès.

Les prochaines étapes du projet nécessiteront de tester la reconstitution de la topologie des tunnels à topologie dynamique, et la capture de trafic réel à très haut débit.

Je suis heureux d'avoir pris part à cette belle aventure. En quelques mois, j'ai participé à plusieurs étapes du projets, de la conception de PathCap à la mise en œuvre de la preuve de concept. Je suis certes monté en compétences en prenant en charge le développement intégral d'`aligregator`, mais j'ai surtout été marqué par le cachet scientifique du projet.

La rigueur exigée dans la justification des choix pousse à se remettre régulièrement en question. La science apporte son lot d'incertitudes, et cette incertititude est le carburant qui nous a poussé à partir en quête de preuves supplémentaires.

J'avais choisi un stage à la frontière de l'innovation et de la recherche appliquée car j'étais hésitant quant à la suite de mon parcours. Fort de cette expérience fructueuse, je sais dorénavant que je vais emprunter la voie académique, en débutant une thèse sur un sujet connexe.

ANNEXE A

Netradar : évaluation de performance pour les réseaux *mobiles*

En 2016, l'institut de statistiques Internet StatCounter constatait pour la première fois que le nombre de pages Web téléchargées depuis des terminaux mobiles étaient supérieur au nombre de pages téléchargées depuis des terminaux *fixes*[16]. Il est donc désormais extrêmement important de prendre en compte cet aspect de l'évaluation de performances des réseaux, qui pose de nouvelles contraintes. Quand on sonde un réseau mobile depuis les terminaux des utilisateurs, on ne peut pas installer du *hardware* chez le client, comme cela se fait parfois lorsqu'on sonde des réseaux filaires.

Netradar [38], est une solution dédiée à l'évaluation des performances de réseaux mobiles. Pour surmonter l'obstacle de la mobilité des utilisateurs, cette solution est une solution logicielle compatible avec les principaux OS mobiles actuels. Grâce à sa simplicité d'utilisation et de déploiement, Netradar compte plusieurs *millions* d'utilisateurs et collecte un grand nombre de données d'un point de vue utilisateur.

Netradar fait des mesures passives : des informations sur l'utilisateur sont stockées, après anonymisation, comme le modèle de téléphone et l'OS de l'utilisateur ou sa position. Mais Netradar effectue aussi des mesures actives. Les concepteurs de Netradar ont conçu leur propre *speed test*, destiné à effectuer des mesures de délai *et* de bande passante. Ce *speedtest* est une connexion TCP d'environ 10 secondes vers le serveur dédié le plus proche qui a été conçu pour répondre aux problématiques propres aux réseaux mobiles. Il s'agit d'une mesure de bande passante effective. Les auteurs ont fixé 10 secondes après plusieurs constatation empiriques : le RTT moyen d'une connexion Internet mobile est de 151,82 ms, et il faut environ 3 secondes en moyenne à une connexion TCP pour devenir stable et atteindre sa bande passante moyenne. Restent 7 secondes afin d'avoir une quantité représentatives de données.

Grâce à ces mesures, les auteurs ont mis en évidence des caractéristiques propres aux réseaux Internet pour mobile. Par exemple que la moyenne de bande passante effective atteinte chez les 100 meilleurs candidats par modèle de smartphone allait de 5680,33 Kb/s à 11772 Kb/s. Ou bien que la bande passante effective moyenne fluctuait énormément en fonction de la journée. A un endroit donné, gardé anonyme, à 6h du matin elle atteint un pic d'environ 6000 Kb/s alors qu'en heure de pointe, à 20h, elle descend à environ 3200 Kb/s.

ANNEXE B

Routage intra et interdomaine

Le routage regroupe l'ensemble des mécanismes mis en œuvre pour acheminer les paquets vers leur destination dans le cadre de la commutation de paquets. D'une manière ou d'une autre, les routeurs implémentent deux entités logiques pour choisir où acheminer les paquets entrant. Le plan de *contrôle* et le plan de *données*. Le plan de contrôle regroupe l'ensemble des informations qu'un routeur acquiert sur la topologie du réseau auquel il appartient : quelles destinations sont joignables, les poids des différentes routes... À partir du plan de contrôle, le routeur établit son plan de données. Le plan de données est la base des règles que le routeur suit pour acheminer un paquet donné. Ainsi, lorsqu'un routeur reçoit un paquet, il ne consulte que le plan de données pour déterminer l'interface sur laquelle ce paquet doit être acheminé. Le plan de contrôle, lui, ne sert qu'à établir le plan de données.

Dans la suite de cette sous-section, nous verrons comment le plan de contrôle et le plan de données sont établis dans les deux grandes familles de routage existantes : le routage intradomaine et le routage interdomaine.

Routage intradomaine

Le routage intradomaine désigne les mécanismes visant à acheminer les paquets au sein d'un domaine. Il est géré par des protocoles de routage interne, appelés Interior Gateway Protocol (IGP). Au sein d'un domaine, le seul objectif du routage est d'acheminer les paquets sur les routes les moins coûteuses du domaine. Pour établir l'ensemble des chemins les plus courts et les distribuer à tous les routeurs, deux grandes familles de protocoles existent : les protocoles à *vecteurs de distance* et les protocoles à *état des liens*.

Les protocoles à *vecteurs de distance*, dont le représentant le plus connu est Routing Information Protocol (RIP), reposent sur une répartition de l'information extrêmement distribuée. Une fois le protocole lancé, chaque routeur annonce sa présence à ses voisins, ainsi que la route menant à lui. Lorsqu'un routeur reçoit un message annonçant une route vers un autre routeur, il compare la route nouvellement reçue avec celle dont il disposait déjà dans son plan de contrôle. Si elle est meilleure, alors il utilisera cette route, sinon il conservera la précédente. Ce routage se base sur un algorithme de Bellman-Ford distribué, dont l'objectif est de converger vers un plan de données optimal pour chaque routeur. Malheureusement RIP a de nombreux défauts. En cas de rupture de lien des boucles de routage peuvent apparaître, et RIP n'offre pas la possibilité de mettre en place différentes zones de routage ou une quelconque hiérarchie de zones.

C'est pourquoi dans les réseaux de plus grande ampleur il est recommandé d'utiliser Open Shortest Path First (OSPF) ou IS-IS pour gérer l'acheminement des paquets. Cet algorithme de routage à état des liens se base sur l'algorithme de Djikstra et permet à chaque routeur du réseau d'établir son propre arbre des plus courts chemins vers chaque destination au sein du domaine. Pour ce faire, les routeurs envoient toutes les informations sur leurs adjacences à un routeur maître du domaine qui va les agréger sous la forme d'une base de données d'état des liens et va ensuite le repropager à tous les routeurs du domaine. A partir de cette base chaque routeur calcule son arbre des plus courts chemins.

Ces deux protocoles permettent aux routeurs d'un domaine donnée d'établir leur plan de données et de contrôle pour toutes les autres destinations du domaine. Dans Internet, le routage est effectué d'une autre manière. Nous allons d'abord présenter la topologie d'Internet, puis l'algorithme de routage de facto qui y est utilisé.

Topologie d'Internet

Internet est un réseau divisé en sous-parties : les AS. Un AS correspond au réseau d'une entité administrative, comme une entreprise ou un campus universitaire. Deux AS peuvent être reliés soit par une relation de client-fournisseur, soit par une relation de pair à pair. Internet est ainsi un arbre d'AS dont la racine est un

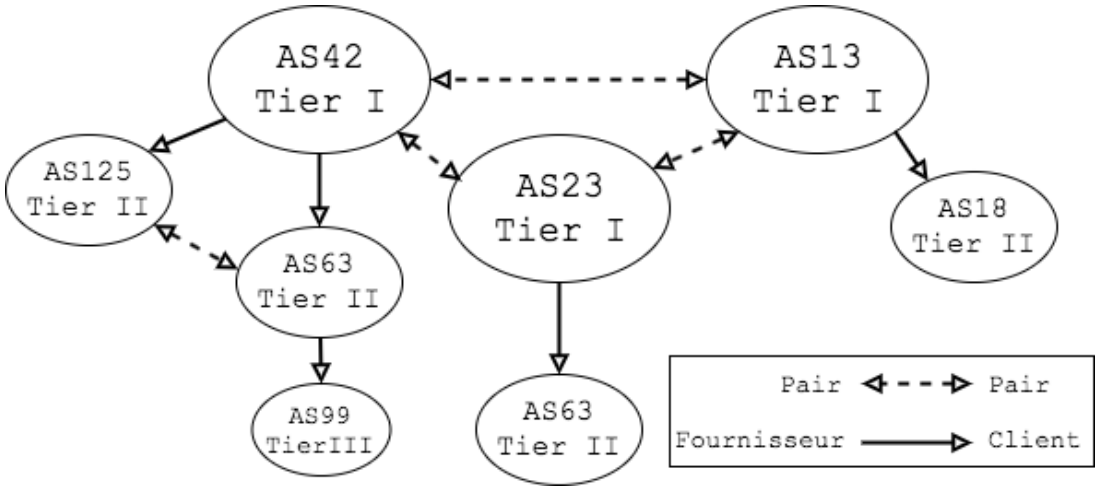

FIGURE B.1: Représentation schématique d'une relation hiérarchique entre des AS d'Internet. Ici, `AS42` est le fournisseur d' `AS125` et il est le pair des deux autres AS *Tier-One* `AS23` et `AS13`.

maillage d'AS *Tier-One* et les feuilles sont des AS de moindre importance. Les fournisseurs facturent à leur client la quantité de données qui est passée par leur propre réseau, tandis que les pairs ne se facturent rien, en se basant sur le principe que chaque pair passe autant par le réseau de l'autre pour accéder à différentes parties d'Internet. La figure B.1 illustre cette hiérarchie. Par exemple, l'AS 99 est le client de l'AS 63 qui lui même est le client de l'glsas *Tier-One* 42. De même l'AS 18 est le client de l'AS 13. Si un utilisateur de l'AS 18 souhaite échanger des données avec un utilisateur de l'AS 125, alors il doit emprunter la route passant respectivement par les AS 13 puis 42. Les AS 125 et 63 sont pairs et ne doivent donc pas passer par l'AS 42 pour accéder aux réseaux de leurs clients mutuellement.

Le routage interdomaines

Le routage *interdomaines* concerne l'acheminement des paquets à travers plusieurs AS d'Internet. Il est régi par les routeurs de bordure des AS. Le routage des paquets n'est pas effectué de la même manière dans le cadre des acheminements interdomaines pour deux raisons. Premièrement, il est très important que les routeurs à l'intérieur d'un domaine ne connaissent pas l'existence des destinations dans les autres domaines. En effet, on compte désormais sur Internet plus de 600 000 AS. Or il est très coûteux de maintenir le plan de contrôle et le plan de donnée d'une table de routage aussi conséquente, c'est pourquoi il est préférable que seul les routeurs de bordure se chargent de cette tâche. L'autre raison pour laquelle le routage interdomaines est géré différemment tient des objectifs du routage interdomaines. En effet, le but du routage intradomaine est d'acheminer les paquets selon les routes de plus faible coût, quelque soit la métrique employée. Or dans l'acheminement des paquets *entre* les domaines, le critère qui prévaut dans les décisions des routeurs est le critère financier. Un routeur doit privilégier la route la moins chère lorsqu'il aiguille un paquet. Comme nous le verrons, cela implique un processus décisionnel plus compliqué que dans le cadre du routage intradomaine ou un seul critère est pris en compte.

Le routage interdomaines s'effectue grâce au protocole BGP. Il repose sur du routage à vecteurs de *chemins*, contrairement par exemple à RIP qui emploie du routage à vecteurs de distances. Concrètement, deux routeurs de bordure appartenant à deux AS voisins vont établir une session BGP afin de s'échanger les routes qu'ils connaissent respectivement. Ainsi dans la figure B.1 par exemple, le routeur de bordure de l'AS 13 va annoncer aux AS 23 et 42 qu'il connaît une route vers l'AS 18 et que cette route passe par lui-même. La différence avec RIP est qu'ensuite, lorsque les AS 23 et 42 vont annoncer à leur clients l'existence d'une route vers l'AS 18, ils annonceront toute la route vers cet AS, c'est à dire une route commençant par leur propre identifiant d'AS - respectivement 23 et 42 - puis passant par l'AS 13 avant d'atteindre l'AS 18.

La différence algorithmique majeure entre BGP et RIP est que dans une session BGP les routeurs ne se contentent pas d'échanger les destinations qu'ils connaissent et le prochain saut vers cette destination mais ils s'échangent les routes dans leur intégralité. En effet, afin de prendre en compte la dimension financière

lors du choix de la meilleure route, les routeurs disposent d'un processus décisionnel plus complexe. Ainsi le processus décisionnel, paramétré par l'administrateur, intervient lors :

— de la réception des routes. Un routeur BGP peut ignorer certaines routes afin de ne pas les prendre en compte la mise en place de son plan de données.

— du partage des routes. Un routeur peut ne pas partager des routes qu'il connaît vers d'autres AS.

— de l'établissement du plan de données à partir du plan de contrôle. Lorsqu'il choisit une route pour une destination donnée parmi les routes dont il dispose dans son plan de contrôle, un routeur se base sur différents critères. Ainsi lorsqu'un routeur choisit une route parmi plusieurs il doit d'abord vérifier si cette route est valide. Puis, il choisit la route que l'administrateur a défini comme la moins coûteuse. En cas d'égalité il départage en se basant dans l'ordre sur : la route la moins longue en nombre d'AS, la route définie comme la moins chère par le voisin pour un même AS, la route vers le routeur de bordure le plus proche en interne. Si plusieurs routes sont toujours à égalité sur tous ces critères, le routeur BGP emprunte la route sortant pas le routeur à l'identifiant le plus fiable, critère arbitrairement défini.

Un routeur BGP sélectionne donc en priorité la route définie comme la moins chère par l'opérateur et seulement en cas d'égalité, il implémente la route la moins longue dans son plan de données.

Nous avons vu comment un routeur BGP choisit quelle route implémenter dans son plan de données et comment il partage les routes qu'il connaît avec les routeurs de bordure des AS voisins. Mais les routeurs BGP doivent aussi communiquer avec les autres routeurs de bordure de leur propre AS afin que chaque routeur sache par quel routeur de bordure un paquet doit sortir pour une destination extradomaine donnée. C'est pourquoi au sein d'un AS les routeurs établissent des sessions iBGP avec tous les autres routeurs de bordure de l'AS. Les sessions iBGP sont des sessions *logiques* : elles peuvent emprunter un chemin traversant plusieurs routeurs internes. En revanche il est indispensable qu'au sein d'un AS tous les routeurs de bordures soient reliés entre eux par une session iBGP. Comme cela impose un grand nombre de connexions, il est aussi possible de connecter tous les routeurs de bordure à un seul et même routeur jouant le rôle de *route reflector*.

La distinction entre sessions iBGP et les sessions entre deux routeurs BGP appartenant à deux AS différents, appelée sessions eBGP, permet d'éviter que les routes soient repropagées indéfiniment au sein d'un AS. En effet, si un routeur apprend une route sur une session iBGP, il ne doit pas la réannoncer aux autre routeurs de bordure de son AS. Le *route reflector* permet de simplifier le processus d'apprentissage de routes au sein d'un domaine : lorsqu'un routeur de bordure apprend une route d'un routeur de bordure appartenant à son voisin, il ne transmet cette route qu'au *route reflector* du réseau qui lui même sera le seul à transmettre cette route aux autres routeurs de bordure du réseau.

ANNEXE C

Ethernet over IP

801.1ad : Q-in-Q

Q-in-Q est une extension de 802.1Q - la norme définissant les VLANs de niveau 2 - qui permet d'ajouter *plusieurs* VLAN-tags sur un paquet.

Son fonctionnement est très simple. Normalement, un identifiant de paquet est une suite de 4 octets inséré dans l'entête Ethernet, entre l'adresse MAC source et le champ EtherType. Deux octets sont destinés à l'identifiant de VLAN, et deux octets sont destinés à l'identifiant de protocole. Dans le cas de 802.1Q il prend la valeur `0x8100`.

Avec Q-in-Q, on applique deux VLAN-tags, c'est ce qu'on appelle en anglais du *double-tagging*. L'identifiant de protocole du premier tag n'est plus `0x8100` mais devient `0x88a8`, le premier VLAN-tag est destiné à l'opérateur du réseau de cœur, le S-TAG, qui est ensuite suivi d'un VLAN-tag normal, celui du client (le C-TAG).

Cette extension permet de propager un VLAN sur plusieurs réseaux locaux séparé par un réseau de cœur : le même VLAN-tag est utilisé par les deux réseaux locaux, et lorsqu'un paquet doit traverser le réseau de cœur, un glsvlan-tag spécifiquement destiné à ce VLAN est appliqué au paquet grâce au double-étiquetage offert par Q-in-Q.

801.1ah : Mac-in-Mac

Paru en 2008, soit 3 ans après Q-in-Q, Mac-in-Mac intervient aussi au niveau 2 dans l'encapsulation des paquets, mais il est plus ambitieux que Q-in-Q. En effet Mac-in-Mac a pour objectif de séparer l'entête Ethernet du paquet en deux parties distinctes lors de son encapsulation, même au niveau des adresses MAC.

Les paquets encapsulés sont sont donc divisés en 3 parties, dans l'ordre :

1. L'entête Ethernet *backbone*, qui contient les adresses MAC source et destination dans le réseau de cœur, en plus du VLAN-tag de fournisseur, qui a exactement la même structure que dans Q-in-Q.
2. Un champ destiné à renseigner quelques options, comme la priorité du paquet ou son *drop eligible indicator*, un booléen indicant si ce paquet peut être détruit ou non.
3. Le paquet Ethernet original.

ANNEXE D

Trace réseau traitées par `aligregator`

Ces traces se divisent en deux catégories de lignes :
— les entêtes de bloc ;
— les lignes de paquet.
Les entêtes de bloc, débutant par une tabulation ont la forme :

```
c_id | t | t + 1
```

Avec `c_id` l'identifiant de bloc et `t` la date de début.
Les lignes de paquets, elles, ont la forme :

```
s_id | f_id | p_id | ts | s
```

Avec :
— `s_id` l'identifiant du **tunnel** auquel le paquet appartient ;
— `f_id` un identifiant de flux de la forme `<ip_dest>:<ip_src>` ;
— `p_id` l'identifiant unique du paquet ;
— `ts` l'horodatage ;
— `s` la taille du paquet.

Exemple :

```
        2867|1499430912.134294|1499430915.188101
200|192.168.200.1:192.168.200.2|14777863512929815907|1499430912.143270731|27136
200|192.168.200.1:192.168.200.2|14777863512079272146|1499430913.540640354|27136
100|192.168.100.1:192.168.100.2|14777863513377757839|1499430913.541876793|27136
200|192.168.200.1:192.168.200.2|14777863512567974900|1499430913.669591188|27136
100|192.168.100.1:192.168.100.2|14777863513177822136|1499430913.698984861|27136
        2868|1499430915.188101|1499430918.273898
100|192.168.100.1:192.168.100.2|14777863513560706372|1499430915.196488619|27136
200|192.168.200.1:192.168.200.2|14777863514874996111|1499430915.229593515|27136
200|192.168.200.1:192.168.200.2|14777863510907069421|1499430915.582141161|27136
100|192.168.100.1:192.168.100.2|14777863511659134675|1499430915.732902527|27136
100|192.168.100.1:192.168.100.2|14777863514232114137|1499430915.754553556|27136
200|192.168.200.1:192.168.200.2|14777863514666646301|1499430915.764070511|27136
```

Bibliographie

[1] Bing 1.1.3. http://fgouget.free.fr/bing/bing_src-readme.shtml. Accessed : 2017-08-02.

[2] Global internet phenomena report. https://www.sandvine.com/trends/global-internet-phenomena. Accessed : 2017-05-02.

[3] Global ripe atlas network coverage. https://atlas.ripe.net/results/maps/network-coverage. Accessed : 2017-05-02.

[4] Iperf. https://iperf.fr/. Accessed : 2017-05-02.

[5] perfsonar bundle selection guide. http://www.perfsonar.net/media/cms_page_media/1227/perfSONAR%20Bundle%20Flow%20Chart.v3.pdf. Accessed : 2017-02-02.

[6] Perfsonar lookup service directory. http://stats.es.net/ServicesDirectory/. Accessed : 2017-02-02.

[7] Guy Almes, Matthew Zekauskas, Sunil Kalidindi, and Al Morton. A one-way delay metric for ip performance metrics (ippm). 2016.

[8] Brice Augustin. Paris traceroute. https://paris-traceroute.net/d. Accessed : 2017-02-02.

[9] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding traceroute anomalies with paris traceroute. In Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, pages 153–158. ACM, 2006.

[10] Brice Augustin, Timur Friedman, and Renata Teixeira. Measuring multipath routing in the internet. IEEE/ACM Transactions on Networking, 19(3) :830–840, 2011.

[11] Paul Baran et al. On distributed communications. Volumes I-XI, RAND Corporation Research Documents, August, pages 637–648, 1964.

[12] Zachary S Bischof, John S Otto, and Fabián E Bustamante. Up, down and around the stack : Isp characterization from network intensive applications. ACM SIGCOMM Computer Communication Review, 42(4) :515–520, 2012.

[13] Steven Blake, David Black, Mark Carlson, Elwyn Davies, Zheng Wang, and Walter Weiss. An architecture for differentiated services. Technical report, 1998.

[14] Barry Constantine, G Forget, R Geib, and R Schrage. Framework for tcp throughput testing. Technical report, 2011.

[15] Sam Crawford. The global platform for internet measurement. https://www.samknows.com/. Accessed : 2017-02-02.

[16] Aodhan Cullen. Mobile and tablet internet usage exceeds desktop for first time worldwide. http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/. Accessed : 2017-10-02.

[17] Carlo Demichelis and Philip Chimento. Ip packet delay variation metric for ip performance metrics (ippm). 2002.

[18] Marcel Dischinger, Andreas Haeberlen, Krishna P Gummadi, and Stefan Saroiu. Characterizing residential broadband networks. In Internet Measurement Comference, pages 43–56, 2007.

[19] Ayush Dusia and Adarshpal S Sethi. Recent advances in fault localization in computer networks. IEEE Communications Surveys & Tutorials, 18(4) :3030–3051, 2016.

[20] Boby George and Laurie Williams. A structured experiment of test-driven development. Information and software Technology, 46(5) :337–342, 2004.

[21] Emanuele Goldoni and Marco Schivi. End-to-end available bandwidth estimation tools, an experimental comparison. In International Workshop on Traffic Monitoring and Analysis, pages 171–182. Springer, 2010.

[22] Sarthak Grover, Mi Seon Park, Srikanth Sundaresan, Sam Burnett, Hyojoon Kim, Bharath Ravi, and Nick Feamster. Peeking behind the nat : an empirical study of home networks. In Proceedings of the 2013 conference on Internet measurement conference, pages 377–390. ACM, 2013.

[23] Ningning Hu and Peter Steenkiste. Evaluation and characterization of available bandwidth probing techniques. IEEE journal on Selected Areas in Communications, 21(6) :879–894, 2003.

[24] Manish Jain and Constantinos Dovrolis. Pathload : A measurement tool for end-to-end available bandwidth. In In Proceedings of Passive and Active Measurements (PAM) Workshop. Citeseer, 2002.

[25] S Shunmuga Krishnan and Ramesh K Sitaraman. Video stream quality impacts viewer behavior : inferring causality using quasi-experimental designs. IEEE/ACM Transactions on Networking, 21(6) :2001–2014, 2013.

[26] Sihyung Lee, Kyriaki Levanti, and Hyong S Kim. Network monitoring : Present and future. Computer Networks, 65 :84–98, 2014.

[27] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, and Christophe Diot. Characterization of failures in an ip backbone. In INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, volume 4, pages 2307–2317. IEEE, 2004.

[28] Jim Martin and Arne Nilsson. On service level agreements for ip networks. In INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 2, pages 855–863. IEEE, 2002.

[29] Luca Martini, E Rosen, N El-Aawar, and G Heron. Encapsulation methods for transport of ethernet over mpls networks. Technical report, 2006.

[30] Warren Matthews and Les Cottrell. The pinger project : active internet performance monitoring for the henp community. IEEE Communications Magazine, 38(5) :130–136, 2000.

[31] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow : enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2) :69–74, 2008.

[32] David L Mills. On the accuracy and stablility of clocks synchronized by the network time protocol in the internet system. ACM SIGCOMM Computer Communication Review, 20(1) :65–75, 1989.

[33] Venkat Mohan, YR Janardhan Reddy, and K Kalpana. Active and passive network measurements : a survey. International Journal of Computer Science and Information Technologies, 2(4) :1372–1385, 2011.

[34] Cristel Pelsser, Luca Cittadini, Stefano Vissicchio, and Randy Bush. From paris to tokyo : On the suitability of ping to measure latency. In Proceedings of the 2013 conference on Internet measurement conference, pages 427–432. ACM, 2013.

[35] Ravi Prasad, Constantinos Dovrolis, Margaret Murray, and KC Claffy. Bandwidth estimation : metrics, measurement techniques, and tools. IEEE network, 17(6) :27–35, 2003.

[36] Lili Qiu, Yin Zhang, and Srinivasan Keshav. Understanding the performance of many tcp flows. Computer Networks, 37(3) :277–306, 2001.

[37] S Shalunov, B Teitelbaum, A Karp, J Boote, and M Zekauskas. rfc 4656 : A one-way active measurement protocol (owamp). Internet Engineering Task Force, 2006.

[38] Sebastian Sonntag, Jukka Manner, and Lennart Schulte. Netradar-measuring the wireless world. In Modeling & Optimization in Mobile, Ad Hoc & Wireless Networks (WiOpt), 2013 11th International Symposium on, pages 29–34. IEEE, 2013.

[39] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A measurement study of available bandwidth estimation tools. In Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, pages 39–44. ACM, 2003.

[40] Wenji Wu and Phil DeMar. Wirecap : a novel packet capture engine for commodity nics in high-speed networks. In Proceedings of the 2014 Conference on Internet Measurement Conference, pages 395–406. ACM, 2014.