Isolation réseau dans un environnement Cloud Public/Hybride

Valentin del Piccolo

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Valentin Del Piccolo

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Isolation réseau dans un environnement Cloud Public/Hybride

soutenue le 23 Mai 2017

devant le jury composé de :

Mme Dominique Gaiti	Rapporteur	Professeur, Université de Technologie de Troyes
Mme Cristel Pelsser	Rapporteur	Professeur, Université de Strasbourg
M. Christian Jacquenet	Examinateur	Directeur du programme IPv6 d'Orange, Orange
M. Stefano Secci	Examinateur	Maître de Conférences, Université Pierre et Marie Curie
M. Djamal Zeghlache	Examinateur	Professeur, Telecom Sud Paris
M. Guy Pujolle	Directeur	Professeur, Université Pierre et Marie Curie
M. Kamel Haddadou	Encadrant	Directeur Recherche et Développement, Gandi SAS

Résumé (Abstract)

Titre :

Isolation réseau dans un environnement Cloud Public/Hybride

Résumé :

Le cloud computing est un modèle informatique permettant l'accès à un grand nombre de ressources tant au niveau du calcul que du stockage. Plusieurs types de cloud existent, le cloud public, le cloud privé et le cloud hybride. Afin de proposer une solution cloud hybride, nous avons débuté avec le protocole TRILL qui permet d'optimiser l'utilisation des ressources réseau d'un data center au sein d'un cloud. Cependant, le protocole TRILL ne permet pas d'interconnecter des data centers sans perdre l'indépendance de leur plan de contrôle. Afin de modifier ce comportement, lequel implique la création d'un unique domaine de broadcast s'étendant sur tous les data centers et sur le réseau d'interconnexion, nous proposons, dans notre première contribution, car le comportement ne convient pas dans un environnement cloud hybride, une solution appelée MLTP qui permet d'interconnecter des réseaux TRILL tout en les maintenant indépendants. Un autre élément qui manque au protocole TRILL est l'isolation des flux des utilisateurs. Notre seconde contribution consiste donc à trouver une solution d'isolation du trafic de chaque client et de l'implémenter au sein de MLTP. Ce nouveau protocole, appelé MLTP+VNT, permet d'avoir une solution de cloud hybride néanmoins elle possède deux désavantages. Le premier est la gestion des pannes. En effet, certains éléments de MLTP+VNT, les Border RBridges, contiennent des informations nécessaires au routage inter-data center et lorsque ceux-ci tombent en panne, les informations qu'ils contenaient sont perdues. Afin d'éviter cela, nous avons, au sein de notre troisième contribution, proposé une modification de MLTP+VNT qui réalise la synchronisation des Borders RBridges. Le second est l'obligation de n'utiliser que des réseaux MLTP+VNT pour réaliser un cloud hybride. Afin de lever cette restriction, nous avons, au sein de notre quatrième contribution, conçu une passerelle entre un réseau TRILL et un réseau OpenFlow. De cette manière, notre solution permet de réaliser un cloud hybride avec la solution MLTP+VNT au sein du cloud public et la solution OpenFlow pour le cloud privé.

Mots clés :

Informatique, Réseau, data center, Informatique en nuage, TRILL, VNT, MLTP, MLTP+VNT, OpenFlow, cloud public, cloud privé, cloud hybride, Isolation réseau.

Title :

Network isolation in a Public/Hybrid cloud environment

Abstract :

Cloud computing uses infrastructure with a lot of computing and storage resources. There are three types of cloud : Public cloud, Private cloud, and Hybrid cloud. In order to provide a hybrid cloud solution, we used as a base the TRILL protocol which optimizes the use of the data center infrastructure. However, TRILL cannot interconnect data centers as doing so will merge the data centers networks and each data center will lose its independence. Our first contribution is to change this behavior and we develop MLTP which allows to interconnect TRILL or MLTP network without merging them. Another functionality missing from TRILL is network isolation. To fill this lack, in our second proposal we add to MLTP a solution called VNT and we then have a new protocol called MLTP+VNT. In this protocol, each user traffic is isolated from one another. Therefore, MLTP+VNT allows to have a hybrid cloud environment. Nevertheless, it has two shortcomings. The first one is its "single" point of failure. As a matter of fact, MLTP+VNT uses a new type of nodes called Border RBridges which contains inter-data centers routing information. If a Border RBridge fails, then the information it contained is lost. In order to prevent this loss, we implement a method to synchronize the Border RBridges in our third contribution. The second shortcoming is the obligation to use MLTP+VNT in each network to form the hybrid cloud. To lift this limitation, we design and develop, in our fourth contribution, a bridge between a MLTP+VNT network and an OpenFlow network. This way, our solution allows to create a hybrid cloud environment with the MLTP+VNT solution in the public cloud and OpenFlow in the public cloud.

Key words :

Computer science, Networking, data center, cloud computing, TRILL, VNT, MLTP, MLTP+VNT, OpenFlow, Public cloud, Private cloud, Hybrid cloud, Network isolation

Cette thèse a été réalisée au sein du laboratoire LIP6 (Laboratoire d'Informatique de Paris 6) situé au 4 place Jussieu, 75005, Paris et au sein du département R&D de la société GANDI SAS située au 63-65 Boulevard Massena, 75013, Paris.

Remerciements

Je tiens à exprimer ma gratitude envers le professeur Guy Pujolle qui m'a permis de réaliser cette thèse CIFRE au sein du Laboratoire d'Informatique de Paris 6 (LIP6) et plus précisément au sein de l'équipe PHARE.
Toujours au sein de l'équipe PHARE, je souhaite remercier Monsieur Stefano Secci pour m'avoir informé de ce sujet de thèse et m'avoir mis en relation avec Monsieur Kamel Haddadou, directeur du département Recherche et Developpement de GANDI SAS, que je tiens à remercier pour l'encadrement qu'il a réalisé lors de cette thèse.
Finalement, je tiens à exprimer ma reconnaissance envers ma famille qui m'a soutenu et aidé tout au long de cette période.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Sommaire

1.1 Motivation

Le Cloud computing est un nouveau modèle économique qui permet, pour ses utilisateurs, de rationaliser et de lisser les coûts des services et départements IT. Au lieu de devoir gérer la maintenance et l'acquisition de nouveaux équipements informatique, l'utilisateur du cloud computing peut, soit souscrire à un abonnement fixe et régulier auprès du fournisseur de cloud, soit ne payer que lorsqu'il utilise des ressources du cloud. Ce modèle économique est en plein essor comme le montre l'étude sur l'adoption du cloud [1], réalisée par les entreprises Everest Group et Cloud Connect auprès de 213 entreprises, qui a révélé que le cloud computing est de plus en plus utilisé par ces entreprises et, pour plus de la moitié de ces entreprises, le cloud computing représente une dépense de plus de 10% du budget IT annuel. De plus, on y voit que le cloud est adopté, certes à des degrés différents, par toutes les industries et que, dans le futur, le cloud hybride est celui qui sera le plus plébiscité. Ce choix du cloud hybride repose sur le fait que le cloud privé est toujours perçu comme plus sûr et sécurisé que le cloud public et, conséquemment les entreprises souhaitent bénéficier des avantages du cloud public tout en conservant un cloud privé pour leurs données sensibles. Toujours en 2014, l'étude réalisée par RightScale [2] montre que, parmi les entités étudiées, 94% expérimentaient avec des solutions cloud et que 58%, parmi ces entités, expérimentaient une solution de cloud hybride. RightScale réalise cette étude annuellement et on peut voir que l'adoption du cloud est en constante augmentation. Cependant, cette adoption varie en fonction du type du cloud. Dans le

rapport de 2015 [3] on peut lire que seule l'adoption du cloud public a augmenté d'un point pour atteindre 30%, mais reste néanmoins inférieure à l'adoption du cloud hybride qui est de 58%. Finalement, le dernier rapport en date [4], celui de 2016, montre une adoption importante du cloud hybride (71%) au détriment du cloud public qui perd 13 points et n'est plus adopté que par 18% des entreprises étudiées.

D'après [5] et [6], en 2014, le marché du cloud hybride s'élevait à 25 milliards de dollars et devrait atteindre 85 milliards de dollars en 2019. Cette perspective d'évolution du marché est partagée par d'autres études qui voient même le cloud hybride atteindre un marché de 91 milliards de dollars en 2021 [7]. Toutes ces études et analyses nous montrent à quel point le marché du cloud hybride est important et a un avenir prometteur avec une croissance importante.

Cependant, avec un tel avenir, les solutions de cloud hybride vont nécessairement avoir à gérer de plus en plus de trafic et d'utilisateurs. En effet, comme le montrent les prévisions de Cisco [8], le trafic global du cloud était de 2,1 Zeta Bytes en 2014 et il est prévu qu'il atteigne 8,6 Zeta Bytes en 2019, ce qui représente un taux de croissance annuel moyen de 33%. Pour parer à cela, il faut donc concevoir des solutions de cloud hybride permettant de s'adapter à de tels débits tout en permettant à chaque utilisateur de bénéficier du cloud sans risque pour ses données. En effet, un des principaux freins à l'adoption du cloud est la sécurité de celui-ci. Dans les rapports [2] [3] [4], on apprend que la sécurité reste d'une année à l'autre l'une des préoccupations principales pour les utilisateurs du cloud. Ils sont encore 28% en 2016 à citer la sécurité comme étant un challenge pour le cloud. C'est pour cela que le cloud privé est toujours aussi plébiscité par les utilisateurs au travers du cloud hybride et que la part du cloud public a autant diminué. Afin de répondre à ce challenge de sécurité pour le cloud public, le cloud privé étant par définition restreint à un seul utilisateur, une solution est de réaliser une isolation de chaque utilisateur ainsi que de son trafic au sein du cloud.

L'isolation des utilisateurs au sein d'un environnement cloud ne peut plus s'effectuer à l'aide des solutions historiques telles que GRE [9] ou les VLANs [10]. En effet, le nombre de VLANs étant limité à 4096, ce nombre n'est pas suffisant pour isoler l'ensemble des utilisateurs d'un cloud. Le problème est donc de trouver, ou de définir, une solution d'isolation capable de passer à l'échelle afin de pouvoir isoler la totalité des utilisateurs d'un cloud.

En plus de cette contrainte de passage à l'échelle au niveau du nombre d'utilisateurs à isoler, la solution doit aussi passer à l'échelle au regard du trafic réseau au sein des data centers. Pour cela, il faut une solution qui permette de maximiser les débits entre chaque nœud de l'infrastructure en réalisant des connexions "any to any" non bloquantes et qui puisse réaliser du multi chemin. Cependant, d'autres contraintes s'imposent comme la rétrocompatibilité de la solution avec les équipements réseau déjà présents au sein du data center. L'utilisation et la configuration de la solution doivent être simplifiées, à l'aide

de mécanismes d'auto configuration, pour l'opérateur de cloud au regard du grand nombre de nœuds au sein des data centers. En effet, l'ajout de nouveaux nœuds doit se faire de façon *plug and play* et lors d'une panne, le nœud fautif doit pouvoir s'auto configurer lorsqu'il redémarre. Finalement, la solution doit pouvoir migrer à chaud (*live migration*) les machines virtuelles (VM) à l'intérieur d'un data center et entre data centers sans interrompre les connexions de cette VM.

Pour répondre à ces diverses contraintes, plusieurs solutions existent. Certaines sont des solutions de couche L3 et les autres de couche L2. Cependant, ces solutions ne répondent pas à l'ensemble des contraintes. Les solutions de couche L3 qui répondent à la majorité de ces contraintes ont, soit des configurations importantes, soit des difficultés à migrer des VMs. En effet, lors de la migration d'une VM, certaines solutions L3 modifient l'adresse IP de la VM résultant en la coupure de ses connexions. D'autres solutions L3 utilisent des agents de redirection à configurer dans chaque data center afin de rediriger le trafic vers le nouvel emplacement de la VM. Il en résulte une augmentation de la taille du chemin et du trafic au sein du réseau.

Au sein de la couche L2, la migration à chaud d'une VM n'implique pas de rompre les connexions de la VM car la VM ne change pas d'adresse IP. De plus, la découverte du changement d'emplacement de la VM ainsi que la redirection de son trafic se font automatiquement. Cependant, d'autres problèmes apparaissent comme le problème d'interconnexion des data centers et l'utilisation du multi chemin. En effet, si l'on utilise une solution de couche L2 au sein des data centers et que l'on communique toujours via la couche L2 entre les data centers, on perd l'indépendance des plans de contrôle de chaque data center. Il en résulte un unique domaine de broadcast qui s'étend sur l'ensemble des data centers ce qui n'est pas souhaitable à cause de l'overhead important que cela entrainerait. De fait, la moindre requête ARP réalisée au sein d'un data center serait inondée sur l'ensemble des data centers créant ainsi du trafic inutile, ce qui dégraderait les performances du cloud. Cependant, l'utilisation d'une solution de couche L2 nous permet de bénéficier de la découverte automatique du nouvel emplacement de la VM ainsi que de la redirection de son trafic une fois que la VM a migré. Néanmoins, nous devons trouver une solution afin d'interconnecter les data centers tout en conservant l'indépendance de leurs plans de contrôle pour éviter une chute importante des performances ainsi qu'une solution pour maximiser les débits entre les nœuds. L'indépendance des plans de contrôles nous permet d'avoir des gestions, des règles et des administrateurs différents au sein de chaque data center. Cela rend possible l'interconnexion de data centers administrés par des entités différentes et donc de réaliser un cloud hybride ou un cloud fédéré. De plus, cette solution multiple data centers doit aussi être multi-utilisateurs et donc fournir une isolation du trafic de chaque utilisateur au sein de la totalité du réseau cloud.

1.2 Contributions

Notre contribution consiste en la conception d'une architecture répondant à toutes ces contraintes. Cette architecture doit entre autres, permettre d'optimiser l'utilisation de l'infrastructure réseau déjà existante afin d'accommoder un trafic plus important au sein d'un data center ainsi que de gérer un grand nombre d'utilisateurs. De plus, la solution doit fournir de la sécurité aux utilisateurs du cloud hybride. En effet, il est nécessaire, afin de répondre aux craintes des utilisateurs, de fournir une solution sécurisée pour la partie cloud public du cloud hybride. Or, au sein du cloud public, les utilisateurs partagent les infrastructures, contrairement au cloud privé. Afin de sécuriser le trafic de chaque utilisateur, il est nécessaire de l'isoler de celui des autres utilisateurs. Ainsi, l'utilisateur possédera, au sein du cloud public, la même sécurité que celle qu'il possède au sein d'un cloud privé hébergé par une tierce partie. De plus, l'isolation réseau dans un environnement cloud public/hybride permet à l'opérateur de cloud de fournir des solutions personnalisées pour chacun de ses clients. En effet, grâce à l'isolation du trafic d'un client, l'opérateur de cloud peut vérifier et prendre les mesures adéquates afin de faire respecter le SLA (Service Level Agrement) de ce client ainsi que la QoS (Quality of Service) qu'il lui a vendue.

Les travaux de cette thèse portent donc sur l'établissement d'une solution permettant de réaliser un environnement cloud public/hybride tout en fournissant une solution d'isolation réseau qui isole le trafic de chaque utilisateur au sein de cet environnement. Cette solution a été découpée en quatre contributions lors de la réalisation de cette thèse.

Notre première contribution (Chapitre 3) consiste en la conception d'un protocole de transport de couche L2 qui sera utilisé au sein des data centers. Ce protocole doit maximiser l'utilisation des ressources réseau de l'infrastructure du data center permettant ainsi d'accroître les débits à l'intérieur de celui-ci et donc de s'accommoder de l'augmentation du trafic général. Pour cela, nous nous sommes basés sur le protocole TRansparent Interconnection of Lots of Links (TRILL)[RFC 6325] [11] que nous décrivons dans la Section 2.4. TRILL permet, de par sa conception, d'abandonner le Spanning Tree Protocol (STP), de pouvoir établir un ou plusieurs chemin(s) entre n'importe quel couple de nœuds et de ne pas former de boucle. Cependant, le protocole TRILL a été conçu pour des réseaux Local Area Network (LAN) et n'est donc pas adapté à un réseau multi data centers. Nous devons donc adapter le protocole TRILL pour pouvoir l'utiliser au sein d'un réseau cloud et ce nouveau protocole est nommé Multi Level Trill Protocol (MLTP) [12]. Le protocole MLTP permet de conserver le plan de contrôle de chaque campus MLTP interconnecté indépendant, contrairement au protocole TRILL qui, lui, fusionne tous les campus TRILL dès qu'ils sont interconnectés.

Une fois le protocole MLTP implémenté, notre seconde contribution (Chapitre 4) consiste en l'ajout d'une solution d'isolation des utilisateurs à MLTP. Nous avons, pour cet objectif, comparé plusieurs solutions d'isolation dans [13] et avons choisi la solution

Virtual Network over TRILL (VNT) [14] qui nous semblait la plus adaptée à notre approche. Nous avons donc intégré la solution VNT dans MLTP et ainsi fourni une isolation de chaque utilisateur au sein du réseau. De plus, nous avons intégré une structure d'enregistrement des informations d'isolation au sein des gateways des réseaux (campus) de telle sorte que l'on peut savoir dans quel campus un VNI est utilisé. Ceci nous permet d'éviter d'envoyer du trafic inutile sur les liens inter campus et donc d'optimiser leur utilisation.

Notre troisième contribution (Chapitre 6) consiste à concevoir un cloud hybride tout en isolant les utilisateurs. Sachant que la définition d'un cloud hybride est un cloud qui utilise deux types de clouds (Section 2.2), nous sommes partis du postulat que le cloud public est celui qui utilise MLTP+VNT et que le cloud privé utilise une solution centralisée. Nous avons choisi OpenFlow comme solution centralisée pour le cloud privé et avons réalisé une interconnexion entre un cloud public utilisant MLTP+VNT et un cloud privé utilisant OpenFlow. Cette interconnexion a nécessité le développement d'une gateway permettant de faire la conversion entre des paquets OpenFlow et des paquets MLTP+VNT [15].

Au sein de notre quatrième contribution (Chapitre 5), nous nous sommes focalisés sur la résilience du protocole MLTP+VNT. En effet, de par les modifications que nous avons réalisées au sein des deux premières contributions, certains équipements, les gateways, sont devenus essentiels pour la communication inter campus. Or, si ces équipements tombent en panne, nous devons pouvoir envoyer le trafic sur d'autres chemins le plus efficacement possible. Pour cela, nous avons conçu une méthode de synchronisation des informations de routage entre les gateways d'un campus permettant ainsi d'améliorer la gestion d'une panne d'une de ces gateways.

1.3 Plan de la thèse

Au sein de ce manuscrit, nous allons, dans un premier temps, détailler l'environnement déjà existant dans lequel les travaux de recherche de cette thèse se sont déroulés. Le chapitre 2 définit la terminologie utilisée au sein de ce manuscrit et présente les protocoles sur lesquels nous nous sommes basés pour nos travaux. Ensuite, le chapitre 3 introduit la problématique de l'interconnexion des data centers (ou campus) lorsque l'on utilise le protocole TRILL. Dans ce même chapitre, nous présentons un état de l'art sur les solutions proposées pour réaliser cette interconnexion, puis nous expliquons pourquoi ces solutions ne nous conviennent pas. Enfin, nous présentons les deux étapes de conception qui nous ont permis d'aboutir à notre solution ainsi que les performances de cette solution.

Dans le chapitre suivant (chapitre 4), nous définissons la problématique concernant l'isolation des utilisateurs au sein du réseau, puis nous réalisons un état de l'art sur différentes solutions d'isolation A. Finalement, nous présentons notre contribution, à savoir l'intégration de la solution VNT au sein de MLTP, ainsi que les performances obtenues par cette nouvelle solution.

Le chapitre 6 se focalise sur la réalisation d'un cloud hybride avec un cloud privé utilisant OpenFlow et un cloud public utilisant MLTP+VNT. La problématique de ce chapitre peut se résumer à une problématique d'interconnexion entre deux environnements utilisant des technologies différentes. Néanmoins, cette interconnexion doit être transparente pour l'utilisateur. Dans le chapitre 5 nous améliorons la résilience du protocole MLTP+VNT en lui ajoutant un système de synchronisation des informations de commutation entre les nœuds qui jouent le rôle de gateway des campus. Grâce à cette synchronisation, lorsqu'une gateway tombe en panne et que le trafic au sein du campus est redirigé sur une autre gateway, celle-ci possède les informations de commutation associées à ce flux et peut donc commuter les paquets sans avoir à réaliser une nouvelle recherche de la destination.

Chapitre 2

Environnement existant

Sommaire

2.1 Introduction

Ce chapitre présente l'environnement dans lequel doit s'insérer mon travail de recherche. Je vais présenter succinctement les éléments déjà présents au début de ma thèse et sur lesquels j'ai dû baser mes recherches pour que mes contributions puissent être utilisées. Ma thèse fait suite à la thèse du docteur Ahmed Amamou [14] qui a développé un protocole appelé VNT (Virtual Network over TRILL) basé sur le protocole TRILL (TRansparent Interconnection of Lots of Links). Ce protocole VNT est actuellement testé au sein d'un data center de Gandi. Mon travail de recherche se base donc sur l'utilisation de ce protocole et l'extension de son utilisation. Je vais commencer par présenter des définitions de termes récurrents dans le cloud computing dans la section Terminologie. Dans un second temps, j'expliquerai le choix du protocole TRILL et de la solution VNT grâce à un état de l'art comprenant une comparaison de plusieurs solutions permettant l'isolation des utilisateurs au sein d'un data center. Ensuite, je présenterai le protocole TRILL puis j'introduirai le protocole de routage IS-IS (Intermediate System to Intermediate System) qui est utilisé au niveau du plan de contrôle de TRILL. J'indiquerai les modifications apportées au protocole IS-IS pour son utilisation avec le protocole TRILL. Enfin, je présenterai le protocole VNT qui se base sur TRILL et sur IS-IS tout en permettant d'avoir un plus grand nombre de clients isolés dans un site TRILL.

2.2 Terminologie

Dans cette section on introduit les définitions des termes cloud privé, cloud communautaire, cloud public et cloud hybride établies par le NIST (National Institute of Standards and Technology) [16] ainsi que d'autres termes utilisés dans ce document.

2.2.1 Cloud Privé

Le NIST définit le cloud privé comme étant une infrastructure de type cloud mise à la disposition d'un unique client, quel que soit le client (organisation, société, département d'une société ou une personne). Cette infrastructure peut appartenir, être maintenue et opérée par le client, une tierce entité ou une quelconque combinaison. Néanmoins, cette infrastructure doit être déployée au sein du réseau du client ou alors au sein du réseau de l'entité tierce, mais ne peut pas s'étendre sur les deux réseaux. Ci-après, nous avons reproduit la définition originale du NIST.

Private cloud. The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

2.2.2 Cloud Communautaire

Le cloud communautaire établit une infrastructure utilisée exclusivement par les membres d'une communauté partageant des intérêts communs. L'infrastructure peut être possédée, maintenue et opérée par un ou plusieurs membres de cette communauté, ou par une tierce partie ou une quelconque des combinaisons possibles. Comme pour le cloud privé, l'infrastructure peut être au sein d'un réseau d'un des membres ou alors au sein d'un prestataire tiers, mais il peut aussi être au sein des deux réseaux.

Community cloud. The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

2.2.3 Cloud Public

Le cloud public, par opposition au cloud privé, est une infrastructure mise à disposition des clients ou des utilisateurs, de telle façon qu'ils ont tous accès à la même infrastructure. Cette infrastructure est la propriété d'une entreprise ou d'une organisation et elle est maintenue et gérée par son propriétaire. Elle se situe au sein du réseau de son propriétaire.

Public cloud. The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

2.2.4 Cloud Hybride

Le cloud hybride est, comme son nom l'indique, l'association de deux des trois types de clouds présentés précédemment. Chaque cloud reste une infrastructure propre, mais les deux clouds sont reliés par un protocole standard ou propriétaire qui permet la portabilité des applications et des données.

Hybrid cloud. The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology

that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

2.2.5 Data center

Le data center est, selon Gartner [17], le département d'une entreprise qui héberge et maintient les systèmes informatiques ainsi que les données informatiques. Ce département et ses systèmes résident habituellement à un seul endroit d'où le nom data center (centre de données).

Pour CIO [18] un data center est le lieu où se trouve et est gérée la majorité des serveurs ainsi que le stockage des données d'une entreprise.

Dans ces deux définitions, on retrouve la notion de lieu ainsi que de gestion. On peut en conclure qu'un data center est un lieu de stockage informatique où le matériel informatique est installé, utilisé et maintenu en état de marche.

2.2.6 Multitenancy

Le terme *multitenancy* est un terme anglais dérivé des termes *tenant* (locataire, utilisateur) et *multiple*. Il caractérise le fait qu'il y ait plusieurs utilisateurs qui se partagent l'infrastructure d'un data center. D'après Cisco [19], *virtualized multitenancy* est un concept clé qui réfère à l'isolation logique des ressources partagées de l'infrastructure. Juniper [20] partage aussi cette définition tout comme TheForce.com [21] qui utilise la métaphore de l'immeuble, dans laquelle l'infrastructure de l'immeuble est partagée par l'ensemble des locataires résidants dans celui-ci. L'immeuble est donc *multitenant*.

2.3 État de l'art

Dans l'article "A Survey of network isolation solutions for multi-tenant data centers" [13] reproduit dans l'annexe A, nous avons réalisé une étude comparative de quinze solutions permettant l'isolation du trafic des utilisateurs au sein d'un réseau partagé. Ces solutions permettent l'établissement de réseaux *multitenant*.

Nous avons, dans un premier temps, divisé ces solutions en fonction de leur ancienneté en regroupant toutes les solutions déjà connues et utilisées dans l'industrie d'un côté et de l'autre, toutes les solutions que l'on considère comme récentes et/ou encore peu utilisées. Dans un second temps, nous avons regroupé les protocoles dans un groupe et les solutions complètes, que l'on appelle architectures dans l'article, dans un second groupe. Puis, nous avons séparé les architectures en fonction de la couche avec laquelle elles fonctionnent. Le regroupement des solutions est présenté dans le Tableau 2.1.

	Protocoles	Architectures		
		Couche 2	Couche 2-3	Couche 3
Solutions connues	• GRE : Generic Routing Encapsulation [9] • PPTP : Point-to-Point Tunneling Protocol [22] • L2TP : Layer Two Tunneling Protocol [23] • L2TPv3 : Layer Two Tunneling Protocol - Version 3 [24] • PWE3 : Pseudo Wire Emulation Edge-to-Edge [25] • VLAN : Virtual LAN [10] • MPLS : Multiprotocol Label Switching [26] • GMPLS : Generalized Multi-Protocol Label Switching [27] • BGP/MPLS IP Virtual Private Networks [28]			
Solutions récentes	• LISP : The Locator/Identifier Separation Protocol [29] • NVGRE : Network Virtualization using Generic Routing Encapsulation [30] • STT : Stateless Transport Tunneling Protocol [31] • 802.1ad or QinQ [32] • 802.1ah or mac-in-mac [33] • Private VLANs [34] • VRF : Virtual Routing and Forwarding [35] • VXLAN : Virtual eXtensible Local Area Network [36]	• Diverter [37] • PortLand [38] • SEC2 [39] • BlueShield [40]	• VSITE [41] • NetLord [42] • VNT [43]	• VL2 [44] • DOVE [45]

TABLE 2.1 – Regroupement des solutions étudiées.

Nous avons ensuite divisé les protocoles en nous basant sur leur méthode d'isolation. Nous avons distingué deux méthodes d'isolation pour isoler les utilisateurs et leur trafic. Ces méthodes sont : *Host isolation* et *Core isolation*.

La première méthode appelée *Host isolation*, ou *Isolation au niveau de l'hôte* en français, sélectionne et applique l'isolation des flux lorsque ceux-ci arrivent à l'hôte de destination. Ceci signifie qu'aucun équipement sur le chemin d'un flux ne va vérifier si sa destination l'accepte ou non. Ces équipements ne se préoccupent que de faire transiter ou router les paquets du flux afin d'atteindre la destination. Ce n'est seulement qu'une fois le paquet arrivé à destination que la machine destination vérifie si elle peut ou non accepter le paquet. Il en résulte que l'isolation des trafics utilisateurs ne se fait pas grâce aux équipements intermédiaires (middleboxes) du réseau, mais grâce aux machines hôte. Ces dernières acceptent ou rejettent les paquets en fonction de l'isolation à appliquer en vérifiant si la VM destination appartient ou non au même utilisateur que la VM source. Cette méthode d'isolation permet de conserver des équipements réseau (les middleboxes) simples, car ils n'ont pas à connaître la topologie complète du réseau. En effet, comme ils ne font pas respecter l'isolation des flux, ils n'ont pas à savoir si la source et la destination d'un paquet appartiennent bien au même utilisateur. Cependant, cette méthode d'isolation possède deux désavantages qui sont :

— L'absence d'information sur l'isolation des utilisateurs de la part des équipements réseau implique que la totalité des paquets transite dans le réseau quand bien même si ces derniers sont rejetés à leur destination. Cela ne permet donc pas de réduire le trafic inutile au sein du réseau.

— La sensibilité à des attaques de type *man in the middle*. En effet, comme le trafic n'est pas restreint au sein du réseau, une sonde sur un nœud peut entendre la totalité du trafic qui transite par ce nœud.

Les protocoles que nous avons identifiés comme utilisant cette méthode d'isolation sont : Generic Routing Encapsulation (GRE) [9], Point-to-Point Tunneling Protocol (PPTP) [22], Layer Two Tunneling Protocol (L2TP) [23], Layer Two Tunneling Protocol - Version 3 (L2TPv3) [24], Pseudo Wire Emulation Edge-to-Edge (PWE3) [25], The Locator/Identifier Separation Protocol (LISP) [29], Network Virtualization using Generic Routing Encapsulation (NVGRE) [30] et Stateless Transport Tunneling Protocol (STT) [31].

La seconde méthode d'isolation, que nous avons nommée *Core isolation*, possède un fonctionnement opposé à la première méthode d'isolation. En effet, elle se base sur des équipements réseau "intelligents" qui connaissent l'ensemble de la topologie du réseau et qui vont vérifier, avant l'envoi d'un paquet, si le nœud suivant sur le chemin accepte ou non le paquet en fonction de l'isolation de l'utilisateur. En utilisant cette méthode, il est donc nécessaire que la totalité des nœuds, sur le chemin entre deux VMs d'un même utilisateur, accepte les paquets de cet utilisateur. Il faut donc que les nœuds soient informés qu'ils se

trouvent sur un chemin utilisé par cet utilisateur et qu'ils acceptent son trafic. Cela impose que la totalité des nœuds du réseau possède une vue globale du réseau afin de savoir quel trafic ils doivent faire transiter ou rejeter. Pour cela, les nœuds du réseau doivent être, soit préconfigurés, soit auto configurables. Dans le premier cas, la topologie du réseau est rigide et toute modification de celle-ci devient extrêmement complexe. Dans le second cas, il est nécessaire que les équipements réseau découvrent la totalité du réseau, ce qui augmente leurs temps d'initialisation et le temps d'attente avant de pouvoir utiliser le réseau. Cependant, une fois configurés, les nœuds seront capables de restreindre le trafic d'un utilisateur aux seuls nœuds situés sur les chemins entre ses VMs. Il en résulte que le trafic inutile est supprimé, d'où une réduction du trafic total à l'intérieur du réseau. Les protocoles qui utilisent cette méthode d'isolation sont : Virtual LAN (VLAN) [10], Multiprotocol Label Switching (MPLS) [26], Generalized Multi-Protocol Label Switching (GMPLS) [27], BGP/MPLS IP Virtual Private Networks [28], 802.1ad ou QinQ [32], 802.1ah ou mac-in-mac [33], Private VLANs [34], Virtual Routing and Forwarding (VRF) [35] et Virtual eXtensible Local Area Network (VXLAN) [36].

Dans notre article, nous détaillons l'ensemble des solutions présentées dans le Tableau 2.1 et nous nous focalisons sur la comparaison de quinze solutions parmi les solutions récentes à savoir : 1. LISP, 2. NVGRE, 3. STT, 4. 802.1ad, 5. 802.1ah, 6. VXLAN 7. Diverter, 8. PortLand, 9. SEC2, 10. BlueShield, 11. VSITE, 12. NetLord, 13. VNT, 14. VL2, 15. DOVE. Nous avons choisi de comparer ces solutions, car elles sont les plus intéressantes du point de vue de l'isolation d'un grand nombre d'utilisateurs. En effet, l'ensemble des protocoles connus étudiés dans l'article possèdent des limitations au niveau du passage à l'échelle qui empêche leur utilisation au sein des réseaux cloud. Par exemple, le provisionnement de GRE ne passe pas à l'échelle et il en est de même pour le nombre maximum de VLANs (4096). Seul MPLS peut passer à l'échelle, cependant, la complexité de la distribution des labels dans le réseau ainsi que son coût ne le rendent pas attractif pour les réseaux cloud.

Notre comparaison de ces quinze solutions a été réalisée en six parties. Nous avons donc, dans un premier temps, comparé la complexité des solutions en étudiant les six critères suivants :

1 La solution est-elle centralisée ou distribuée ?

2 Les restrictions imposées au réseau sous-jacent.

3 La configuration et l'établissement des tunnels.

4 La gestion et l'entretien des tunnels.

5 La solution accepte-t-elle de multiples protocoles ?

6 Quels sont les mécanismes de sécurité de la solution ?

Dans un second temps,nous avons comparé l'overhead induit par chaque solution. Ensuite, nous avons analysé les capacités de chaque solution à migrer les VMs puis leur résilience

et leur passage à l'échelle. Le dernier critère de comparaison est la possibilité d'avoir de multiples data centers et la facilité de gérer cette interconnexion. Les Tableaux 2.2 et 2.3, extrait de l'article, résument la comparaison de ces quinze solutions en se basant sur les quatre derniers critères.

Au cours de notre étude et comparaison de ces quinze solutions, nous avons constaté que les solutions ayant un plan de contrôle centralisé ont de plus grandes difficultés pour passer à l'échelle que les solutions distribuées. Cela s'explique par le fait que les solutions centralisées utilisent des contrôleurs centralisés pour gérer l'ensemble des messages de contrôle de l'infrastructure. Or, l'exemple de la solution Portland montre que, pour gérer uniquement les requêtes ARP dans un réseau avec plus de 27000 hôtes, il est nécessaire d'avoir un contrôleur avec plus de 15 CPUs utilisés en permanence. Par opposition, les solutions distribuées n'ont pas se problème mais elles doivent fournir un protocole d'échange d'information topologique entre les nœuds. Chacun de ces nœuds doit ensuite calculer la topologie.

Un autre élément que nous avons constaté est la difficulté à réaliser une migration à chaud lorsque l'on utilise une solution opérant dans la couche 3. Ces solutions doivent recourir à des agents de redirection afin de conserver les connexions de la VM en redirigeant le trafic vers le nouvel emplacement de la VM. Sans ces agents de redirection, la migration implique la rupture des connexions de la VM et donc l'impossibilité de réaliser une migration à chaud de VM. Au contraire, les solutions de niveau 2 peuvent migrer plus aisément à chaud les VMs, car elles n'ont pas besoin d'un agent de redirection. Cependant, cela implique une unicité du plan de contrôle entre tous les data centers afin de pouvoir découvrir et informer les nœuds de cette migration de VM.

Un troisième point intéressant, détaillé dans l'article, est la capacité de ces quinze solutions à interconnecter de multiple data centers. Nous avons constaté que, pour un grand nombre (6) de ces solutions, aucune mention n'était faite sur leur possibilité d'interconnexion de data centers. Au contraire, trois solutions (LISP, SEC2 et VSITE) sont par conception des solutions multi sites. Enfin, pour les solutions restantes, l'interconnexion de site implique une fusion des plans de contrôle et donc une perte d'indépendance pour chaque data center.

En conclusion de notre veille, nous avons déterminé que la solution la plus prometteuse pour la réalisation de nos travaux est la solution VNT basée sur TRILL. Cette solution permet de réaliser la migration à chaud des VMs sans perte de connexion, elle fournit aussi une solution d'isolation qui passe à l'échelle, car elle permet d'isoler plus de 16 millions d'utilisateurs. De plus, de par sa conception il est possible d'interconnecter des data centers cependant, cette interconnexion résulte en la fusion de tous les plans de contrôle des data centers en un unique plan de contrôle. Dans les sections suivantes de ce chapitre, nous détaillons le protocole TRILL et la solution VNT ainsi que leur fonctionnement.

TABLE 2.2 – Comparaison des solutions utilisant la méthode Host isolation

Host isolation	**Diverter**	**BlueShield**	**NetLord**	**VL2**
Migration	Migration live or offline depending on time out values.	Live migration.	Live migration. Uses NetLord Agent messages (NLA) : NLA-HERE, NLA-NOTHERE and NLA-WHERE to signal the VM migration. VM's IP or MAC address unchanged.	Live migration. Separation of location Addresses (LA) and application-specific addresses (AA).
Resilience	ECMP for multipath. Virtual gateway distributed among all the VM of the Sub-network.	Multiple replicas of the directory server. Possibility to use ECMP.	Relies on SPAIN for multipath. Configuration Repository might be replicated.	Resilience provided by a Clos topology. Redundancy of the Directory server
Scalability	16 millions VMs system wide. However number of client depend on the division of the IP address. The division must be done before starting the network, no modification after.	Centralized controller. CPU load lessen by replicating directory server but memory is limited.	16777216 Tenant_IDs (24 bits) $V \times R \times \sqrt{(\frac{F}{2})}$ virtual machines V = number of VMs per physical server R = switch radix, F = FIB size in entries.	One directory server can manage up to 17000 lookups/sec. The lookup rates increase linearly with the increase of servers.
Multi data center	Not specified. Possible with a Layer 2 interconnection. Control traffic travel between DC. Creates one big network over multiple DC	Not specified. Possibly a directory server replicated in each data center for inter-data centers communication rules.	Not specified. But possible with Layer 2 tunnels between data centers and one control plane spanning over all the data centers.	Not specified but possible. It will require an important directory system to manage the whole network.

Host isolation	**DOVE**	**LISP**	**NVGRE**	**STT**
Migration	Tunneling protocol dependent. Additionally dSwitch must inform the DPS when a new VM is detected by it.	IPv4 Mobility (RFC5944), IPv6 Mobility (RFC 6275, RFC 4866). Endpoint is an xTR itself.	REDIRECT messages	No STT mechanisms
Resilience	Multipath and routing resilience thanks to tunnel protocol. Redundancy of the Dove Policy Server.	Redundancy of xTR, MR and MS.	Multipath possible but not included in NVGRE	Multipath (ECMP) possible but not included in STT
Scalability	Number of tenants is tunnel protocol dependent : VXLAN has a 24 bits long VNI ≈ 16000000, NVGRE also has a 24 bits long VSID and STT has a 64 bits long Context ID $\approx 1.8 \times 10^{19}$. DPS is the scalability limiting component.	Big number of EIDs and RLOCs possible. One RLOC address associated with multiple EIDs addresses. Issue with the MS and MR maximum information saved.	One PA associated with multiple CA. Suppress most of the control plane broadcasts messages and convert some of them in multicast messages.	Context ID fields is 64 bits long. Issue with the virtual switch which can not manage this much IDs.
Multi data center	Not specified but possible. It will require an important DPS to manage the whole network.	Yes even with non LISP data center	Yes as a site-to-site VPN. Each site must have a NVGRE gateway	Theoretically, yes as a site-to-site VPN. Practically, no because of the middle boxes issue

TABLE 2.3 – Comparaison des solutions utilisant la méthode Core isolation

Core isolation	**PortLand**	**SEC2**	**VSITE**	**VNT**
Migration	Live migration thanks to gratuitous ARP. Possibility of lessening the number of lost packets with redirection.	Live migration thanks to gratuitous ARP.	Live migration if the VM stays in the same location otherwise offline migration.	Live migration. Based on TRILL or MLTP which uses RBridge nicknames for forwarding the messages so VM's IP or MAC address unchanged.
Resilience	Fat-tree topology induced resilience. Fabric manager back up even with slightly non identical information.	Multiple FEs. Backups of Central Controller (CC). Can uses a Distributed controller instead of CC.	Master/slave switches configuration with the virtual router redundancy protocol.	ECMP for multipath. Redundant multicast distribution tree. No centralized controller.
Scalability	Centralized controller. Huge stress on Fabric manager. Not scalable by default : $\approx$ 27000 hosts with 25 ARP request/second = Fabric Manager with 15 CPUs.	Centralized controller. Huge stress on Centralized Controller. Possibility to transform the CC in a Distributed Controller. Only 4096 VLANs by edge domain. Number of edge domain is not limited but depend on MAC address usage.	VLAN for isolation. Aggregation of multiple VMs under a locIP.	Multiple VMs MAC addresses aggregated under one RBridge nickname. VNI TAG (24 bits) allows for 16777216 virtual networks.
Multi data center	Not specified. Layer 2 Fat-tree topology to interconnect the core switches of each data centers and get one network spanning over multiple DC. Not really feasible in reality seeing the cost induced by the interconnection topology.	Multi domains by design but scalability issue, only 4096 VLANs per domain.	Multi sites by design but scalability issue, only 4096 VLANs per sites.	Ready for multi data center. When using TRILL it creates one big network with one control plane spanning over all the data centers. Whereas MLTP keeps each data center independent. Needs Layer 2 tunnels between data centers.

Core isolation	**802.1ad**	**802.1ah**	**VXLAN**
Migration	Need to allocate resources for the VLAN in the destination network ahead of time to have session continuity. Migration restricted to the same Layer 2 network.	Need to allocate resources for the VLAN in the destination network ahead of time to have session continuity. Migration restricted to the same Layer 2 network.	Need to allocate resources for the VXLAN in the destination network ahead of time to have session continuity. Migration across Layer 3 network possible.
Resilience	Link aggregation and switches redundancy.	Link aggregation and switches redundancy	VTEP in hypervisor so redundancy of server in order to migrate the VMs to a new server if the hypervisor is down.
Scalability	VLAN limit up to 16777216	VLAN limit up to $\approx 7 \times 10^{16}$	16777216 VNI possible.
Multi data center	Yes with the same VLANs on all data center.	Yes with the same VLANs on all data center.	Possible to use VXLAN as a site-to-site VPN with VTEP gateways.

2.4 TRILL

Le protocole TRansparent Interconnection of Lots of Links (TRILL) a été standardisé par le RFC 6325 [11, 46]. L'idée de TRILL est d'utiliser des techniques de routage de couche 3 pour concevoir un grand ensemble de liens qui ressemble à un seul sous-réseau IP pour les nœuds IP. Tous les nœuds de la couche 2 de cet ensemble peuvent être mobiles à l'intérieur de cet ensemble sans avoir à changer leur adresse IP, car le routage s'effectue via les adresses MAC. Ce routage bénéficie des techniques de la couche 3 tels que le chemin le plus court entre les communicants et le multi chemins. En plus de ces techniques de routage, TRILL supporte les Virtual Local Area Networks (VLANs) , possède la capacité de s'auto configurer et permet le multicast/broadcast sans recourir à des protocoles additionnels. Un autre avantage de TRILL se situe au niveau de la taille de la table de forwarding dans les RBridges de transit. Celle-ci est réduite, car elle ne contient que les nicknames des RBridges et non plus les adresses MAC des correspondants. TRILL permet aussi de se passer du protocole de Spanning Tree (STP) et donc de conserver tous les liens physiques du réseau. Le protocole TRILL fonctionne au niveau de la couche 2. Cependant, il nécessite des équipements particuliers que sont les RBridges en remplacement des bridges "classiques". Ces RBridges sont compatibles avec les bridges respectant la norme 802.1, l'IPv4 et l'IPv6 ainsi que les nœuds terminaux. Vis-à-vis des routeurs IP , les RBridges ont le même comportement que les bridges ; ils sont invisibles et ils terminent le protocole spanning tree au niveau des bridges.

2.4.1 RBridges nicknames

Les RBridges ont, pour adresse, des nicknames sur 16 bits qui sont, soit choisis par le RBridge, soit configurés par un administrateur. Le nickname doit être unique au sein du réseau TRILL. Comme les RBridges peuvent choisir des nicknames, et bien que cette sélection soit faite aléatoirement parmi les nicknames disponibles d'après les Link State Packets (LSPs) que le RBridge a déjà reçu, il peut se produire des collisions. Lors de ces collisions, on utilise un système de priorité, une valeur sur 8 bits, pour indiquer quel RBridge doit conserver le nickname. Par exemple :
Si le nickame (N1) a été configuré manuellement sur un RBridge (A), celui-ci possède alors la priorité maximale pour le conserver. Ceci implique que si un autre RBridge (B) choisit ce nickname (N1), il y aura alors une collision de nickname entre les RBridges A et B. Comme A possède la priorité maximale pour conserver N1 et que ce n'est pas le cas de B, qui a choisi N1 au hasard, le RBridge B doit changer de nickname pour conserver l'unicité du nickname au sein du site TRILL.
Si cette collision de nicknames se produit avec deux RBridges ayant la même priorité pour conserver le nickname, alors, le RBridge ayant l'identifiant IS-IS le plus grand conserve le nickname. Une fois que le RBridge a son nickname et qu'il n'y a plus de collision de

nicknames avec d'autres RBridges, alors, il doit essayer de réutiliser ce nickname lors de son prochain démarrage. De plus, plus le RBridge garde longtemps le nickname et plus sa priorité pour le conserver augmentera, mais cette priorité n'atteindra jamais la valeur maximale réservée aux RBridges dont le nickname a été configuré. Par défaut, un RBridge ne possède qu'un seul nickname pour l'identifier. Toutefois, il peut posséder plusieurs nicknames pour dissocier ses différents arbres de distribution.

2.4.2 Encapsulation

Le protocole TRILL utilise un en-tête TRILL qui est ajouté aux messages par le premier RBridge sur le chemin du message, le Ingress RBridge , et est retiré par le dernier RBridge, le Egress RBridge . Cet en-tête a une taille par défaut de 64 bits. Cependant, cette taille est variable, due à la présence possible d'options. Au maximum, on peut obtenir 124 octets d'options correspondant à une taille maximale pour l'en-tête de 132 octets soit 1056 bits. La Figure 2.1 décrit l'en-tête TRILL.

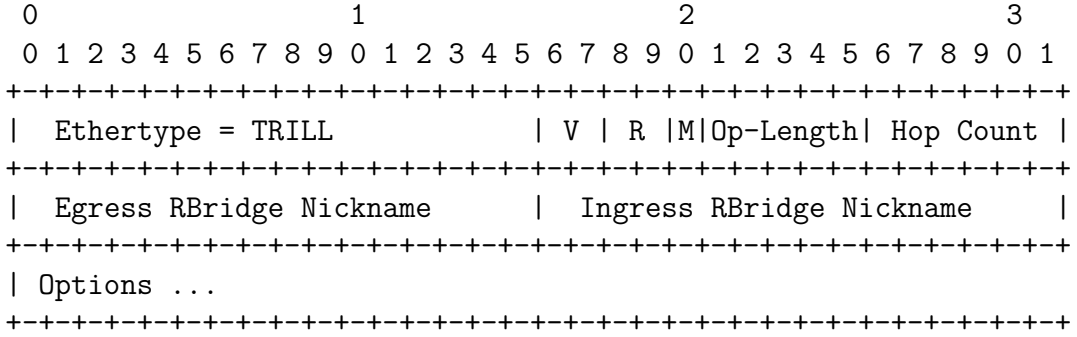


FIGURE 2.1 – En-tête TRILL

Les différents champs de l'en-tête sont :

— V : Version du protocole TRILL utilisé.

— R : Réservé pour de futures extensions.

— M : Destination Multiple indique si le message doit être délivré à un groupe de destinations en utilisant l'arbre de distribution indiqué par le Egress RBridge nickname.

— OP-Lenght : Longueur des options en unités de 4 octets.

— Hop Count : Le nombre de sauts correspond au nombre de sauts que le paquet peut effectuer avant d'être rejeté. La valeur initiale doit être supérieure au nombre de sauts requis pour parcourir le chemin.

— Egress RBridge nickname : le nickname du RBridge destination.

— Ingress RBridge nickname : Le nickname du RBridge source.

— Options : S'il y a des options, il faut que le champ commence par 8 bits de flags.

— Le premier bit CHbH (Critical Hop by Hop) mis à un indique la présence d'options critiques à chaque saut. Ceci implique que les RBridges de transit doivent supporter ces options pour que le message puisse continuer sa route.

— Le second bit CItE (Critical Ingress-to-Egress) indique la présence d'options critiques de bout en bout.

Ce message TRILL n'est pas modifié du Ingress RBridge jusqu'au Egress RBridge. Néanmoins entre chaque RBridge de transit il est encapsulé dans une trame Ethernet pour permettre le transport de cette trame sur des réseaux contenant des bridges et switchs qui ne sont pas compatibles avec TRILL.

2.4.3 Protocole TRILL-Hello

Le protocole TRILL-Hello est basé sur le protocole IS-IS LAN Hello de couche 3. Néanmoins, à la différence de IS-IS, dans le protocole TRILL-Hello, le Designated RBridge (DRB) est élu seulement sur sa priorité et son adresse MAC. Les messages TRILL-Hello sont un nouveau type de message IS-IS. Ils commencent avec le même en-tête qu'un message IS-IS LAN Hello et sont envoyés avec la même périodicité. Les messages TRILL-Hello ne doivent pas dépasser 1470 octets et contrairement aux messages IS-IS LAN Hello, ils ne doivent pas être complétés (padded). Le fait de compléter le message Hello dans IS-IS permet de ne garder que les liens de capacité maximale. En faisant cela dans TRILL-Hello, on risque de créer des cliques de routeurs entrainant alors la création de multiples pseudonodes. Pour éviter cela, cette contrainte n'est pas utilisée dans TRILL-Hello permettant ainsi de découvrir tous les voisins.

C'est par ces messages que le DRB désigne les RBridges diffuseurs pour les différents VLANs. Néanmoins, dans les data centers, les RBridges sont connectés en point à point. Il n'est donc pas nécessaire d'avoir un pseudonode créé par le DRB sur un lien point à point. Il est même plus intéressant de ne pas en avoir puisque cela évite d'avoir des messages de contrôle supplémentaires. Pour éviter la création d'un pseudonode sur les liens point à point, le DRB doit activer le *bypass pseudonode bit*. Le DRB est configuré pour activer le *bypass pseudonode bit* par défaut. Il le désactive à partir du moment où il détecte deux nœuds adjacents (deux voisins) simultanément.

S'il y a plusieurs voisins sur le même lien local, il est alors nécessaire d'élire un DRB ; le RBridge, avec la priorité la plus importante sur le lien, est élu DRB. Si la valeur de la priorité ne permet pas de départager les RBridges, ce sont alors les adresses MACs des RBridges qui sont utilisées. Le DRB, sur le lien où il est élu, va gérer les différents RBridges en associant à un RBridge la tâche de diffuser un certain VLAN sur le lien. Le DRB peut s'occuper de diffuser tout le trafic de tous les VLANS ou alors se décharger de cette tâche en désignant, pour cela, d'autres RBridges.

2.4.4 Arbre de distribution

Les arbres de distribution sont utilisés par les RBridges pour diffuser les messages avec destinations multiples. Un arbre de distribution est bidirectionnel et un seul arbre est suffisant pour un campus entier. Néanmoins, la création d'arbres de distribution supplémentaires permet l'utilisation de multiples chemins lors du routage, et permet d'avoir une racine d'arbre plus proche pour chaque RBridge. Cette proximité avec la racine d'un arbre de distribution améliore l'efficacité de la livraison de message à destinations multiples.

Comme un RBridge peut acquérir plusieurs nicknames, il peut donc être la racine pour plusieurs arbres de distribution. Le numéro de l'arbre étant, de surcroît, utilisé comme séparateur pour les chemins de coûts égaux, il s'avère qu'il y ait, même si les arbres ont la même racine, de grandes probabilités pour qu'ils n'utilisent pas les mêmes chemins.

Les RBridges doivent donc calculer tous les arbres qui pourraient être utilisés ainsi que mémoriser les états nécessaires aux filtres de vérification pour les chemins de retour (Reverse Path Forwarding Check filters). De plus, le numéro de l'arbre est utilisé comme élément de décision. Donc, tous les RBridges doivent connaître :

— Le nombre d'arbres à calculer.

— Quels sont les arbres qu'ils doivent calculer.

— Le numéro de chaque arbre.

— Savoir quels arbres vont utiliser les différents Ingress RBridges pour construire les filtres de vérification des chemins de retour.

Le nombre total d'arbres K dans un campus correspond, soit au nombre S d'arbres souhaités par le RBridge RB1 ayant le nickname le plus prioritaire, soit au nombre maximal d'arbres G gérés par le RBridge qui en gère le moins au sein du campus. Si S est supérieur à G alors K=G sinon K=S. Pour annoncer son nombre souhaité, le RBridge RB1 utilise son LSP dans lequel il indique la valeur S. Si jamais RB1 n'annonce pas S alors les K arbres suivants dans l'ordre de priorité sont calculés. Entre autres, si S est inférieur à K, alors les S arbres sont calculés et on complète avec les K-S arbres suivants toujours en respectant l'ordre de priorité des arbres.

Le calcul d'un arbre de distribution ne nécessite pas l'échange de messages supplémentaires. Comme chaque RBridge connaît la topologie complète du réseau, ils sont en capacité de calculer, en utilisant l'algorithme SPF, l'arbre ayant un RBridge précis comme racine. Néanmoins, tous les arbres calculés pour un arbre de distribution doivent utiliser les mêmes liens. Or, avec la possibilité d'avoir des liens de même coût, il faut utiliser les mêmes éléments décisionnels pour choisir les mêmes liens sans que les RBridges communiquent entre eux. Pour cela, la décision se réalise grâce à l'identifiant du parent et celui qui possède le plus petit identifiant est sélectionné.

L'utilisation des VLANs permet de réduire l'arbre de distribution aux seuls nœuds qui

appartiennent à ce VLAN permettant d'obtenir un domaine de diffusion de plus petite taille et donc une meilleure utilisation du réseau avec moins de paquets en broadcast.

Notre but étant d'interconnecter différents data centers fonctionnant avec TRILL en ne réalisant pas de fusion, nous cherchons donc à établir un réseau TRILL qui ne se limite plus à un réseau de type LAN. Or, le protocole TRILL a été conçu pour être un protocole limité au réseau LAN et restreint à la taille des réseaux LAN déjà existants. Il n'a pas été conçu pour accroître la taille des réseaux LAN et permettre un passage à l'échelle.

2.5 IS-IS

Le protocole IS-IS (Intermediate system to intermediate system) est un protocole de routage interne à états des liens (Link state routing). Il a été défini dans la norme internationale ISO/EIC 10589 :2002 de l'Open Systems Interconnection (OSI) et dans [47, 48]. C'est un Interior Gateway Protocol (IGP) , il est utilisé à l'intérieur d'un Autonomous System (AS) et permet aux routeurs d'avoir une vue complète de la topologie du réseau de l'AS. Une fois que les routeurs connaissent toute la topologie de l'AS, ils utilisent l'algorithme SPF (Shortest Path First) de Dijkstra pour calculer les plus courts chemins. Il est néanmoins possible d'utiliser ECMP (Equal-Cost Multi-Path) pour obtenir plusieurs chemins de plus courte distance.

L'avantage de IS-IS est le fait qu'il peut être utilisé au niveau de la couche 2 du réseau (MAC). La trame a donc le format suivant :

En-tête Lien de données	En-tête IS-IS	Données IS-IS

Il est également nécessaire d'identifier le type de sous réseau car IS-IS supporte les sous-réseaux de type multipoints et de type point à point, mais avec des gestions différentes. Pour le multipoint, IS-IS va réaliser l'élection d'un DIS (Designated IS) pour réguler les autres IS sur le lien.

2.5.1 DIS (Designated IS) et pseudonode

Le DIS est élu pour générer le LSP qui aurait été généré si tous les routeurs avaient été connectés à ce routeur par une topologie en étoile. Le DIS crée un pseudonode qui est le routeur central de cette topologie en étoile (Figure 2.2). Chacun des autres IS ayant pris part à cette élection va créer une interface pour établir une communication avec ce pseudonode et ainsi simuler une topologie en étoile. Cette élection se base sur deux critères. La priorité du RBridge est le premier critère et le RBridge ayant la plus haute priorité est élu. S'il y a égalité au niveau des priorités des RBridges, alors le second critère, qui est l'adresse MAC, est utilisé et le RBridge avec la plus grande adresse MAC est élu. Contrairement à Open Shortest Path First (OSPF), il n'y a ni Backup Designated Router

(BDR), ni de priorité égale à zéro pour rendre une route inutilisable. De plus l'ajout d'un nouveau Intermediate System (IS) , de type RBridge, peut causer une nouvelle élection.

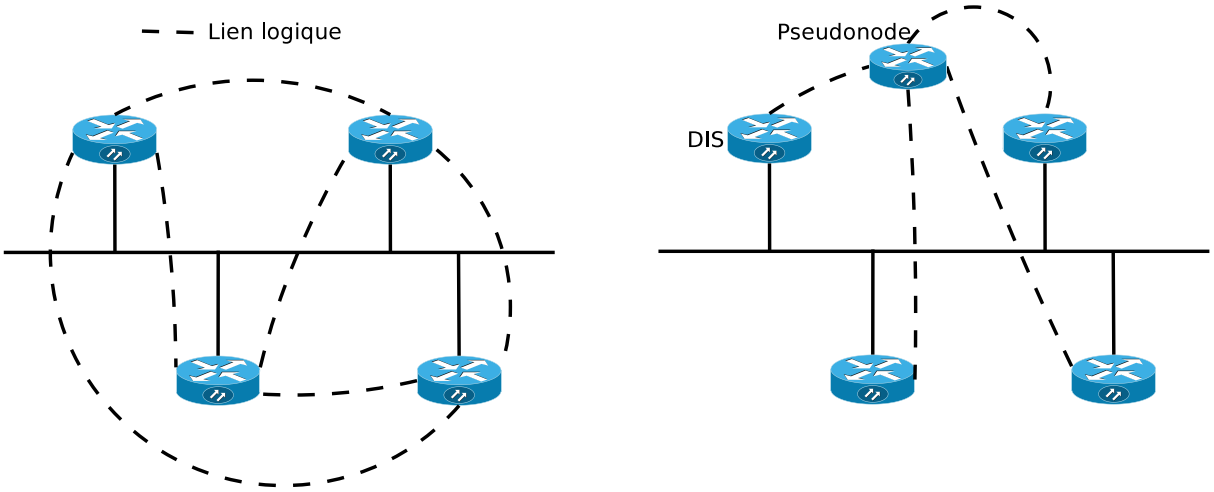


FIGURE 2.2 – Avant et après l'élection d'un DIS et la création d'un pseudonode.

2.5.2 Hiérarchie

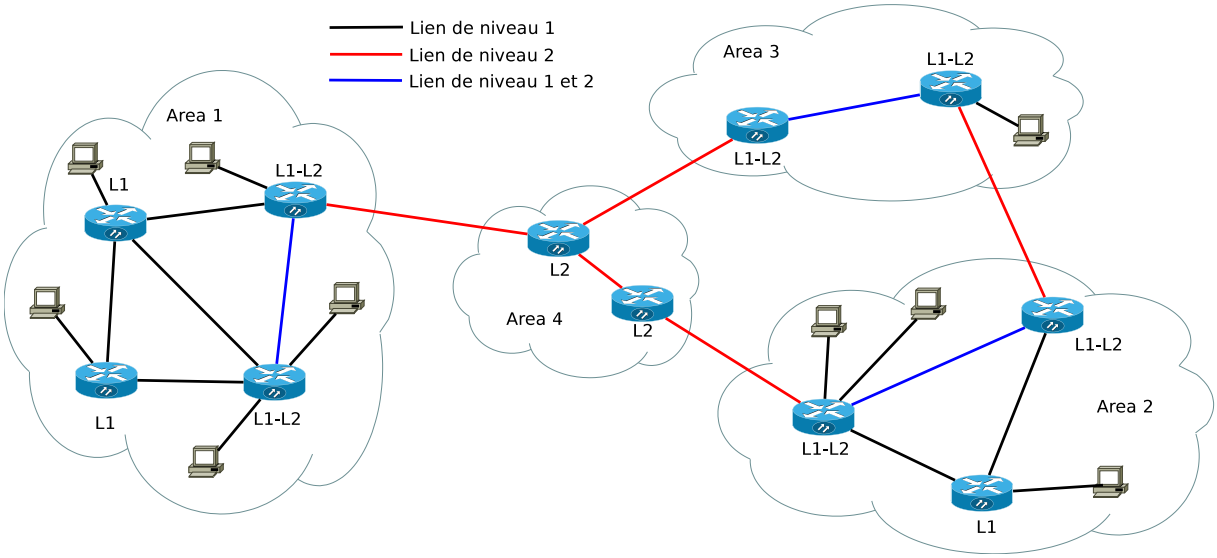


FIGURE 2.3 – Hiérarchie établie par IS-IS

Le protocole IS-IS utilise une hiérarchie qui s'établit sur deux niveaux comme présenté dans la Figure 2.3.

— Niveau 1 : Areas

— Niveau 2 : Backbone (correspond aux équipements L1-L2 et L2 ainsi qu'aux liens de niveau 2 et de niveau 1 et 2)

Du fait de ces deux niveaux, il y a une nécessité à différencier les acteurs(équipements) qui utilisent IS-IS. Pour cela, on distingue trois types d'acteurs :

1. Les machines hôtes : « End systems » (ES).
2. Les routeurs de niveau 1 : « Level 1 Intermediate Systems » (Level 1 IS).
3. Les routeurs de niveau 2 : « Level 2 Intermediate Systems » (Level 2 IS).

De plus, au sein de l'implémentation et des équipements physiques, un routeur de niveau 2 peut aussi être de niveau 1 créant ainsi un quatrième type d'acteur : les routeurs de niveau 1-2 : « Level 1-2 Intermediate Systems » (Level 1-2 IS). Ces routeurs de niveau 1-2 étant capables de fonctionner sur les deux niveaux, ils doivent donc avoir les fonctionnalités de chacun des deux niveaux.

Le routeur de niveau 1 (Level 1 IS) connaît toute la topologie du domaine au niveau 1, mais ne connaît pas la topologie des autres domaines ou du niveau 2. Il calcule l'arbre des plus courts chemins en utilisant le protocole SPF de Dijkstra. Finalement, il possède 5 bases de données :

1. L'état des liens de niveau 1.
2. Les données d'adjacence, pour connaître ses routeurs voisins.
3. La base de données Circuit, pour connaître le niveau (1 ou 2 ou les deux)
4. La table des plus courts chemins au niveau 1.
5. La table de forwarding du niveau 1 avec une table par métrique de routage.

Le routeur de niveau 2 (Level 2 IS) ne connaît que la topologie du niveau 2 et possède 2 bases de données :

1. L'état des liens de niveau 2.
2. La table des plus courts chemins au niveau 2.

Le routeur de niveau 1-2 (Level 1-2 IS) possède les mêmes fonctions que le routeur de niveau 1 et le routeur de niveau 2. Il connaît la topologie complète de l'*Area*(niveau 1) dans laquelle il se trouve et connaît aussi la topologie du *Backbone* (niveau 2) avec les routeurs de niveaux 1-2 et 2, y compris des autres domaines. Il possède donc 7 bases de données. Les 5 bases du routeur de niveau 1 ainsi que les 2 bases de niveau 2.

Comme on peut le voir sur la figure 2.3, le Backbone doit être continu sur tout le réseau. Il ne peut y avoir de nœud de transition sur le réseau Backbone qui ne soit que de niveau 1, car cela couperait le site TRILL.

2.5.3 Network Entity Title (NET)

Chaque acteur dans le protocole IS-IS utilise une adresse *Network Entity Title* (NET) détaillée dans le Tableau 2.4. C'est une adresse Network Service Access Point (NSAP) dont le N-Selector (NSEL) , l'identifiant de service, a pour valeur 0. L'identifiant du

système de niveau 1 (ES ou IS) doit être unique au sein de l'area. L'identifiant du routeur de niveau 2 doit être unique au sein du Backbone.

Initial Domaine Part (IDP)		Domain Specific Part (HDP)		
AFI Authority and Format Identifier	IDI	High Order DSP Identifiant de l'Area	System ID Identifiant d'un IS ou ES dans une Area	NSEL Identifiant de service
Taille variable entre 1 et 13 octets			6 octets	1 octet

TABLE 2.4 – Détails d'une adresse NET

2.5.4 Messages

Le protocole IS-IS utilise quatre types de paquets. Les paquets *"Hellos" (ISH, ESH, IIH)*, les *Link State Packet (LSP)*, les *Partial Sequence Number Packet (PSNP)* et les *Complete Sequence Number Packet (CSNP)*.

Pour établir les *areas* et savoir quels ISs en sont membres, les ISs échangent des paquets "Hello" pour découvrir la topologie locale et connaître leurs voisins (Figure 2.4). Le protocole IS-IS possède trois types de paquets "Hellos PDUs" différents. Les ESH (End Station Hello) sont envoyés par les ends stations et ont pour destination 09-00-2B-00-00-05 (AllIntermediateSystems) qui est l'adresse pour atteindre tous les ISs sur le lien. Les ISH (Intermediat System Hello) sont envoyés par les ISs et ont pour adresse de destination 09-00-2B-00-00-04 (AllEndSystems) pour atteindre tous les end systems sur le lien. Les IIH (Intermediate system to Intermediate system Hello) sont échangés entre les ISs. Environ toutes les 20 secondes l'échange régulier des paquets ESH et ISH dont la période peut être modifiée, permet aux ISs de découvrir leurs voisins. Cette découverte permet aux routeurs de niveau 1 ou de niveau 1-2 de remplir leurs tables d'adjacences de niveau 1 avec les adresses des ES qui sont validées en comparant la valeur de l'*area*.

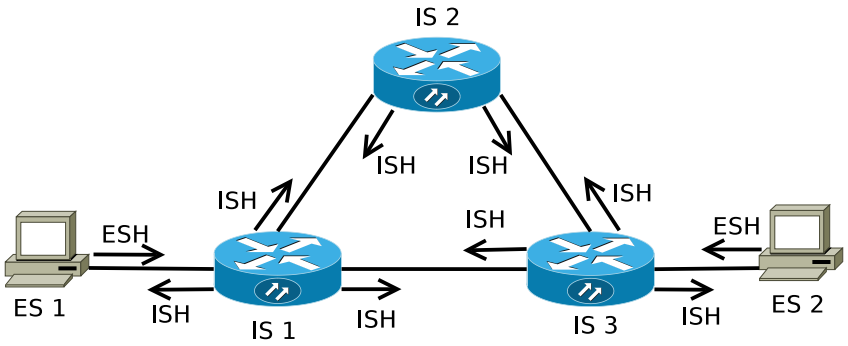


FIGURE 2.4 – Échange de messages ESH et ISH

De la même façon, les paquets ISH réceptionnés par le routeur lui indiquent quels sont les autres routeurs qu'il a comme voisins, mais ne lui donnent pas toutes les informations,

comme le niveau de l'IS, ce qui l'empêche d'établir une table d'adjacence complète. C'est pourquoi, après réception des ISHs, commencent alors des échanges de paquets IIH entre les ISs. Un IS partagera ses informations avec les autres IS concernant les ES et IS qui lui sont voisins par des paquets IIH de niveau 1. Les paquets IIH de niveau 1 ne sortent pas de l'*area* et seuls les ISs de niveau 1 et de niveau 1-2 peuvent les lire. Ces paquets sont envoyés en broadcast à l'adresse AllL1ISs (All Level 1 IS). Les paquets IIH de niveau 2 ne sont lus que par les ISs de niveau 2, du Backbone. Ils sont envoyés en broadcast à l'adresse AllL2ISs (09-00-2B-02-00-00). Ces paquets IIH permettent d'échanger les informations comme le type de circuit (le niveau) et l'identifiant du circuit local et sont échangés toutes les 10 secondes. Si aucun paquet IIH n'est reçu pendant 30 secondes, alors, le lien est considéré comme rompu. Le DIS a un comportement particulier, car il envoie les paquets IIH toutes les 3,3 secondes.

À la suite de ces messages, les ISs de niveau 1 inondent l'*area* avec des messages Link State PDU (LSP) de niveau 1 pour que tous les autres ISs du même niveau et de la même *area* connaissent la topologie complète de cette *area*. Les ISs de niveau 2 envoient leurs LSPs de niveau 2 à tous les ISs du backbone. Un LSP contient toutes les informations de l'IS qui l'émet, sa liste d'adjacence, les préfixes IP qu'il connaît, les End Systems (ES) et les adresses des areas. Un LSP est donc unique pour chaque IS. C'est pourquoi, lorsque les ISs de niveau 1 reçoivent les LSPs de niveau 1, ils connaissent la topologie de l'*area* et peuvent donc calculer les plus courts chemins en utilisant l'algorithme SPF de Dijkstra. Il en va de même pour les ISs de niveau 2 qui s'échangent des LSPs de niveau 2 et apprennent par ces LSPs la topologie du Backbone.

Les autres paquets PSNP et CSNP sont utilisés lors de l'envoi des informations des link-state databases (LSPDBs) pour synchroniser les bases de données des ISs.

Pour notre problème d'interconnexion de data centers, le principe de hiérarchie à deux niveaux, *area* et *Backbone*, permettrait d'avoir une séparation déjà établie des plans de contrôle avec chaque data center représentant une *area* et le réseau inter-data centers étant le *backbone*. Néanmoins, cette division des plans de contrôle n'est pas possible avec l'utilisation du protocole TRILL actuel, comme nous l'indiquons dans la section suivante (section 2.6).

2.6 L'utilisation du protocole ISIS au sein de TRILL

Le protocole TRILL utilise le protocole IS-IS dans son plan de contrôle pour la découverte de la topologie du réseau ainsi que le routage. Le protocole TRILL étant au niveau 2, il faut utiliser les extensions de niveau 2 pour IS-IS définies dans le RFC 6165 [49]. Ce RFC n'introduit aucune nouvelle technique de routage, mais deux nouvelles extensions utilisant le format TLV pour Type, Length, Value. Ces deux extensions sont :

1. Multi-Topology-Aware Port Capability.
2. MAC Reachability.

La première extension permet d'indiquer si un port est capable de gérer de multiples topologies. Cette extension est ajoutée aux messages "Hello IS-IS", les messages IIH.La seconde extension permet aux équipements de connaître les machines qu'ils peuvent atteindre et donc contacter.

De plus, il est nécessaire d'introduire des extensions spécifiques pour le protocole TRILL ainsi que de modifier certains comportements du protocole IS-IS. Ces modifications sont décrites dans le RFC 7176 [50]. Quatre nouvelles catégories d'extensions sont introduites et les extensions présentées dans [49] sont étendues.

La première nouvelle catégorie d'extension est *The Group Address (GADDR) TLV*. Seulement une extension est définie pour cette catégorie dans le document, elle s'appelle *The Group MAC Address Sub-TLV* et est utilisée pour indiquer les adresses MAC des machines appartenant à un groupe multicast. D'autres extensions pour la catégorie GADDR TLV sont possibles comme *The Group IP Address Sub-TLV* qui indique les adresses IP des membres d'un groupe multicast.

La seconde catégorie d'extension, *Sub-TLVs for the Router Capability TLV* comporte des Sub-TLV indiquant les capacités du RBridge. Dans cette catégorie de TLV, sept Sub-TLV sont définis.

1. *The TRILL Version Sub-TLV* est utilisé pour indiquer la version maximale du protocole TRILL utilisable. Ceci indique que le RBridge ayant émis la *TRILL version Sub-TLV* peut utiliser toutes les versions de TRILL jusqu'à celle qu'il a annoncée dans la Sub-TLV.
2. *The Nickname Sub-TLV* transporte les informations concernant les nicknames comme le ou les nicknames sélectionnés par le RBridge ayant émis ce message ainsi que leurs priorités pour les garder. Les priorités sont utilisées comme précisé dans [11].
3. *The Tree Sub-TLV* utilisé par tous les IS utilisant TRILL annonce trois valeurs nécessaires au calcul des arbres. Ces valeurs sont définies dans le RFC 6325 [11].
4. *The tree Identifiers Sub-TLV* correspond à une liste ordonnée de nicknames qui, si elle émane de l'IS avec la plus haute autorité (le *tree root*), liste les arbres de distribution que les autres IS doivent calculer.
5. *The Trees Used Identifiers Sub-TLV* a le même format que la Sub-TLV précédente (*The Tree Identifiers Sub-TLV*), mais contient les arbres de distribution que l'IS, ayant émis la Sub-TLV, souhaite utiliser.
6. *Interested VLANs and Spanning Tree Roots Sub-TLV* contient une plage de VLAN ainsi que les informations communes à tous les VLANS dans cette plage.

7. *The VLAN Group Sub-TLV* contient des identifiants de VLAN pour indiquer que l'apprentissage des VLANs s'effectue correctement entre les IS qui les annoncent.

La troisième TLV est la *MTU Sub-TLV of the Extended Reachability TLV* qui est utilisée pour annoncer la MTU d'un lien.

La quatrième et dernière nouvelle catégorie de TLV est la *TRILL Neighbor TLV*. Cette TLV est utilisée dans les messages "hello" entre les IS (IIH PDUs) à la place de l'*IS Neighbor TLV* pour découvrir les voisins et créer les états d'adjacence qui permettent à chaque nœud de connaître son voisinage.

Les modifications des extensions introduites dans le RFC 6165 [49] impliquent la création de trois Sub-TLVs pour l'extension *Multi-Topology-Aware Port Capability Sub-TLVs*. *The Special VLANs and Flags Sub-TLV* est transportée dans tous les IIH PDUs. *The Enabled-VLANs Sub-TLV* indique quel est le VLAN activé pour le service des End station au niveau du port de l'IS ayant émis le message "Hello". *The Appointed Forwarders Sub-TLV* est utilisée par le *Designated RBridge* sur un lien pour informer les IS s'ils sont ou non les diffuseurs désignés pour le VLAN-x.

En plus de ces ajouts de Sub-TLV, certains comportements du protocole IS-IS sont modifiés. Les messages IIH ne comportent plus de *IS-IS Neighbor TLV*, mais une *TRILL Neighbor TLV* à la place. La notion d'*area* est supprimée dans TRILL, car tous les messages TRILL doivent avoir une adresse d'*area* égale à zéro. Cette suppression de la notion d'*area* nous pose problème puisqu'elle implique que, lors de l'interconnexion de sites TRILL, il n'y ait plus qu'une seule *area* et donc une fusion des sites TRILL en un seul site. Cette fusion crée des problèmes de gestion ainsi que de collisions des nicknames qui doivent être résolus sur l'ensemble de la nouvelle *area*.

2.7 VNT

Au sein des data centers de la société Gandi [51], un test de fonctionnement est réalisé avec un TRILL particulier permettant une meilleure isolation des clients, appelé Virtual Network over TRILL (VNT) [43, 14]. VNT est une technique de réseau overlay basée sur TRILL qui utilise des options ajoutées à l'en-tête TRILL. Ces options sont détaillées dans la section suivante (section 2.7.1).

2.7.1 En-tête VNT

L'en-tête VNT est un en-tête TRILL standard auquel on ajoute une option propre à VNT. Dans [52] on nous indique le format que l'on doit utiliser pour définir de nouvelles options. Ce format, de type TLV, doit être structuré suivant l'exemple de la Figure 2.5.

Le premier champ APP indique à quel équipement s'adresse cette option.

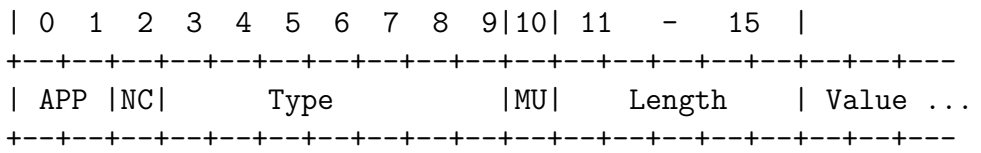


FIGURE 2.5 – Structure TLV pour une extension

— La valeur 0 indique que l'option doit être traitée à chaque saut sur le chemin. Tous les RBridges sur le chemin doivent donc gérer cette option.

— La valeur 3 indique que l'option n'est traitée que de bout en bout, entre Ingress et Egress RBridge.

— Les valeurs 1 et 2 sont réservées pour de futures utilisations.

Le champ NC indique si l'extension est critique (valeur de 0) ou non (valeur 1). Le champ MU vaut 0 si l'extension est immuable et sinon 1.

En suivant les obligations du format TLV, l'en-tête de VNT se compose de l'en-tête TRILL (de 64 bits) suivi de deux champs de 18 bits pour l'identifiant de flow et d'un champ de 64 bits pour l'extension VNT.

L'extension VNT se compose d'un champ Virtual Network Identifier (VNI) de 24 bits qui contient un identifiant unique dans tout le réseau VNT. L'utilisation du champ VNI dans l'en-tête VNT permet d'avoir la même fonctionnalité de restriction du domaine de diffusion qu'avec les VLANs. Néanmoins, le champ VNI permet d'avoir beaucoup plus de valeurs (environ 16 millions), et donc de domaines de diffusion, qu'en utilisant des VLANs (4094).

L'en-tête complet nécessaire pour VNT est présenté dans la Figure 2.6 ci-dessous.

2.7.2 Modifications apportées au protocole TRILL

Comme vu dans la section précédente (2.7.1), le protocole VNT est basé sur le protocole TRILL auquel on a ajouté une extension spécifique contenant un champ VNI. Il en résulte qu'aucune modification n'a été apportée au plan de contrôle du protocole TRILL. Seul le plan de données a été modifié avec l'ajout de l'extension VNT à l'en-tête TRILL. Néanmoins, il faut apporter des modifications aux RBridges pour pouvoir gérer ce champ VNI tant au niveau de l'envoi et la réception des messages qu'au niveau de l'association de la valeur du VNI avec les bonnes machines virtuelles.

L'ajout d'un nouvel élément réseau appelé Virtual Switch (VS) permet au RBridge de déterminer le tag VNI d'une VM plus facilement. Lors de la création d'une VM, les interfaces de cette VM sont connues et sont connectées au VS local qui gère toutes les VMs locales membres d'un VNI. Cette connexion au VS permet d'avoir toutes les VMs locales membres du même VNI accessible facilement lors de l'inondation d'un message

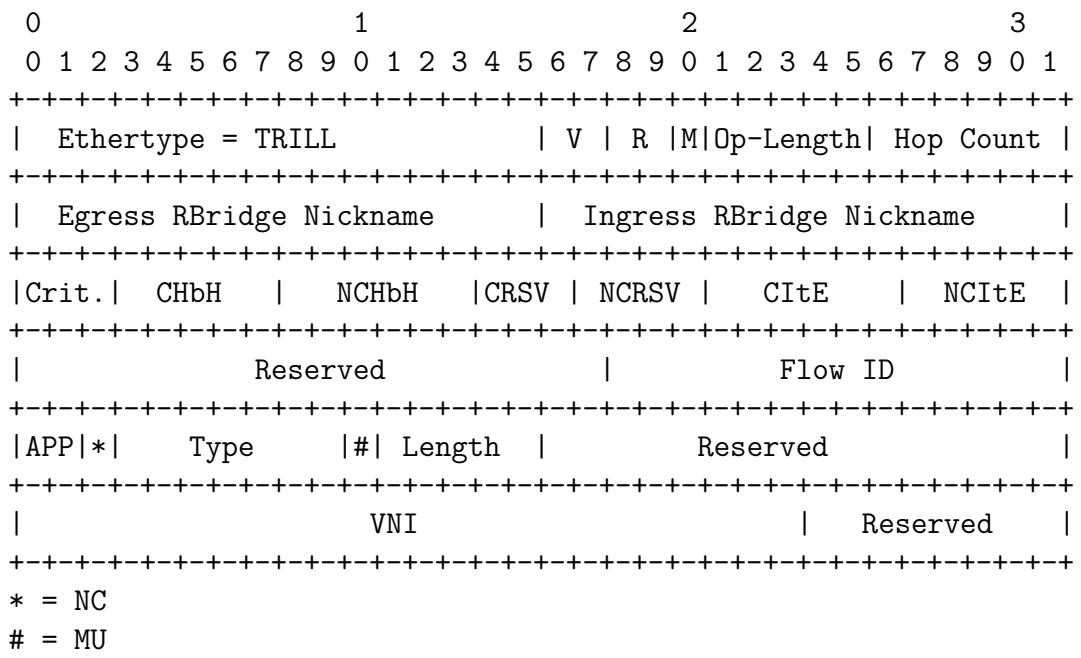


FIGURE 2.6 – VNT Header

puisqu'elles sont toutes connectées à ce VS. Le RBridge détermine donc le tag VNI d'une VM en se basant sur le VS qui lui transmet le paquet.

2.7.2.1 Modification des RBridges

2.7.2.1.1 Gestion de l'envoi

Pour l'envoi d'un paquet VNT, le processus d'envoi hérité de TRILL est modifié pour améliorer l'isolation des flux des clients à l'aide des champs VNIs. Plus précisément, il y a deux modifications dans le processus d'envoi de TRILL, ces deux modifications sont :

1. Lors de l'inondation d'un paquet, le RBridge va envoyer le paquet seulement au VS qui gère le VNI du paquet. Le VS va ensuite inonder le paquet vers toutes les VMs qui lui sont rattachées et qui partagent le même VNI.
2. Le RBridge doit aussi ajouter l'en-tête VNT avant d'envoyer le paquet. Cet en-tête VNT est ajouté à tous les paquets de données des clients, qu'ils soient de type unicast ou multicast.

2.7.2.1.2 Gestion de la réception

Pour réaliser l'isolation des flux des clients, VNT ajoute un champ VNI lors de l'envoi de ces paquets. Néanmoins, si, lors de la réception on ne vérifie pas la valeur de ce champ, on ne peut pas garantir l'isolation des flux. C'est pour cela que VNT modifie le processus de réception des paquets. Ces modifications sont identiques, que le paquet soit multicast ou unicast. Lors de la réception du message, le RBridge doit vérifier s'il supporte le VNI du paquet. Si oui, alors il le transfère au VS, sinon il rejette le paquet. À l'aide de cette

vérification du champ VNI, VNT garantit une isolation des flux de chaque client dans le réseau du data center.

2.8 Conclusion

Un avantage de VNT est de permettre au client de configurer ses équipements virtuels avec les adresses MAC ou IP qu'il souhaite sans restriction sur ces adresses. Ces adresses n'étant pas visibles dans le réseau cœur, elles ne peuvent pas affecter le routage des messages au niveau du réseau cœur. Le processus diffère au sein du réseau virtuel du client, car le routage du paquet se fait en deux temps. Dans un premier temps, le champ VNI est utilisé pour que le paquet reste dans le réseau virtuel du client et donc soit isolé des autres clients du data center. Grâce à ce champ VNI le paquet va être diffusé au sein d'un arbre de distribution, créé avec le protocole IS-IS, ce qui va restreindre sa diffusion aux RBridges de cet arbre qui hébergent l'ensemble des VMs du VNI. Dans un second temps, les adresses IP, ou MAC, du client, sont utilisées pour router le paquet à l'intérieur du réseau virtuel du client. S'il y a une erreur d'adressage de la part du client, alors, c'est à ce moment que des problèmes apparaissent. Cependant, ces problèmes restent restreints au réseau de cet utilisateur et ne perturbent pas le routage des paquets des autres clients.

VNT est utilisé dans les data centers de la société Gandi et il présente des performances intéressantes que la société souhaiterait améliorer et étendre à l'ensemble de ses data centers. C'est pourquoi mon sujet de recherche se situe dans le domaine de l'interconnexion des data centers et plus précisément des data centers fonctionnant avec VNT et donc TRILL.

Chapitre 3

Interconnexion des data centers

Sommaire

3.1 Introduction

Avec l'augmentation de l'utilisation du cloud computing, deux solutions d'expansion sont possibles. La première consiste à créer de nouveaux data centers pour répondre à la demande et amener au plus près des clients ces data centers afin de fournir de meilleures performances. La seconde vise à augmenter la taille des data centers déjà existants pour regrouper encore plus de ressources et donc de clients dans un seul data center. Quel que soit la solution qui sera choisie, l'interconnexion des data centers va se compliquer. Dans le premier cas, il y aura plus de data centers qui vont communiquer, alors que dans le second cas, bien qu'il y aura moins de data centers, les communications entre eux vont augmenter.

De plus, un data center traditionnel doit être interconnecté avec un autre data center appelé "Twin Data Center" qui est assez proche du premier, de l'ordre de la dizaine de kilomètres. Ce "Twin Data Center" permet la continuité de services sans interruption ainsi que la reprise rapide des services lors de l'interruption d'activité d'un site. Le Plan de Reprise d'Activité utilise le second data center comme sauvegarde du premier, le second prenant la relève du premier pendant que celui-ci est inactif. Dans ces deux cas, l'interconnexion entre les deux data centers sert uniquement à transférer les données d'un site à l'autre pour avoir un data center de sauvegarde.

Par opposition, le cloud computing utilise des data centers mais de façon dynamique en répartissant des tâches sur différents data centers et surtout en présentant les ressources disponibles sans préciser leurs positions.

L'expansion des data centers, l'adoption du cloud computing et de service à la demande, ainsi que la consommation importante de vidéo, imposent de plus en plus la nécessité d'augmenter les ressources tant à l'intérieur des data centers qu'en quantité de data centers. Néanmoins, il est aussi nécessaire de concevoir un système d'interconnexion permettant à ces data centers de communiquer entre eux. L'interconnexion des data centers permet, entre autre, de supporter des catastrophes naturelles en transférant le ou les services sur un autre data center. C'est également une nécessité pour établir la continuité d'un réseau d'une entreprise. Pour que les informations puissent être échangées, dans le cas d'une société travaillant dans plusieurs pays distants, les data centers doivent être interconnectés car chaque succursale se connecte au data center le plus proche. L'interconnexion des data centers permet, lors d'une maintenance, d'une mise à jour, ou d'une panne d'un data center, de pouvoir transférer les services sur un autre data center afin de préserver les clients des inconvénients résultants de ces événements.

3.2 Problématique

Notre problème réside dans la présence de plusieurs data centers qui fonctionnent en utilisant TRILL mais qui ne sont pas connectés entre eux. Nous cherchons une méthode qui permettrait d'obtenir cette interconnexion sans avoir l'obligation de fusionner tous les sites TRILL pour n'en former plus qu'un, comme indiqué dans le RFC6325 [11] de TRILL. Plus précisément, nous souhaiterions avoir des data centers fonctionnant avec TRILL et qui soient les plus indépendants possible, tant au niveau du plan de contrôle, qu'au niveau des nicknames. De cette façon, on résoudrait le problème du nombre limité de nicknames et on maintiendrait le niveau de complexité de gestion des data centers tout en évitant de surcharger le réseau public, et donc payant, avec des messages de contrôle. De plus, nous n'avons pas accès au réseau public et le choix de la technologie utilisée pour le VPN n'est pas défini. Ce point de décision est commun avec la deuxième partie de la thèse qui est l'établissement d'un cloud hybride entre le data center public, fonctionnant avec TRILL,

et le réseau privé d'un client. En effet, le client peut avoir configuré son réseau en utilisant diverses technologies VPN comme VXLAN, NVGRE ou STT, pour n'en citer que trois. Il faut donc que l'on puisse établir une passerelle entre ces VPN et le data center TRILL. Pour réaliser l'interconnexion des data centers TRILL nous allons nous intéresser aux différentes solutions proposées puis nous présenterons notre solution ainsi que ses performances.

3.3 Les solutions proposées pour interconnecter des data centers TRILL.

3.3.1 TRILL Data Center Interconnect

Les auteurs de [53] proposent une solution pour interconnecter les data centers TRILL via des réseaux WAN. Leur solution a pour but d'étendre les sites TRILL sans avoir à révéler les informations qu'ils contiennent comme les adresses MAC des clients des réseaux LAN ou les adresses MAC des fournisseurs. Pour cela ils proposent d'établir des connexions entre les différents sites TRILL via l'extension des protocoles de routage permettant un échange minimal d'information. Leur solution permet donc de conserver des sites TRILL indépendants et d'éviter d'avoir à fusionner ces sites en un grand site TRILL où tous les nicknames doivent être uniques.

Dans le draft, les sites TRILL sont appelés campus TRILL ou domaine L1 IS-IS. Ces sites doivent tous posséder un RBridge de bordure désigné. Ils ont pour tâche de s'échanger les informations permettant l'établissement des connections entre les différents sites TRILL. Néanmoins, l'établissement de la connexion nécessite la création préalable des routes entre les différents RBridge et, pour cela, une extension du protocole BGP est proposée. Cette extension est une nouvelle annonce de route BGP et elle est présentée dans la Table 3.1.

RD (8 octets)
Adresse MAC du RBridge Source
Adresse IP(v4/v6) du RBridge Source
Taille du Nickname (1 octet)
Nickname du RBridge (2 octets)
Label MPLS (n*3 octets)

Table 3.1 – Extension BGP

Les sites TRILL ne communiquent entre eux que via les RBridges de bordure. Chaque RBridge de bordure connaît tous les autres RBridges de bordure. Cet échange se fait à l'aide de l'extension *Affinity TLV* dont l'utilisation est détaillée dans [54]. Comme chaque RBridge de bordure connaît les adresses des autres RBridge de bordure, il est possible

de créer un chemin VPN spécifique dans le réseau WAN. Une fois ces chemins créés, les RBridges de bordure vont s'échanger trois types de nicknames :

1. Les nicknames des RBridges de bordure.
2. Les nicknames des RBridges de chaque campus.
3. Les nicknames des RBridges participant à des VLANs de client ou des VPNs.

Avec toutes ces informations, les données n'ont plus à être broadcastées à tous les RBridges de bordure mais seulement à être envoyées aux RBridges de bordure concernés. Néanmoins, cette quantité d'informations implique que les routeurs de bordure doivent avoir des ressources importantes. Si l'on réserve 1000 nicknames pour les RBridges de bordure cela permet d'avoir $2^{16} - 1000 = 64536$ RBridges dans chaque site. Il faut donc qu'un RBridge de bordure maintienne en mémoire les 999 nicknames des autres RBridges de bordure ainsi que potentiellement $999 \times 64536 = 64.471.464$ nicknames pour l'ensemble des RBridges de tous les sites.

Cette solution crée un point de congestion avec le RBridge de bordure. Cet équipement doit stocker une grande quantité d'informations (64 472 463 nicknames plus les différents VLANs et VPNs des clients), participer à la création des routes et des connexions entre les sites TRILL puis ensuite traiter tous les paquets à destination non locale du site dans lequel il se trouve. C'est pourquoi, en plus de son utilisation de BGP, nous n'avons pas retenu cette solution.

3.3.2 Connecting Disparate TRILL-based Data Center/PBB/Campus sites using BGP

Le problème de l'interconnexion de site TRILL est sous divisé en trois cas dans [55]. Le premier cas est celui de l'interconnexion de data centers TRILL. Le second est l'interconnexion de réseaux cœur de fournisseur en TRILL et le dernier cas est celui de l'interconnexion de campus TRILL, autre que data center. Le premier cas nous intéressant plus particulièrement sera donc celui que nous aborderons. La Figure 3.1 ci-après représente l'architecture d'une interconnexion entre deux data centers TRILL proposée dans le document.

U-PE : *User-near PE device*. Les U-PEs sont les équipements en bordure au sein d'un site client ou d'un site tier-2. Cet équipement possède une instance VRF pour chaque client auquel il est connecté.

N-PE : *Network Transport PE device*. Cet équipement possède les caractéristiques et fonctionnalités d'un RBridge sur ses interfaces orientées vers le data center TRILL. De plus, il possède une instance VRF pour chaque site TRILL qu'il connecte. Sur les interfaces orientées vers le réseau IP core, c'est un équipement de niveau 3 supportant l'utilisation de IP+GRE ou IP+MPLS. Il s'occupe aussi d'annoncer les area nicknames à

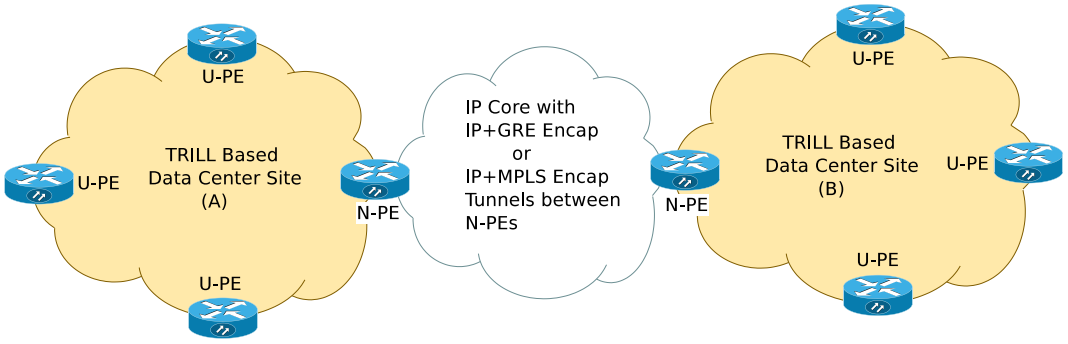


FIGURE 3.1 – Connexion entre data centers via un réseau IP.

ses pairs N-PEs via des annonces de routes BGP-MAC-VPN VRF qui sont enregistrées suivant le format présenté dans la Table 3.2.

Au niveau du plan de contrôle, les U-PEs correspondants aux Top of Rack Switches, doivent avoir des fonctions leur permettant d'apprendre les adresses MAC sources qu'ils voient passer. Les N-PEs doivent aussi apprendre les adresses MAC et le nickname de l'U-PE correspondant via l'ARP snooping. Un N-PE possède ainsi une instance VRF par client dans laquelle sont enregistrées toutes les correspondances adresse MAC/U-PE nickname.

MAC adresse	U-PE Nickname
00 :be :ab :ce :fg :9f (local)	10254
...	...

TABLE 3.2 – Format de la BGP-MAC-VPN VRF dans un N-PE.

En résumé, les N-PEs sont des RBridges de bordure qui doivent enregistrer les adresses des autres RBridge de bordure pour pouvoir établir la connexion inter data centers ainsi qu'enregistrer les adresses MAC et les nicknames des RBridges ayant des communications inter data centers. Le N-PE s'occupe ainsi d'encapsuler et désencapsuler les paquets ainsi que modifier les en-têtes des trames TRILL pour pouvoir les envoyer dans le site TRILL de destination.

Le draft [56] propose d'enrichir [55] en y ajoutant du trafic engineering. Néanmoins cette proposition utilise des nicknames pour identifier des chemins or ces nicknames faisant partie du niveau 2 doivent être uniques dans tous le réseau TRILL. Cette extension utilise encore plus de nicknames uniques et, de ce fait, réduit la quantité de nicknames restants au sein du data center.

3.3.3 TRILL-EVPN

La solution proposée dans [57] au problème de l'interconnexion de site TRILL est l'utilisation de différents VPN Ethernet (E-VPN) sur des réseaux MPLS/IP. Cette solution

permet la transparence des adresses MAC client au point de distribution, au nœud MES (MPLS Edge Switch), ainsi que l'isolation des plans de contrôle de chaque site TRILL. Elle utilise le protocole de routage IS-IS dans les sites TRILL y compris dans les MES puis le protocole BGP dans le réseau MPLS, comme représenté dans la Figure 3.2.

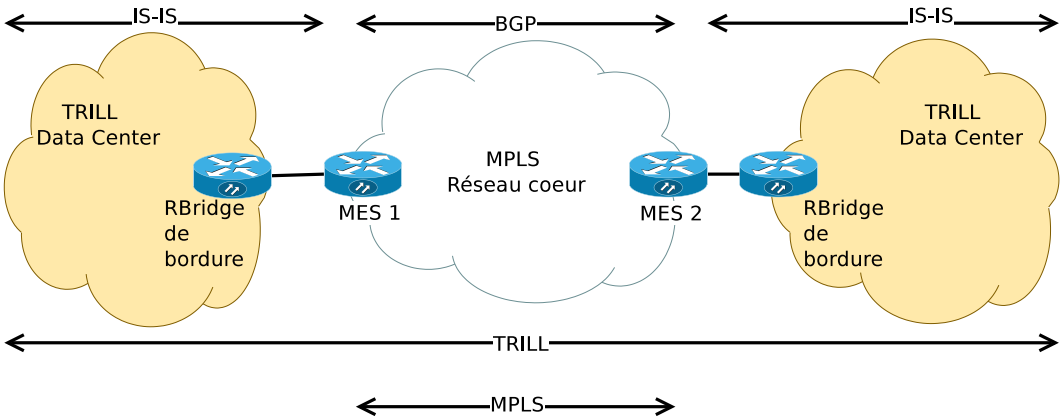


FIGURE 3.2 – Connexion entre data centers via TRILL-EVPN.

Il est nécessaire que le MES participe à l'échange d'informations du protocole IS-IS dans le plan de contrôle du site TRILL auquel il est rattaché et ne retire pas l'en-tête TRILL des paquets de données. Le MES apparaît donc comme un RBridge de bordure au niveau du plan de contrôle mais comme un RBridge cœur pour le plan de données. Sa tâche consiste à encapsuler un paquet TRILL dans un paquet MPLS et de décapsuler un paquet MPLS pour avoir un paquet TRILL. Bien que le MES serve de terminaison du protocole IS-IS sur le plan de contrôle, comme il est transparent pour le plan data, les nicknames ne sont pas restreints à chaque site TRILL. Il est donc nécessaire d'avoir des nicknames uniques au sein de chaque site. Dans ce but, les auteurs proposent un nickname hiérarchique comprenant deux informations.

— L'identifiant du site.

— L'identifiant du RBridge.

La taille de chaque partie est laissée au choix de l'administrateur tout en sachant que la taille limite est de 16 bits pour un nickname.

Dans le réseau MPLS, entre les différents MES, il est nécessaire de définir un nouveau message pour annoncer les routes pour atteindre les différents RBridges. Le format de ce message est présenté dans la table 3.3.

Cette solution implique d'avoir accès au MES du fournisseur internet ce qui n'est pas forcement possible. Elle oblige aussi à avoir des nicknames uniques dans tout le réseau TRILL ce qui limite le nombre de RBridges et d'arbres à 2^{16} et donc ne permet pas le passage à l'échelle. Cette approche ne correspond pas à ce que nous recherchons.

RD (8 octets)
Identifiant du segment Ethernet (10 octets)
Ethernet Tag ID (4 octets)
Taille du Nickname (1 octet)
Nickname du RBridge (2 octets)
Label MPLS (1 octets)

TABLE 3.3 – Annonce des routes pour les Nicknames TRILL

3.3.4 Default Nickname Based Approach for Multilevel TRILL

L'approche présentée dans [54] se base sur l'utilisation de routes par défaut pour interconnecter différents sites TRILL et sur une nouvelle méthode pour la construction des arbres à destinations multiples à l'aide de nicknames partiels.

Les différents sites TRILL correspondent à des areas IS-IS de niveau L1. Ils sont tous connectés à une area IS-IS de niveau L2 qui correspond au réseau cœur. Dans chaque site TRILL est présent au moins un RBridge de bordure qui annonce sa capacité à joindre d'autres sites via le bit "Attach bit" du IS-IS Link State PDU. Si un message est destiné à une machine dans une area différente, alors il suivra la route par défaut jusqu'au RBridge de bordure. Pour les messages à destinations multiples et sur différents sites, ils sont distribués selon un arbre de diffusion qui sera divisé en plusieurs parties. Une partie est l'arbre de diffusion propre au réseau cœur de niveau L2, les autres parties étant les arbres de diffusion dans chaque site TRILL de niveau L1. Néanmoins, bien que ces arbres soient dans des sites différents, ils partagent tous le même nickname, ce qui permet d'éviter son changement à chaque RBridge de bordure.

Pour une trame unicast, la trame est envoyée normalement dans le site local. Le RBridge de bordure récupère la trame si elle lui est destinée ou si elle est diffusée sur le réseau. Comme celui-ci possède les ensembles de nicknames de chaque site TRILL (exemple : site 1 : 100-200, site 2 : 201-500, etc...),il peut donc envoyer la trame à la bonne destination.

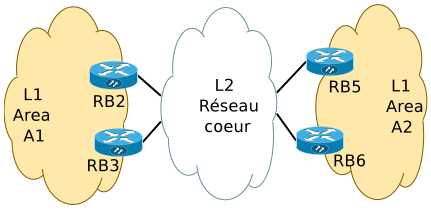


FIGURE 3.3 – Réseau physique.

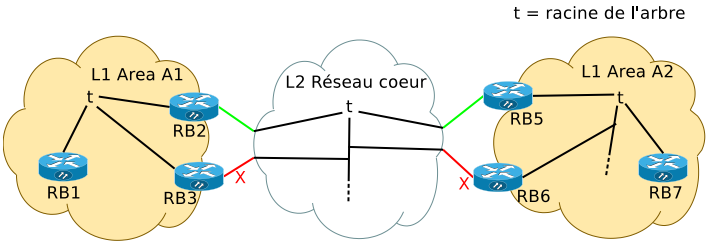


FIGURE 3.4 – Réseau logique avec TRILL Multilevel

Comme il est possible d'avoir plusieurs RBridges de bordure (voir Figure 3.3), il est nécessaire d'utiliser la technique des *Affinity TLV* pour en désigner un qui sera connecté à l'arbre correspondant dans le réseau cœur. La Figure 3.4 montre quels sont les RBridges

de bordure qui permettent la connexion avec les arbres de diffusion. Le RBridge RB2 est le RBridge de bordure désigné, grâce à la technique de l'*Affinity TLV*, pour transiter les messages entre t (la racine de l'arbre de diffusion) de l'area A1 et le réseau cœur L2. Le RBridge RB3 a dû désactiver son lien avec le réseau cœur pour les messages de diffusion de l'arbre t. Il en est de même pour l'Area A2 ou RB5 est le RBridge désigné pour faire transiter le trafic. Cependant, alors que chaque arbre de diffusion possède un RBridge de bordure désigné, un RBridge de bordure peut être désigné pour plusieurs ou aucun arbre de diffusion.

L'inconvénient de cette solution est qu'il faille avoir des nicknames uniques dans tous les sites TRILL vu que chaque RBridge de bordure doit être en mesure de router le message. De plus, l'attribution des nicknames est imposée par leur gestion. En effet, un site doit avoir une suite de nicknames forcément contiguës pour réduire la taille des enregistrements dans les RBridges de bordure. Étant donnée leur obligation d'unicité, on retrouve le problème du manque de nicknames. À cela s'ajoute le fait que tous les RBridges de bordure doivent connaître, au minimum, tous les nicknames seuils de chaque site TRILL en plus des nicknames des RBridges de bordure. Si jamais un site venait à manquer de nicknames, résultant d'une réservation initiale trop faible, alors les nouveaux RBridges devraient utiliser un nickname non encore attribué dans tout le réseau. Ceci implique que les RBridges de bordure doivent aussi enregistrer les nicknames attribués hors des plages réservées pour chaque site TRILL. Pour savoir quels sont les nicknames attribués de façon dynamique, il faut donc les annoncer à tous les RBridges de bordure. Il est donc défini, dans le draft, une nouvelle extension de type TLV pour la gestion des nicknames avec le format présenté dans le Tableau 3.4.

En conclusion, cette solution implique d'avoir des nicknames uniques, des RBridges de bordure qui enregistrent toutes les plages locales et dynamiques de nicknames et de créer des messages pour gérer ces plages de nicknames. Elle ne correspond donc pas à ce que nous souhaitions obtenir et n'a pas été retenue.

3.3.5 Flexible Multilevel TRILL

Dans [58] les auteurs commencent par définir deux approches différentes pour interconnecter des data centers TRILL. La première approche, qu'ils appellent l'approche du nickname unique (the unique nickname), consiste à utiliser des nicknames uniques dans tout le réseau TRILL qui s'étend à tous les data centers. Ceci implique une limite de 2^{16} RBridge dans tout le réseau car il n'y a que 2^{16} nicknames possibles. Cette approche consiste à fusionner les réseaux des data centers en un seul réseau qui les regroupe tous, d'où la nécessité d'avoir des nicknames uniques. Deux techniques sont possibles dans cette approche. La première impose aux RBridges de bordure d'annoncer les nicknames disponibles aussi bien à l'intérieur du data center qu'à l'extérieur de celui-ci. Les RBridges de bordure doivent donc maintenir, gérer et annoncer les nicknames non utilisés. La seconde

Type=NICK-MGMT (1 octet)
Longueur (1 octet)
Longueur Area-ID (1 octet)
Identifiant IS-IS de l'Area (1 à 13 octets)
Nombre de plages local de nicknames (1 octet)
Premier nickname de la plage locale 1 (2 octets)
Dernier nickname de la plage locale 1 (2 octets)
. . .
Premier nickname de la plage locale n (2 octets)
Dernier nickname de la plage locale n (2 octets)
nombre de plages dynamiques (1 octet)
Premier nickname de la plage dynamique 1 (2 octets)
Dernier nickname de la plage dynamique 1 (2 octets)
. . .
Premier nickname de la plage dynamique n (2 octets)
Dernier nickname de la plage dynamique n (2 octets)

TABLE 3.4 – Structure de l'option TLV pour la gestion des Nicknames

technique consiste à diviser le nickname en deux avec une partie contenant l'identifiant du data center et l'autre partie l'identifiant du RBridge au sein de ce data center. Ici, la répartition de la division est laissée au choix de l'administrateur mais réduit néanmoins le nombre de RBridges disponibles au sein d'un data center. De plus, si la division a été faite en sous évaluant, soit le nombre de RBridges au sein d'un data center, soit le nombre total de data centers, alors la modification de ce nickname devient très complexe puisqu'il faut le modifier dans tous les data centers.

La seconde approche, l'agrégation des nicknames (aggregated nickname), permet de réutiliser des nicknames dans les différents data centers. Seulement, les nicknames des RBridges de bordure ainsi que de certains éléments propres à chaque data center, différents selon la technique utilisée, doivent être uniques dans tous le réseau TRILL. Cette approche permet d'éviter la fusion des différents sites TRILL et de conserver des sites de moindre taille avec une gestion propre et indépendante des autres sites. Néanmoins pour les communications inter data centers, les RBridges de bordure doivent réécrire les en-têtes des paquets pour que ceux-ci soient dirigés vers l'autre data center et la bonne machine au sein de ce data center. Cette réécriture implique que les RBridges de bordure enregistrent diverses informations telles que les adresses MAC et les VLANs pour savoir qui communique.

Une autre solution est d'ajouter deux champs dans l'en-tête TRILL, le champ "*in-*

gress swap nickname" et le champ "*egress swap nickname*". Ces adresses sont échangées à chaque RBridge de bordure mais ce sont les machines qui communiquent qui doivent stocker ces informations.

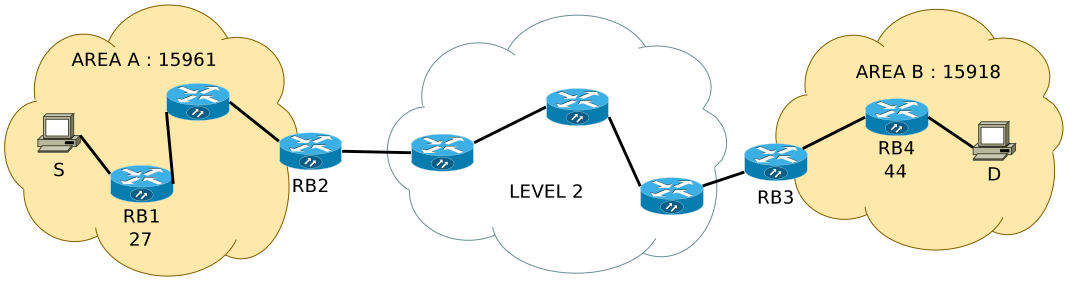


FIGURE 3.5 – Exemple de topologie inter data centers

Ci-après, on présente un exemple d'utilisation des swaps fields en utilisant la topologie introduite dans la Figure 3.5). On suppose que RB1 sait que D est dans l'Area B avec le nickname 15918 et que RB3 sait que D est attaché à RB4 qui possède le nickname 44. L'utilisation des champs "*ingress swap nickname*" et "*egress swap nickname*" se fait de la façon suivante :

— RB1 encapsule le paquet de S destiné à D avec comme *egress nickname* le nickname de l'area B à savoir 15918. Dans le champ "*egress swap nickname*" RB1 met le nickname de RB4 (44).

— RB2 copie le *ingress nickname* du paquet dans le champ *ingress swap nickname* et met le nickname de l'area A (15961) dans le champ *ingress nickname*.

— RB3 échange les valeurs des champs *egress nickname* et *egress swap nickname*.

— RB4 apprend la correspondance entre l'adresse MAC et le numéro de VLAN de la source S ainsi que la paire de nicknames (*ingress nickname*, *ingress swap nickname*) lorsqu'il décapsule le message et qu'il l'envoie à D.

Après cette introduction, les auteurs présentent leur solution générale qui peut être utilisée indépendamment de l'approche choisie. Cette architecture, appelée *multilevel TRILL*, nécessite plusieurs modifications du protocole TRILL. Ci-après sont listées les sept modifications majeures apportées au protocole TRILL pour utiliser *multilevel TRILL* :

1. La configuration d'une adresse d'*area* différente de zéro ainsi que son encodage dans les PDUs d'IS-IS. Il est nécessaire d'avoir un mécanisme compatible avec les anciens RBridges qui fonctionnent seulement avec une adresse d'*area* égale à zéro.
2. Une gestion des nicknames en fonction de l'approche choisie (unique ou agrégée).
3. Annoncer les informations de *pruning* au travers des *areas*.
4. Calculer les arbres de distribution multi-areas pour les trames à destinations multiples.

5. Calculer les informations de RPF (Reverse Path Forwarding) pour ces arbres.
6. Calculer les informations concernant le pruning.
7. Concevoir ces modifications tout en restant compatible avec les anciens RBridges.

Concernant le point numéro 4, la construction des arbres *multi area*, la solution proposée est de construire un arbre dans chaque *area*. Ces arbres doivent chacun, posséder un RBridge de bordure, ce qui permet de connecter tous ces arbres via l'*area* de niveau 2 (Level 2 dans la figure 3.5). Pour sélectionner le RBridge de bordure dans chaque *area*, les auteurs proposent d'utiliser le mécanisme *affinity TLV*. La répartition de la charge s'effectue de cette façon au niveau des flux des différentes communications car chaque communication sortira et entrera dans l'*area* par le même RBridge de bordure.

Le calcul du RPF ne change pas pour ce qui est de l'arbre local. En ce qui concerne les arbres *multi area*, les RBridges de chaque area doivent savoir quel est le RBridge de bordure qui fait transiter les messages entre l'*area* local et l'*area* de niveau 2. Pour cela, il est possible de réutiliser le mécanisme *affinity TLV* pour diffuser l'information aux RBridges de l'*area* locale. Une autre possibilité serait d'avoir un DBRB (Designated Border RBridge) qui annonce les informations à tous les RBridges. La proposition des auteurs est de diviser le nickname en plusieurs champs :

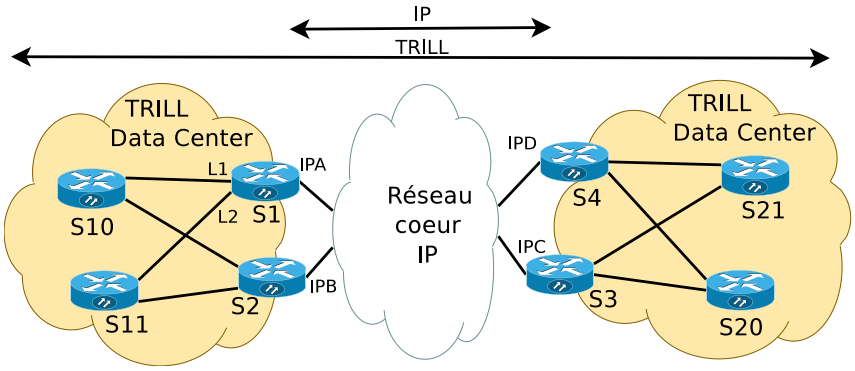
— Les 6 derniers bits pour l'identifiant de l'arbre, et donc seulement 64 arbres possibles dans une *area*.
— Les 10 autres bits seraient utilisés pour :
  — Scope : un bit indiquant si l'arbre est local ou *multi area*.
  — Border injector : un identifiant du RBridge de bordure qui a fait transiter ce paquet entre l'*area* de niveau 2 et l'*area* locale.

En conclusion, ce draft résume et propose des approches pour étendre le protocole TRILL avec du multilevel ce qui permettrait d'interconnecter différents sites fonctionnant avec TRILL.

3.3.6 Extending TRILL over WAN

Dans [59] la solution proposée implique d'avoir l'unicité des nicknames dans tous les sites TRILL et d'utiliser les adresses IP pour réaliser l'interconnexion de ces sites. Cette solution n'impose aucune spécificité pour le réseau public puisqu'il est utilisé au niveau 3 avec IP sans technologie de VPN. Le fait d'utiliser les adresses IP pour l'interconnexion des sites TRILL oblige les RBridges de bordure à encapsuler les messages TRILL dans des paquets IP et de connaître les associations adresses IP / nicknames. Ici, on parle d'association car ce ne sont pas seulement des couples 1 adresse IP / 1 nickname. Il est possible d'avoir plusieurs RBridges de bordure à chaque site et donc d'avoir autant d'adresses IP pour un seul nickname. Plus précisément, tous les nicknames des RBridges

d'un site seront associés à toutes les adresses IP des RBridges de bordure de ce site (voir Figure 3.6).



Nickname	Interface
S1	self
S2	-
S3	IPC
S4	IPD
S10	L1
S11	L2
S20	IPC/IPD
S21	IPC/IPD

FIGURE 3.6 – Exemple de topologie (réseau cœur WAN et data centers TRILL) avec la table de commutation de S1

Dans cette solution, le problème d'épuisement des nicknames n'est pas traité et le fait d'avoir à stocker tous les nicknames avec, à chaque fois, la liste de toutes les adresses IP pour les joindre, impose d'avoir des RBridges de bordure avec d'importantes ressources. De plus, les messages de contrôle des différents sites doivent être diffuser à tous les sites, car cette solution réalise une fusion des différents sites TRILL en un seul site. C'est pourquoi cette solution ne nous convient pas.

3.3.7 TRILL : Campus Label and Priority Regions

Le draft "draft-ietf-trill-rbridge-vlan-mapping" [60] propose l'ajout d'une nouvelle fonction pour les RBridges appelés "cut set". Ces RBridges sont ceux qui interconnectent différentes régions d'un campus TRILL. Si ces RBridges sont enlevés, alors les régions ne peuvent plus communiquer entre elles. Il est possible de configurer ces RBridges pour traduire les *data labels* d'une région vers les *data Labels* d'une autre région. Un *data Label* peut correspondre, soit à un identifiant de VLAN, soit à un Fine Grained Label. Dans notre cas, avec VNT, le label peut aussi être le tag VNI d'un client. De plus, on peut aussi configurer la priorité de ces *data Labels*. Cette notion de traduction de data Labels et de priorité est la proposition de ce draft. L'implémentation de la proposition est expliquée dans le draft. Cette implémentation ne concerne que les RBridges appartenant au "cut set" et ne modifie en rien, ni le comportement, ni la configuration des autres RBridges du campus TRILL. L'implémentation oblige les RBridges du "cut set" à définir l'appartenance à une région pour chacun de leurs ports. De plus, même les ports connectant des RBridges du "cut set" entre eux doivent appartenir à une région. Il est possible d'avoir plusieurs régions interconnectées par un "cut set". Cependant, tous les RBridges d'un "cut set" interconnectant les mêmes régions doivent être configurés de façon similaire. De plus, il est recommandé d'avoir une traduction symétrique des labels et des priorités. Par

exemple, le flux de la région A avec le Label 1 est traduit dans la région B avec le label 3 et vice versa.

(Region A / Label 1) <———> (Region B / Label 3)

Les RBridges d'un "cut set" apprennent les nicknames et adresses MAC comme les autres RBridges, lors de leurs découvertes.

Cette approche permet de relier deux régions d'un campus ayant chacune leurs propres VLANS ou Fine Grained Label. Néanmoins, il faut tout de même avoir l'unicité des nicknames au sein de ce réseau, ce qui ne permet pas de parer au manque de nicknames, ni de pouvoir réutiliser ces nicknames.

3.3.8 RBridge : Pseudo-Nickname for Active-active Access

Le problème évoqué dans "draft-hu-trill-pseudonode-nickname" [61] concerne la possibilité qu'une station terminale puisse être connectée à plusieurs RBridges d'un même campus. Cette station possède donc de multiples connexions point à point avec différents RBridges d'un même campus. Elle agrège ces connexions multiples et forme un Multi-Chassis Link Aggregation (MC-LAG). Ce MC-LAG permet d'utiliser tous les liens vers tous les RBridges. Cependant, au sein du MC-LAG, les messages "HELLO" reçus sur un port ne seront pas diffusés sur les autres ports. Cette fonction implique que chaque RBridge connecté à ce MC-LAG va se percevoir comme étant le diffuseur élu pour tous les VLANs utilisés par ce MC-LAG. De ce fait, chaque RBridge va donc faire entrer et sortir les messages de ces VLANs dans ou depuis le campus TRILL. On a donc un souci de duplication du message initial puisque l'on peut avoir autant de copies de ce message que de RBridges connectés à ce MC-LAG.

3.4 Contributions

Les solutions pour interconnecter des sites TRILL, présentées dans le chapitre précédent (3.3), ne permettent pas d'avoir des sites TRILL indépendants. En effet, soit les sites TRILL sont fusionnés en un seul site, soit des informations de chaque site doivent être partagées avec les autres sites. Seule la solution présentée dans le draft "Flexible Multilevel TRILL" [58] permet d'avoir des sites TRILL interconnectés sans les fusionner. Cependant, cette solution possède d'autres inconvénients comme le multi chemins qui n'est plus possible pour les flux entre sites. En raison de ces contraintes, nous avons choisi de développer notre propre solution. Ce développement s'est déroulé en deux étapes. La première concerne le développement d'une solution permettant d'interconnecter des sites TRILL sans les fusionner tout en augmentant le passage à l'échelle de TRILL. Cette solution, appelée "Simple Multi Level TRILL Protocol" (SMLTP), est détaillée dans la

section 3.4.1. Néanmoins, elle est proche du draft "Flexible Multilevel TRILL" [58] dans sa conception. Le trafic inter-campus passe actuellement par une seule gateway. Pour supprimer cette contrainte nous avons, dans un second temps, développé la solution "Multi Level TRILL Protocol" (MLTP) détaillée dans la section 3.4.2

3.4.1 Simple Multi Level TRILL Protocol (SMLTP)

3.4.1.1 Présentation

Nos objectifs, lors de la conception de SMLTP, étaient d'avoir une solution qui permette d'interconnecter des sites TRILL sans les fusionner et en préservant l'indépendance de leur plan de contrôle. Pour réaliser cela, nous avons conçu une "frontière" entre les différents sites TRILL. Celle-ci permet d'isoler leurs plans de contrôle tout en leur permettant de communiquer entre eux.

Le protocole TRILL utilise le protocole IS-IS pour établir la topologie du réseau. Nous avons donc étudié son fonctionnement (Section 2.5). IS-IS fonctionne à l'aide d'"areas" qui sont interconnectées via un réseau cœur appelé "backbone". Pour réaliser cette séparation, IS-IS possède une hiérarchie à deux niveaux dans laquelle le premier niveau est attribué aux "areas" et le second au "backbone". Les "areas" sont connectées au "backbone" et peuvent communiquer entre elles. Cependant, le protocole IS-IS utilisé au sein de TRILL a été modifié pour répondre aux contraintes de celui-ci. L'une de ces contraintes est de n'avoir qu'une seule "area" dans un réseau TRILL. Cela résulte de ce que le protocole TRILL a été conçu pour être utilisé au sein de réseaux LAN où seulement une "area" est présente.

Notre solution (SMLTP) s'inspire du fonctionnement du protocole IS-IS original. Nous y avons introduit une notion de hiérarchie à deux niveaux. Le premier niveau est associé aux sites TRILL. Nous appelons ces sites TRILL de niveau 1 des "Campus TRILL". Le second niveau est dévolu au réseau cœur, le "backbone". Nous avons aussi développé un nouvel équipement appelé Border RBridge (BRB) (Section 3.4.1.2).

Ces améliorations, tant du plan de contrôle (Section 3.4.1.3) que du plan de données (Section 3.4.1.4), permettent d'obtenir des plans de contrôle indépendants tout en interconnectant les différents sites au réseau cœur. Les performances de SMLTP sont présentées dans la section 3.4.1.6. Néanmoins, cette solution possède des limites qui sont détaillées dans la section 3.4.1.7.

3.4.1.2 Nouvel équipement : Le Border RBridge (BRB)

Le Border RBridge (BRB) appartient aux deux niveaux et représente la "frontière" entre ceux-ci. Il réalise la jonction entre un site TRILL de niveau 1 et le réseau cœur

TRILL de niveau 2. Il permet aussi de limiter le domaine de broadcast du site TRILL ainsi que du réseau cœur en empêchant les informations de leurs plans de contrôle d'atteindre l'autre niveau. Le plan de contrôle du site TRILL, de niveau 1, est isolé de celui du "backbone" de niveau 2. Néanmoins, un BRB doit permettre la transition des paquets de données d'un niveau à l'autre pour que les sites TRILL puissent communiquer.

Un BRB possède trois niveaux d'interfaces (Figure 3.7). Ceci permet d'établir la frontière entre les deux niveaux car le BRB ne peut envoyer de message d'un niveau qui ne correspond pas avec l'interface. Les interfaces du BRB, orientées vers le campus, possèdent le niveau 1. Celles tournées vers le backbone sont de niveau 2. Le troisième niveau d'interface correspond au niveau 1&2 et est réservé aux interfaces reliant deux BRBs d'un même campus.

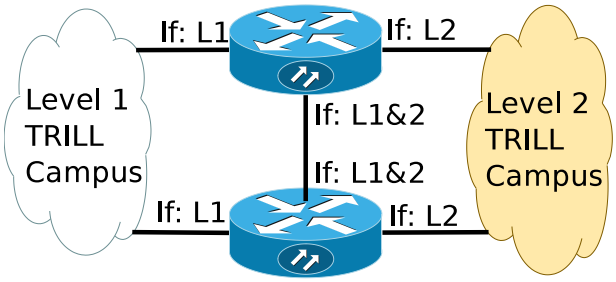


FIGURE 3.7 – Détail des interfaces des BRBs

3.4.1.3 Conception du plan de contrôle de SMLTP

Le plan de contrôle de SMLTP diffère de celui de TRILL sur trois points. Le premier est la gestion des nicknames (Section 3.4.1.3.1). Ensuite, l'ajout d'un niveau implique de modifier l'apprentissage de la topologie (Section 3.4.1.3.2). Finalement, le calcul des arbres de distribution est aussi modifié (Section 3.4.1.3.3). Malgré ces modifications, SMLTP est compatible avec TRILL (Section 3.4.1.5).

3.4.1.3.1 Gestion des nicknames

Grâce à l'isolation des plans de contrôle de chaque campus, les nicknames utilisés pour les RBridges du campus peuvent être réutilisés au sein des autres campus. Néanmoins, seuls les nicknames attribués aux RBridges de niveau 1 peuvent être réutilisés dans chaque campus. En effet, les nicknames attribués aux BRBs ne peuvent pas être réutilisés dans le réseau. Par exemple dans la Figure 3.8, si le RBridge E utilise le même nickname que le BRB W alors les deux BRBs Y et Z voient qu'il y a un conflit de nicknames, mais les responsables de ce conflit (BRB W et RBridge E) ne peuvent le résoudre. En effet, la détection de collision des nicknames de TRILL ne peut pas fonctionner au sein de SMLTP car les deux nœuds (W et E) ne sont pas dans le même plan de contrôle. Y et Z ne savent donc pas à quel nœud envoyer les paquets car le nickname destination est en conflit entre

W et E. Pour résoudre ce problème, nous avons défini que les BRBs ont une priorité plus importante que les RBridges et, donc, dans notre exemple, les BRBs Z et Y vont associer le nickname au BRB W. Il en résulte que le RBridge E peut continuer de communiquer au sein du campus mais ne peut plus avoir de communication inter-campus. Pour y remédier, il faut que l'administrateur du réseau change son nickname.

Les nicknames des BRBs ne peuvent donc pas être réutilisés car ils sont utilisés pour router les paquets entre sites. Comme TRILL ne permet pas de distinguer les nicknames utilisés par les RBridges et les BRBs, puisqu'il ne possède pas de notion de niveau, nous avons divisé l'ensemble des nicknames en deux groupes. Le premier groupe contient les nicknames de niveau 1 qui possèdent le "Most Significant Bit" (MSB) à 0. Ces nicknames ont une signification restreinte au sein d'un site et peuvent donc être réutilisés dans les autres sites. Par opposition, les nicknames de niveau 2 doivent être uniques dans tout le réseau SMLTP et possèdent le MSB à 1.

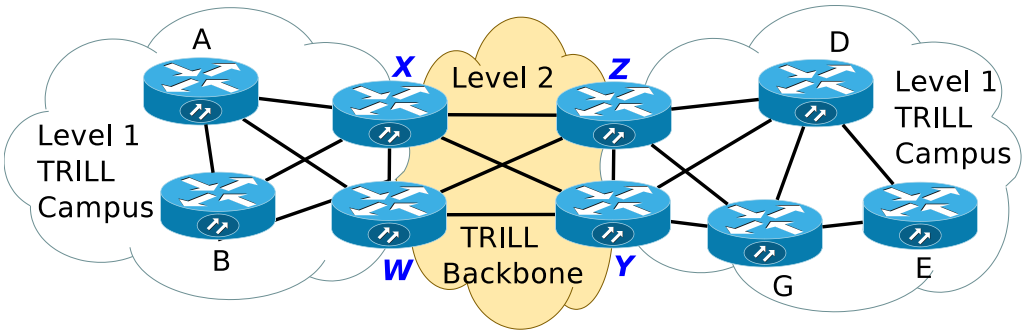


FIGURE 3.8 – Position des Border RBridges dans le réseau

3.4.1.3.2 Apprentissage de la topologie

La découverte et l'apprentissage de la topologie du réseau au sein de TRILL sont réalisés à l'aide d'une version modifiée du protocole IS-IS [50]. Cette découverte ne se fait qu'au niveau 1 de IS-IS, à l'aide des messages "Hello" et LSPs de niveau 1, car TRILL ne possède pas de notion de niveau et, par extension, ne possède pas de niveau 2. Par opposition, SMLTP utilise cette notion de niveau et possède un niveau 2. Nous avons donc modifié la découverte et l'apprentissage de la topologie en utilisant aussi les messages de niveau 2 de IS-IS. Les messages de niveau 1 sont utilisés pour découvrir le campus et ceux de niveau 2 pour découvrir le backbone.

Concernant le niveau 1, l'utilisation des messages, ainsi que la découverte et l'apprentissage de la topologie du campus suivent les instructions du RFC 7177 de TRILL [62]. Les RBridges du campus n'utilisent que les messages de niveau 1 et suivent aussi le RFC 7177. Les BRBs participent à ces échanges de messages de niveau 1 en suivant le RFC 7177 via leurs interfaces de niveau 1 et de niveau 1&2. De cette façon, tous les nœuds du campus (les RBridges et les BRBs) participent à la découverte de la topologie du campus.

Appartenant aux deux niveaux, les BRBs réalisent aussi la découverte du backbone en

utilisant des "Hello" et des LSPs de niveau 2. Ces messages sont envoyés par les BRBs via leurs interfaces de niveau 2 et de niveau 1&2 pour que tous les BRBs du même campus partagent la vue du niveau 2 et puissent annoncer les mêmes voisins. Les BRBs doivent aussi conserver les plans de contrôle indépendants. Il est donc nécessaire qu'ils empêchent les messages d'atteindre le mauvais niveau. Ceci est réalisé lors de l'envoi du message à l'aide du niveau de l'interface (Section 3.4.1.2 qui permet de décider si le message doit ou non être envoyé.

Les interfaces de niveau 1&2 envoient les messages des deux niveaux ce qui permet aux BRBs d'un même campus de partager les informations de topologie de chaque niveau. Néanmoins, pour que ce partage puisse avoir lieu, il est obligatoire que les BRBs soient connectés directement par le biais des interfaces de niveau 1&2. Cette contrainte provient de ce que le réseau cœur formé par IS-IS ne peut être segmenté. Ceci implique que les nœuds de niveau 1 sur le chemin entre deux nœuds de niveau 2 doivent changer de niveau pour devenir des nœuds de niveau 1&2.

3.4.1.3.3 Calcul des arbres de distribution

TRILL "forward" les trames multicast suivant un ou plusieurs arbres de distribution. Ces arbres sont connus de tous les RBridges du réseau TRILL, or ce n'est pas possible au sein de SMLTP car tous les RBridges ne partagent pas le même plan de contrôle. Pour envoyer les trames multicast au sein de SMLTP il est nécessaire de calculer un arbre de distribution dans chaque site et dans le réseau cœur. Les BRBs doivent connaître les arbres de distribution de leurs campus et du backbone pour pouvoir joindre la destination. Il est donc nécessaire que les BRBs convertissent les paquets de niveau 1 en niveau 2 et vice versa (Section 3.4.1.4). L'arbre de distribution des sites est calculé comme décrit dans le RFC 7177 de TRILL. Concernant l'arbre de niveau 2, il est nécessaire de conserver tous les liens inter BRBs des campus lors du calcul de celui-ci. Pour empêcher la formation des boucles au sein de cet arbre, il est possible de "couper" les liens redondant du backbone. Pour savoir quel lien inter campus conserver, le choix est réalisé en se basant sur la valeur des priorités et/ou de l'adresse MAC des BRBs. Par exemple, dans la Figure 3.9, le lien A-B est le lien L2 qui possède soit la somme des priorités (prio) la plus élevée, soit la somme des adresses MAC la plus élevée si la priorité ne permet pas de décider. On a donc (prioA + prioB) > (prioA + prioD) > (prioC + prioD) > (prioC + prioE) > (prioF + prioE). Si on a une égalité du type (prioA + prioB) = (prioA + prioD) alors on décide via la valeur des MAC et on obtient ainsi (MAC A + MAC B) > (MAC A + MAC D) donc le lien conservé est le lien A-B.

3.4.1.4 Conception du plan de données de SMLTP

Le plan de contrôle de TRILL a aussi dû être modifié pour gérer les deux niveaux. L'en-

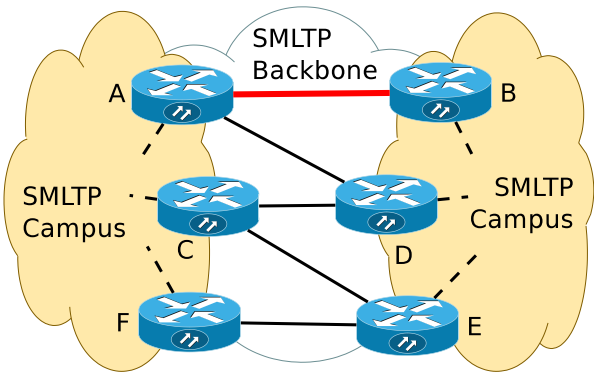


FIGURE 3.9 – Détail de l'arbre de diffusion du niveau 2

tête du protocole TRILL a été amélioré pour enregistrer la valeur du niveau du message. Cette modification est détaillée dans la section 3.4.1.4.1. En raison de cette modification, le processus d'envoi des messages a été modifié pour gérer cette notion de niveau (Section 3.4.1.4.2). La modification de l'en-tête TRILL implique aussi de modifier les processus de réception des messages unicast et multicast détaillés dans les Sections 3.4.1.4.3 et 3.4.1.4.4.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                | V |Lvl|M|Op-Length| Hop Count |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Egress RBridge Nickname     |    Ingress RBridge Nickname   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Options...
+-+-+-+-+-+-+-+-+-+-+-+-
V=Version    Lvl=Level    M=Multicast    Op-Length=Option length
```

FIGURE 3.10 – SMLTP header

3.4.1.4.1 Modification de l'en-tête

Pour enregistrer le niveau du message, nous avons utilisé le champ réservé de deux bits que l'on a renommé Level. La Figure 3.10 montre la disposition du champ Level au sein de l'en-tête de SMLTP. Ce champ Level est utilisé pour empêcher la création de boucle. Par exemple, si un BRB reçoit une trame de niveau 2 depuis le backbone et qu'elle doit être inondée dans le campus, le BRB va identifier cette trame comme étant une trame de niveau 1&2. De cette façon, les autres BRBs du campus reconnaîtront ce message comme provenant du backbone et donc ne le renverront pas vers le backbone.

Pour être compatible avec TRILL, nous avons volontairement utilisé le champ réservé. Cependant, comme ce champ est nécessaire pour le fonctionnement des BRBs, les RBridges TRILL ne peuvent être utilisés qu'à l'intérieur des campus. Les spécifications de TRILL indiquent que le champ réservé ne doit pas être utilisé par les RBridges et doit

avoir la valeur 0. C'est dans ce but que l'on utilise la valeur 0 pour indiquer le niveau 1 afin de pouvoir utiliser les RBridges TRILL au sein des campus comme RBridge de niveau 1.

3.4.1.4.2 Processus d'envoi de message

Du fait de la modification de l'en-tête que nous avons réalisé (Section 3.4.1.3.1) le processus d'envoi des messages a été modifié. Les modifications doivent intégrer le niveau du message au paquet. De plus les BRBs doivent aussi réaliser des opérations pour transiter un message de niveau 1 vers le niveau 2 et vice versa. Dans la Figure 3.11, les nouvelles étapes sont en rouge. On peut remarquer que la majorité des modifications ne concerne que les BRBs.

La modification pour les RBridges de niveau 1 consiste à ajouter le niveau du message dans le champ level de l'en-tête du paquet. Pour les RBridges TRILL qui ne sont pas compatibles SMLTP et qui ne possèdent pas ce nouveau processus d'envoi, l'ajout du niveau dans le message n'est pas possible. Cependant, comme précisé dans la section 3.4.1.3.1, ces RBridges vont suivre les spécifications de TRILL et mettre la valeur par défaut, à savoir 0, dans ce champ. Or, pour SMLTP, cette valeur correspond au niveau 1.

En ce qui concerne les BRBs, le processus d'envoi est modifié pour prendre en considération le niveau de l'environnement dans lequel le paquet est émis. En effet, un BRB peut envoyer des paquets, soit dans le campus (niveau 1), soit dans le backbone (niveau 2). Conséquemment, le processus d'envoi doit être distinct pour chaque niveau. Pour les paquets unicast dont le nickname destination est connu, un BRB doit vérifier le niveau du prochain nœud dans le chemin. En fonction de celui-ci, il va mettre, soit un en-tête de niveau 1, soit de niveau 2. Pour les paquets multicast, le BRB doit, dans un premier temps, vérifier s'il existe un Designated Tree root (DTroot) pour chaque niveau. Si le DTroot n'existe pas pour un niveau, alors le paquet n'est pas envoyé dans ce niveau. Une fois que le BRB a vérifié l'existence du DTroot, il vérifie s'il existe un nœud sur l'arbre de distribution à qui il peut envoyer le paquet. Si ce nœud existe, alors le BRB encapsule le paquet avec un en-tête SMLTP possédant les champs level et destination correspondants respectivement soit au niveau 1 et au nickname du DTroot L1 si le paquet est émis dans le campus, soit au niveau 2 et au nickname du DTroot L2 si le paquet est émis dans le backbone.

3.4.1.4.3 Processus de réception de messages unicast

À l'instar du processus d'envoi d'un message, le processus de réception d'un message unicast diffère entre les RBridges et les BRBs. Pour les RBridges, le processus de réception a été modifié en raison de l'intégration d'une nouvelle étape qui vérifie si le niveau du paquet est identique au niveau de l'interface ayant reçu le paquet. Si ce n'est pas le cas, alors le paquet est supprimé. C'est la seule modification apportée au processus de

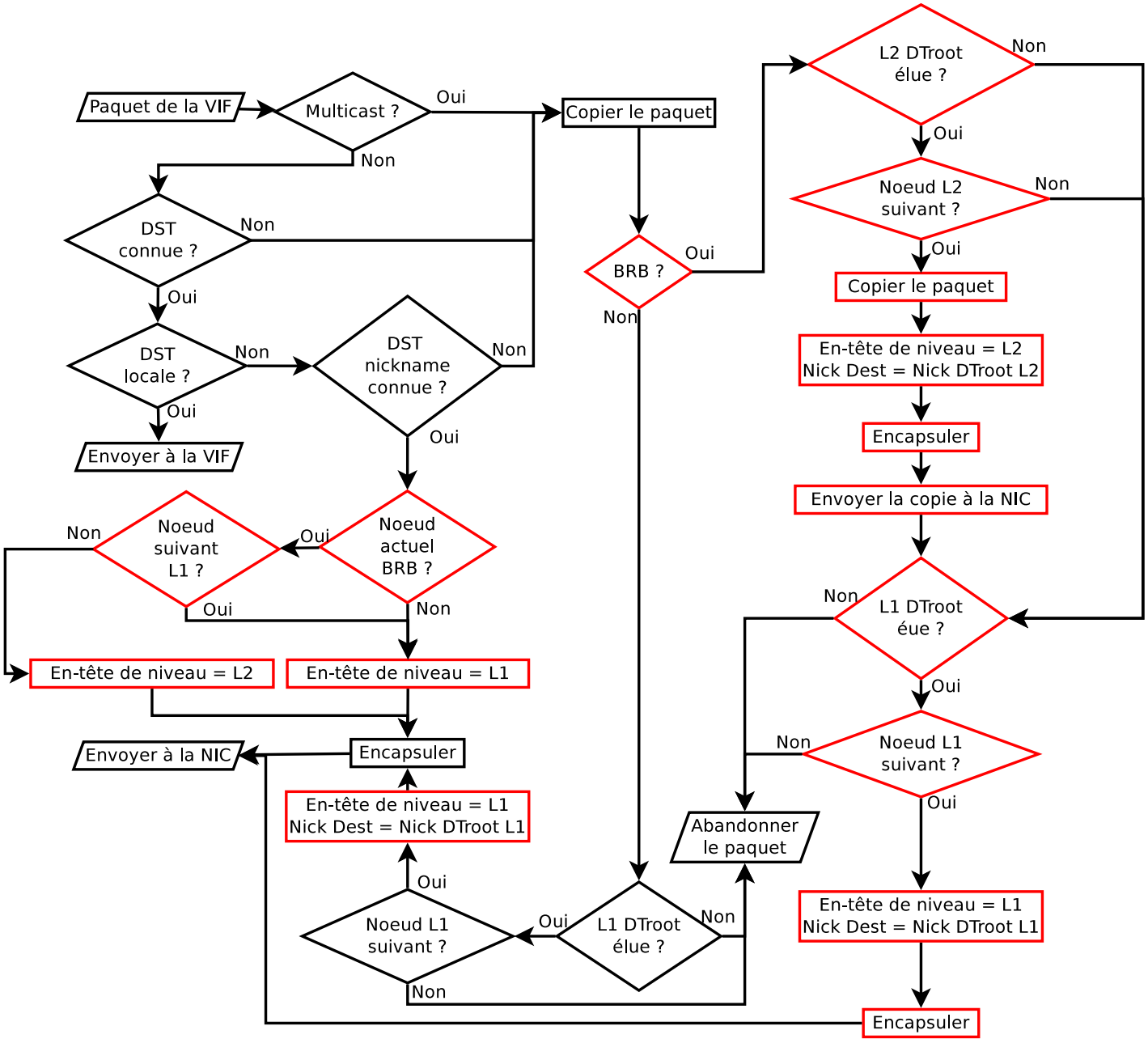


FIGURE 3.11 – Processus d'envoi d'un message

réception des paquets unicast pour les RBridges.

À l'opposé, le processus de réception des paquets unicast pour les BRBs est complètement nouveau. Lorsqu'un BRB reçoit un paquet, celui-ci vérifie si le niveau du paquet est compatible avec le niveau de l'interface qui l'a reçu. Ensuite il vérifie si le nickname destination du paquet est le sien ou non. Si le paquet est destiné au BRB, alors ce paquet appartient à une communication inter-campus et le BRB doit faire transiter le paquet du backbone vers le campus ou vice versa. Pour cela le BRB enregistre le nickname source du paquet ainsi que l'adresse MAC source, puis vérifie s'il connaît l'adresse MAC destination. Dans le cas où cette adresse lui est inconnue, alors il supprime le paquet car cela indique que la résolution préalable de l'adresse MAC via ARP n'a pas été effectuée

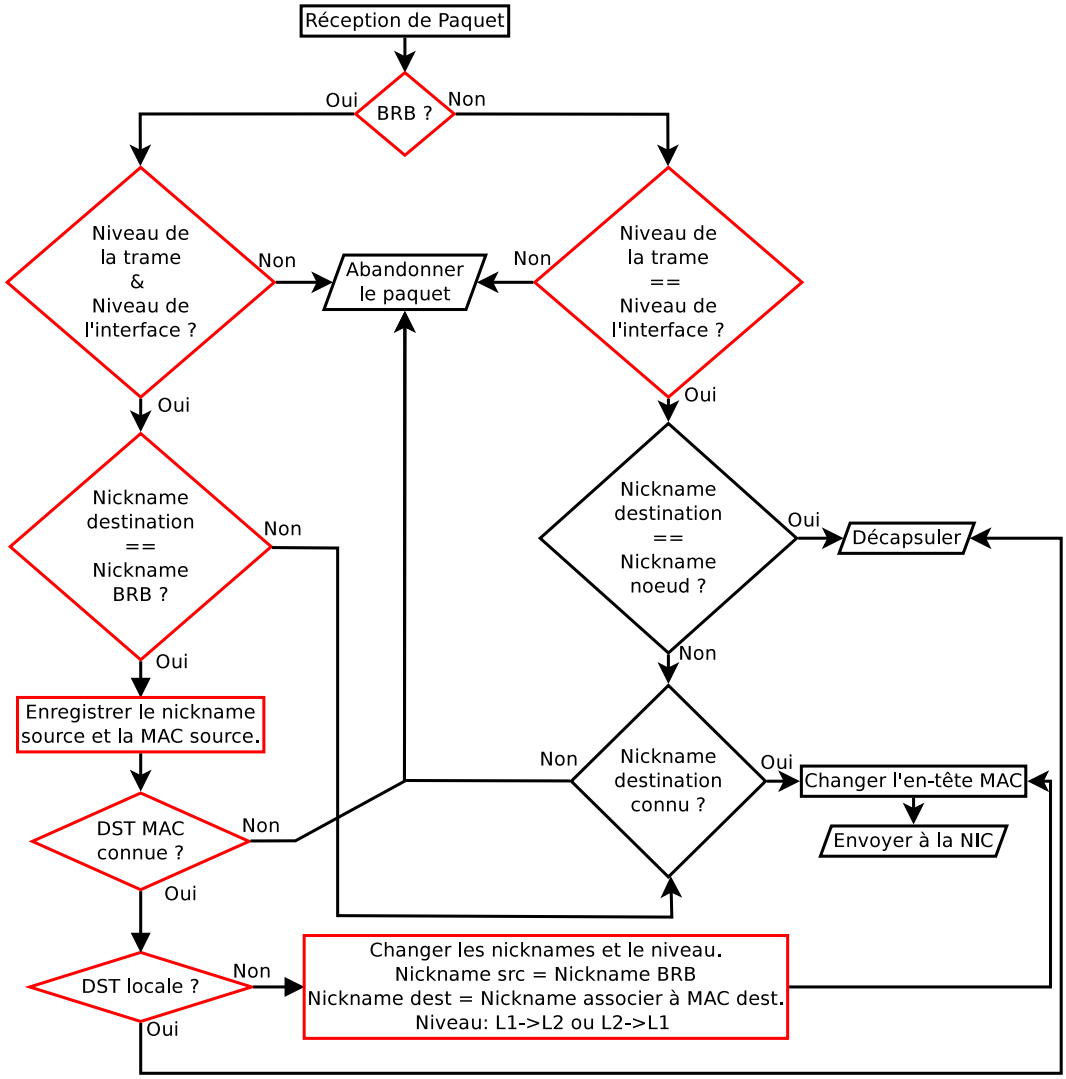


FIGURE 3.12 – Processus de réception d'un message unicast

et, donc, qu'aucun BRB ne connaît la destination. Au contraire, si le BRB connaît le nickname associé à cette adresse MAC, alors il modifie l'en-tête SMLTP en remplaçant le nickname source par le sien et le nickname destination par celui associé à l'adresse MAC destination. Il change aussi le niveau du paquet en fonction du niveau original de celui-ci. En effet, si le paquet est de niveau 1, alors le nouvel en-tête sera de niveau 2 et vice versa. Le processus de réception des paquets unicast au niveau des BRBs doit permettre le transit des paquets entre les niveaux 1 et 2, c'est-à-dire entre le campus et le backbone. La Figure 3.12 résume les différentes étapes du processus de traitement des messages unicast.

3.4.1.4.4 Processus de réception multicast

Comme pour le processus de réception unicast, on peut voir sur la Figure 3.13 que le processus de réception des paquets multicast est légèrement modifié pour les RBridges mais complètement nouveau pour les BRBs. Concernant le processus pour les RBridges,

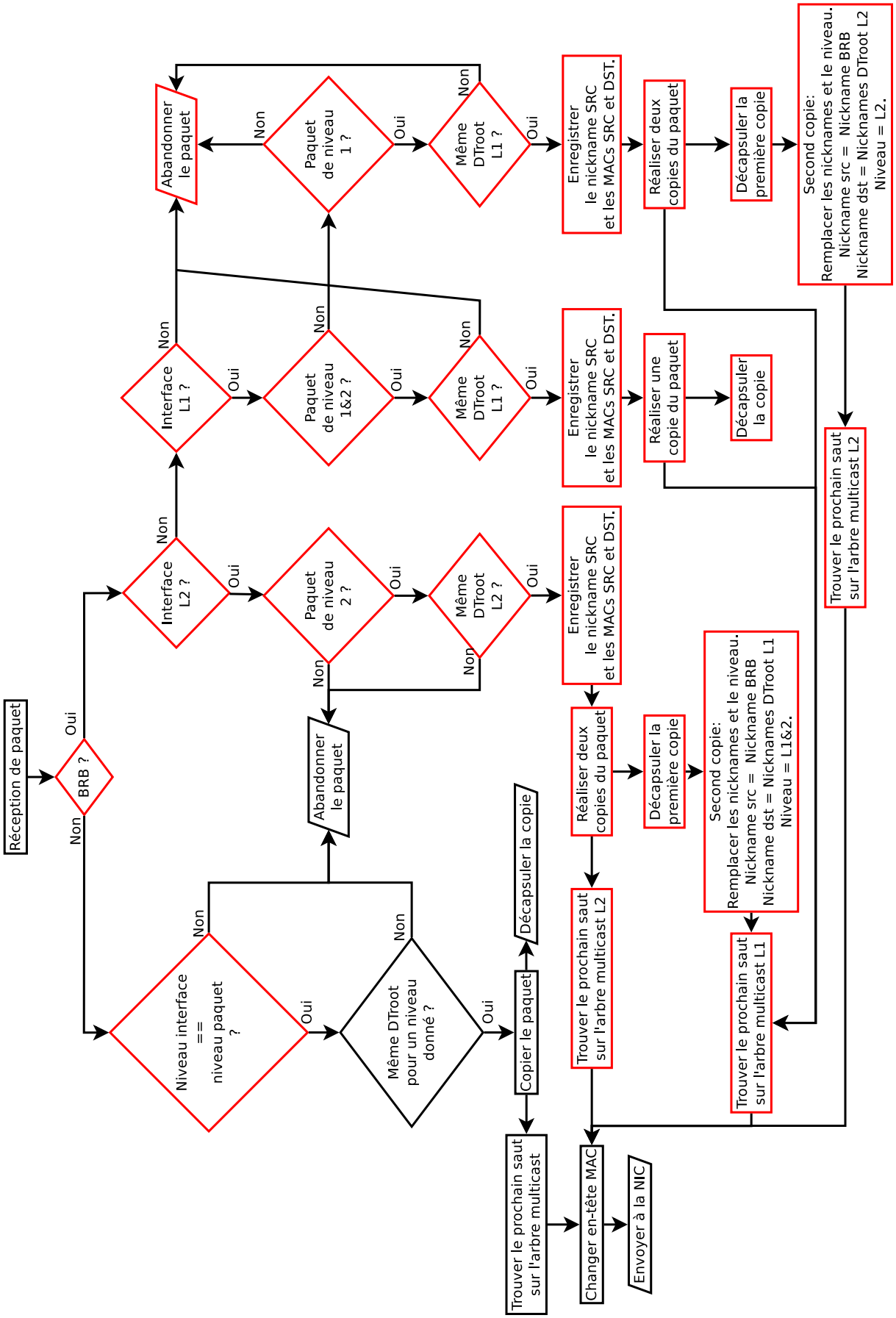


FIGURE 3.13 – Processus de réception d'un message multicast

on a ajouté l'étape de vérification de niveau du paquet qui est nécessaire pour assurer le respect de l'isolation des campus et du backbone. Pour les BRBs, nous avons conçu un nouveau processus de réception des paquets multicast qui doit gérer la propagation du paquet suivant l'arbre de distribution et faire transiter le paquet d'un niveau à l'autre. Dans un premier temps, le BRB vérifie si le niveau du paquet est supporté par l'interface, puis, si le paquet est supporté, le BRB vérifie s'il connaît le nickname destination et s'il correspond au DTroot qu'il connaît pour ce niveau. Si le nickname destination diffère du DTroot connu par le BRB, alors le paquet est supprimé car le BRB est dans l'incapacité de propager ce paquet car il ne connaît pas l'arbre de distribution associé au nickname destination. Au contraire, si le nickname destination et le DTroot correspondent, alors le BRB enregistre le nickname source du paquet ainsi que les adresses MAC source et destination. Le BRB enregistre ces informations pour être en mesure de faire transiter la ou les réponses dans le sens inverse. Ensuite, il réalise une copie du message et ainsi obtient deux messages identiques. La copie du message est propagée le long de l'arbre de distribution du niveau de réception. C'est à dire que, si le paquet vient du backbone, alors la copie du message sera propagée le long de l'arbre de distribution du backbone. Si le paquet émanait du campus, alors la copie serait propagée le long de l'arbre de distribution du campus. Une fois la copie propagée, le BRB modifie l'en-tête du paquet en remplaçant les nicknames et le niveau du paquet. Le BRB place son nickname comme nickname source du paquet et remplace le nickname destination avec le nickname du DTroot du niveau opposé. En effet, le nickname destination devient le nickname du DTroot du backbone si le paquet original provient du campus et il devient le nickname du DTroot du campus si le paquet provient du backbone. La modification du niveau du paquet dépend aussi de la provenance du paquet. Si le paquet vient du campus, alors le nouveau niveau du paquet sera le niveau L2 alors que si le paquet vient du backbone, le nouveau niveau sera le niveau L1. Le fait de déclarer un paquet comme étant de niveau L1&2 permet d'annoncer aux autres BRBs du campus que le paquet est originaire du backbone et qu'il ne faut pas qu'il le fasse à nouveau transiter vers le backbone. Une fois les modifications sur l'en-tête SMLTP réalisées, le paquet est propagé le long de l'arbre de distribution adéquat.

3.4.1.5 Rétro-compatibilité

Le plan de contrôle de SMLTP est compatible avec TRILL tant que les RBridges TRILL sont utilisés en tant que RBridges de niveau 1 avec des nicknames de niveau 1. Il se trouve que le plan de contrôle de niveau 1 est identique au plan de contrôle de TRILL car notre travail s'est focalisé sur les BRBs et le niveau 2. L'unique problème pouvant se poser résulte de la possession conjointe d'un même nickname par un BRB et un nœud TRILL d'un autre site. Il y a donc un problème de routage au niveau de la gateway du site dans lequel se trouve le nœud TRILL. Pour résoudre ce problème, la gateway ne conserve que le

nickname du BRB car il possède une priorité supérieure au RBridge TRILL. Néanmoins, le RBridge TRILL ne peut plus communiquer qu'au sein de son site et, pour résoudre cela, il faut donner un nickname de niveau 1 au RBridge.

Concernant le plan de données, SMLTP a modifié l'en-tête TRILL en utilisant le champ réservé pour y enregistrer le niveau du paquet. Comme précisé dans la section 3.4.1.4.1 nous avons utilisé la valeur par défaut de TRILL pour coder le niveau 1, nous permettant ainsi d'être compatible avec TRILL au sein des campus. En effet, le protocole TRILL impose que les RBridges d'origine du paquet mettent le champ réservé à zéro. Cependant, les RBridges de transit et le RBridge destination doivent ignorer le champ réservé comme expliqué dans la Section 3.3 du RFC6325 [11]. Ceci nous permet donc de propager les paquets de niveau 1&2 au sein du campus tout en étant compatible avec les RBridges TRILL. Il en résulte que SMLTP est retro-compatible avec TRILL à condition de n'utiliser les RBridges TRILL qu'au sein des campus.

3.4.1.6 Performances

Dans cette section, nous étudions les performances du plan de contrôle et du plan de données de SMLTP. Nous comparons les performances de SMLTP avec celles d'un réseau Ethernet standard, qui est la solution native de Linux, et avec les performances de TRILL.

3.4.1.6.1 Implémentation de la solution

Avant de tester SMLTP, il nous a fallu l'implémenter. À cette fin, nous avons utilisé le daemon Quagga en version 0.99.22.4 [63], qui a été préalablement modifié dans les travaux réalisés en [14], comme base pour le plan de contrôle de SMLTP. Nous l'avons ensuite modifié pour qu'il gère la hiérarchie, à deux niveaux, nécessaire pour SMLTP. Nous avons, plus précisemment, amélioré la partie concernant IS-IS au sein du daemon Quagga ainsi que la partie TRILL ajoutée via les modifications réalisées par [14]. Concernant le plan de données, nous avons modifié le module Bridge de Linux proposé dans [64] pour qu'il puisse réaliser les nouveaux processus d'envoi et de réception des paquets décrits dans les Sections 3.4.1.4.2, 3.4.1.4.3 et 3.4.1.4.4.

3.4.1.6.2 Plate-forme de test

Notre plate-forme de test est composée de six serveurs. Les serveurs 1, 2 et 3 sont des serveurs DELL C6100 qui utilisent chacun un processeur Intel Xeon L5640 fonctionnant à 2.27GHz et qui possèdent 46 GB de mémoire RAM. Les trois autres serveurs (4, 5 et 6) sont des serveurs DELL PowerEdge R310 avec chacun un processeur Intel Xeon L3426 fonctionnant à 1.87GHz et qui ont 16 GB de mémoire RAM. Ces serveurs font tourner XEN 3.0 [65] avec une version modifiée de Linux 3.13.0 comme DOM0. La topologie physique de la plate-forme de test est présentée dans la Figure 3.14. On y voit deux

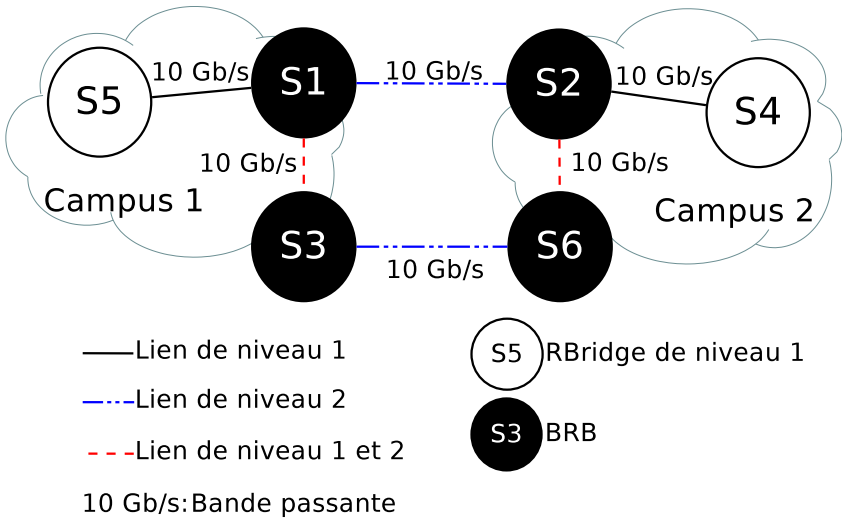


FIGURE 3.14 – Topologie physique de la plate-forme de test

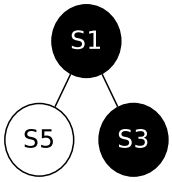


FIGURE 3.15 – Arbre de distribution du campus 1

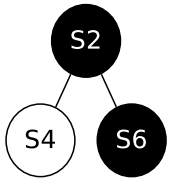


FIGURE 3.16 – Arbre de distribution du campus 2

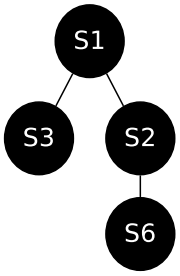


FIGURE 3.17 – Arbre de distribution du backbone

campus, chacun composé d'un RBridge et de deux BRBs. Les trois arbres de distribution, un pour chaque campus et un pour le backbone, sont présentés dans les Figures 3.15,3.16 et 3.17. On y voit que les BRBs possèdent une priorité supérieure aux RBridges car le DTroot des arbres de distribution est toujours un BRB.

3.4.1.6.3 Étude du plan de contrôle

Pour évaluer le plan de contrôle de SMLTP nous avons étudié son temps de convergence. Nous définissons le temps de convergence comme le temps écoulé entre l'apparition d'un nouveau nœud sur le réseau et le moment où tous les autres nœuds de ce réseau l'ont découvert. Pour récupérer ce temps de convergence, nous avons modifié les messages "Hello" de niveau 1 en leur ajoutant le temps de vie (uptime) du nœud qui envoie le message "Hello". Grâce à ce temps de vie, chaque nœud sait depuis combien de temps le nœud à qui appartient le "Hello" est démarré. De plus, en utilisant ce système de temps de vie, nous n'avons pas le problème de synchronisation temporelle entre les nœuds. En effet, chaque nœud calcule le moment de démarrage des autres nœuds en utilisant sa propre référence temporelle. À l'aide de ces informations nous pouvons calculer le temps de convergence de chaque nœud.

Pour obtenir le temps de convergence du plan de contrôle de SMLTP nous utilisons

des VMs comme nœuds, ce qui nous permet d'augmenter le nombre de nœuds dans la topologie. Nous divisons les nœuds en trois catégories qui sont :

— Le nouveau RBridge.

— Le DTRoot (Designated Tree Root).

— Les RBridges déjà présents dans le réseau.

Lors de cette évaluation nous définissons le temps de rafraîchissement des LSPs à 15 secondes et le premier LSP n'est envoyé que 30 secondes après le début de l'expérience. Au cours de cette expérience, de nouveaux nœuds sont connectés à la topologie à des intervalles de temps aléatoires.

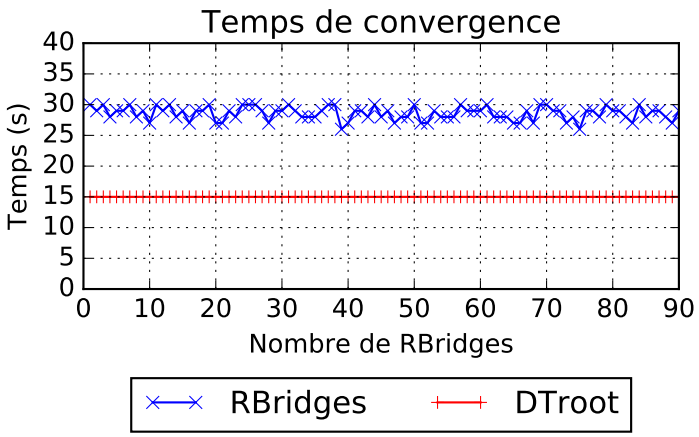


FIGURE 3.18 – Évolution du temps de convergence du plan de contrôle de SMLTP en fonction du nombre de RBridges.

La Figure 3.18 nous apprend que seul le temps de rafraîchissement des LSPs impacte le temps de convergence du plan de contrôle. En effet, le temps de convergence est au maximum de 30 secondes, ce qui correspond au temps de deux LSPs. La raison est que, pour converger, deux nœuds doivent se voir réciproquement. Dans le pire des cas, le nouveau nœud (A) envoie son premier LSP et un autre nœud (B) le reçoit juste après avoir rafraîchi sa base de données contenant les LSPs. B ne va pas voir A pendant encore, au maximum, 15 secondes, le temps de rafraîchir sa base de données contenant les LSPs. Après ces 15 secondes, B va voir A et envoyer son LSP contenant cette information. A reçoit le LSP de B et sait maintenant que B le voit en plus du fait qu'il voit B. À ce moment-là, les deux nœuds A et B ont convergé. Il aura donc fallu attendre les 15 secondes nécessaires à l'envoi du LSP de A et les 15 secondes pour l'envoi du LSP de B pour que les deux nœuds convergent. Il faut donc, dans le pire des cas, attendre un temps maximum de 30 secondes pour que deux nœuds convergent.

Cependant, les résultats présentés dans la Figure 3.18 ne s'appliquent qu'au sein d'un campus. En effet, de par la conception de SMLTP, chaque campus possède un plan de contrôle indépendant et le backbone possède son propre plan de contrôle. Néanmoins,

ils fonctionnent tous suivant le même principe. Il en résulte que, pour chaque plan de contrôle, la convergence est au maximum de 30 secondes.

Si un nœud tombe en panne, le temps de convergence maximum est toujours de 30 secondes, le temps de deux LSPs. En effet si un nœud E tombe en panne juste après avoir envoyé son LSP, alors les autres nœuds continueront à voir E pendant 15 secondes au minimum. Dans le pire des cas, un nœud F rafraîchit sa base de données de LSPs 14 secondes après avoir reçu le LSP de E. En fait F va réactualiser sa base de données juste avant que le LSP du nœud E n'expire. F continue donc de voir E pendant encore 15 secondes jusqu'à la prochaine réactualisation de sa base de données de LSPs. Au total, F va voir le nœud E pendant encore au maximum 30 secondes après que celui-ci soit tombé en panne, en raison des 15 secondes de temps de vie du LSP et des 15 secondes nécessaires à la réactualisation de la base de données des LSPs.

3.4.1.6.4 Étude du plan de données

Pour étudier le plan de données de SMLTP, nous nous sommes focalisés sur l'étude de la latence et du débit de SMLTP et l'avons comparée aux protocole Ethernet et TRILL. Pour chacune des expériences, nous avons réalisé cent fois cinquante-cinq mesures à une seconde d'intervalle, et nous présentons la moyenne de ces résultats ainsi que l'écart type. Concernant l'étude du débit, nous avons utilisé une MTU de 1500 octets pour les trois solutions. Lors de ces expériences, la VM source est hébergée au sein du serveur S5 et la VM destination est hébergée successivement dans les serveurs S1, S3, S2 et S4 pour les tests de latence, et dans les serveurs S1, S2 et S4 pour les tests de débit.

La Figure 3.19 présente les résultats que nous avons obtenus lors des mesures de latence. La première constatation pouvant être émise est que la latence est corrélée au nombre de sauts sur le chemin. Plus celui-ci est important et plus la latence augmente. Ensuite, on peut constater que, quelle que soit la longueur du chemin, la solution SMLTP a toujours une latence supérieure à celles des solutions Ethernet et TRILL qui ont des résultats identiques. Lorsque le chemin reste au sein d'un campus, la latence de la solution SMLTP est très proche de celle de TRILL ou de celle d'Ethernet car, au sein d'un campus, le paquet ne subit pas de modification. Cependant, on voit que, lorsque le chemin passe par le backbone, la latence de SMLTP est supérieure à celles de TRILL et Ethernet. Ces résultats étaient attendus car, contrairement à Ethernet et TRILL, SMLTP doit faire transiter le message entre le campus et le backbone et vice versa. Cette transition implique un temps de traitement plus long du paquet. Il faut, en effet, réaliser une copie du paquet ainsi qu'une modification de l'en-tête TRILL du paquet et vérifier, lors de la réception du paquet, la correspondance des niveaux. La transition du paquet ne peut donc être réalisée sans impacter la latence. Néanmoins, la dégradation de la latence reste minime car elle est d'environ 1% par rapport à TRILL.

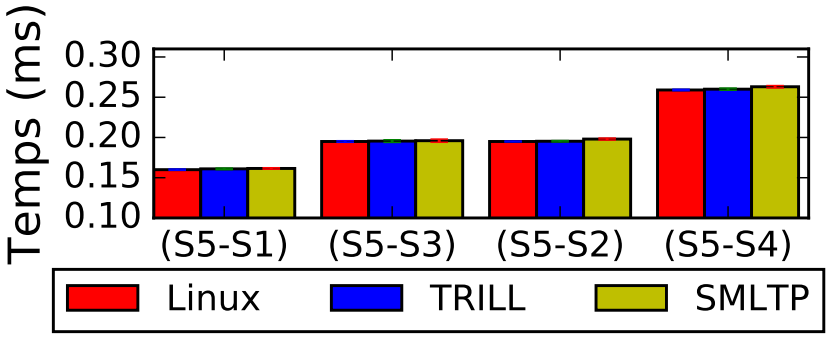


FIGURE 3.19 – Comparaison de la latence de SMLTP avec Ethernet et TRILL en fonction du chemin.

L'étude du débit de SMLTP a aussi été réalisée en comparaison avec les débits de TRILL et d'Ethernet. La Figure 3.20 présente les résultats que nous avons obtenus. On y voit clairement qu'il y a une différence entre les résultats pour les chemins intra-campus (S5-S1) et les chemins inter-campus (S5-S2 et S5-S4). Ici aussi, la transition du campus vers le backbone, et vice versa, impacte les performances de SMLTP. Ceci s'explique par le fait que, lors de cette transition, le BRB doit vérifier s'il connaît le nickname associé à l'adresse MAC destination du paquet et, ensuite, modifier l'en-tête SMLTP en changeant les deux nicknames. Ces opérations nécessitent un temps de traitement plus long pour les paquets, impliquant que l'on traite moins de paquets par unité de temps. Cependant, le débit de SMLTP est seulement inférieur de 3% à celui de TRILL. L'écart de débit entre Ethernet et TRILL s'explique en raison de ce que l'en-tête des paquets TRILL est plus grand de 64 bits que celui des paquets Ethernet, ce qui réduit la taille du payload pour les données.

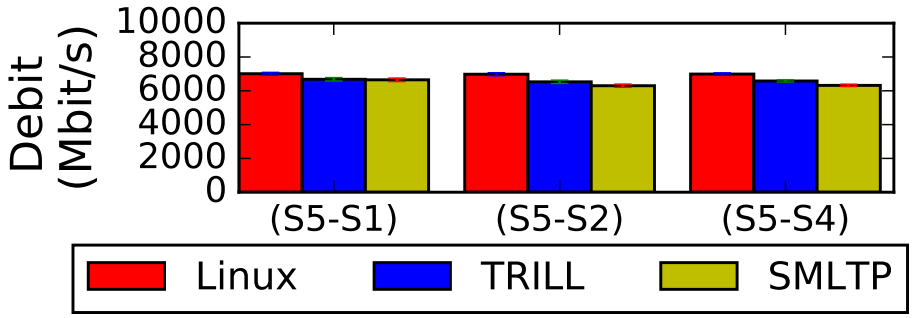


FIGURE 3.20 – Comparaison du débit de SMLTP avec Ethernet et TRILL en fonction du chemin.

3.4.1.7 Limites

SMLTP possède une contrainte concernant la transition des messages entre les niveaux. Cette contrainte est que tout le trafic inter-campus doit passer par un seul BRB qui devient

l'unique gateway du campus. Ceci est dû à la construction de l'arbre de distribution du backbone qui ne va retenir qu'un seul lien entre deux campus, comme présenté dans la Section 3.4.1.3.3. De ce fait, les messages provenant du backbone arrivent toujours sur le même BRB du campus qui sera le seul à faire transiter les paquets à l'intérieur du campus et, donc, le seul BRB utilisé par les communications inter-campus. Cette limite permet d'éviter la duplication, en autant d'exemplaires que de BRBs dans le campus, d'un message changeant de niveau, mais empêche aussi le partage de charge et l'utilisation du multi chemins. De plus, elle crée un unique point de panne qui implique la déconnection de l'ensemble du campus du backbone pendant au moins quinze secondes s'il tombe en panne.

3.4.2 Multi Level TRILL Protocol (MLTP)

3.4.2.1 Présentation

Le protocole MLTP [12] est la deuxième étape de notre première contribution et consiste en l'amélioration de SMLTP. En effet, l'objectif de MLTP est de supprimer la limite de SMLTP (Section 3.4.1.7) ; à savoir l'unique BRB de transition entre le campus et le backbone. Pour cela, nous avons conçu un nouvel équipement logique appelé la Pseudo Gateway (PG) (Section 3.4.2.2) et nous avons dû modifier les messages envoyés par les BRBs au sein du plan de contrôle (Section 3.4.2.3), ainsi que les processus de gestion de paquets des BRBs au niveau du plan de données (Section 3.4.2.4).

3.4.2.2 Nouvel équipement : La Pseudo Gateway

La Pseudo Gateway (PG) est un équipement logique qui regroupe tous les BRBs d'un campus. La Figure 3.21 nous montre l'évolution de la topologie présentée dans la Figure 3.8 avec l'ajout en rouge des Pseudos Gateways. On peut voir que chacun des campus ne possède qu'une seule PG. Cette PG devient l'unique "nœud" de transition entre le campus et le backbone et a pour rôle de n'exposer qu'une seule gateway aux RBridges du campus et aux autres campus. De cette façon, les RBridges du campus utilisent toujours le BRB le plus proche de leur position comme gateway pour leurs paquets unicast et multicast. Cependant, il est nécessaire que tous les BRBs d'un campus soit reliés entre eux afin de pouvoir former la PG.

La PG forme donc un réseau au sein du campus que seuls les BRBs doivent connaître. Pour isoler ce réseau, nous avons défini un troisième niveau dans la hiérarchie de SMLTP que nous avons attribué au réseau de la PG. Nous avons ensuite modifié le plan de contrôle de SMLTP pour tenir compte de ce nouveau niveau.

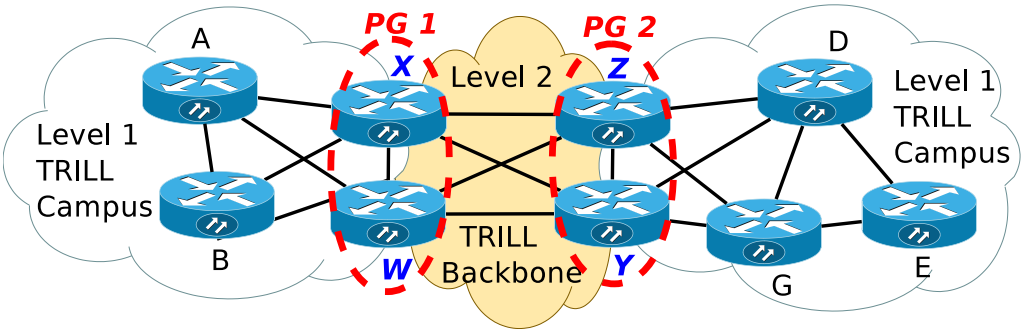


FIGURE 3.21 – Détail des Pseudo Gateways

3.4.2.3 Conception du plan de contrôle de MLTP

Le plan de contrôle de MLTP diffère de celui de SMLTP pour prendre en compte le nouvel équipement logique qu'est la PG et le nouveau réseau qu'elle forme au sein du campus entre les BRBs. Ce réseau ne doit pas être vu par les RBridges du campus, ni par les nœuds du backbone. Nous avons donc modifié les messages de contrôle des BRBs pour que ceux-ci masquent le réseau de la PG aux autres nœuds.

3.4.2.3.1 Gestion des nicknames

Nous avons conservé la séparation des nicknames en deux groupes présentés dans SMLTP. Les nicknames de niveau 1 sont toujours attribués aux RBridges au sein du campus et sont réutilisables au sein de tous les campus, par opposition aux nicknames de niveau 2, qui doivent être uniques dans le réseau. Comme la PG est visible dans le backbone et dans le campus, elle doit posséder un nickname qui est unique au sein du réseau et, donc, nous attribuons aux PGs des nicknames de niveau 2, comme pour les BRBs. Si les nicknames des PGs n'étaient pas uniques, nous pourrions avoir le même cas de conflit de nicknames que dans la Section 3.4.1.3.1. La seule différence étant qu'au lieu d'avoir un conflit entre un RBridge et un BRB, le conflit s'opèrerait entre un RBridge et une PG.

3.4.2.3.2 Modification du plan de contrôle pour utiliser la Pseudo Gateway

La Pseudo Gateway regroupe tous les BRBs d'un campus dans le but de n'exposer qu'une seule gateway à son campus et au backbone. De ce fait, nous avons un nouveau niveau entre le campus et le backbone qui réalise la jonction du niveau 1 au niveau 2. Ce nouveau niveau de type 1&2 doit permettre à un paquet de niveau 2 entrant par un BRB de sortir par un autre BRB pour joindre le campus suivant sans être réellement entré dans le campus. Par exemple, dans la Figure 3.22, la PG 3 peut communiquer avec la PG 2 via la PG 1. La PG 3 n'a pas à savoir que, pour communiquer avec la PG 2, son trafic doit emprunter un lien inter-BRB au sein de la PG 1.

Dans ce but, nous avons, à l'aide des messages de contrôle ("hello" et LSPs) des deux niveaux, mis en place un arbre de diffusion de niveau 1&2 spécifique à la PG. Cependant,

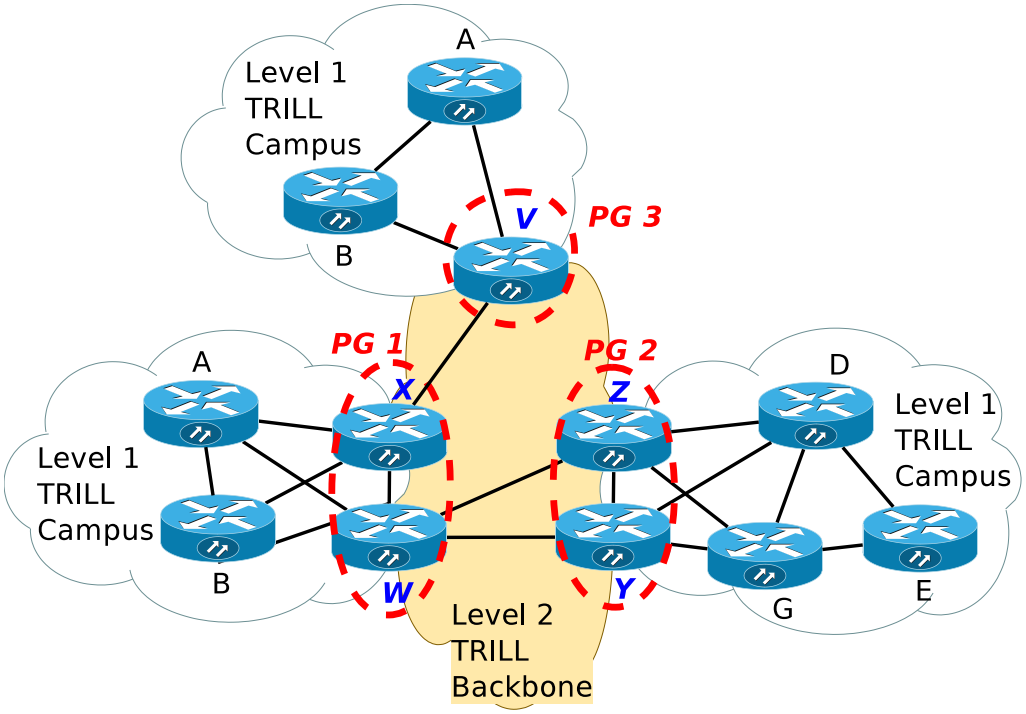


FIGURE 3.22 – Interconnexion Pseudo Gateways

	No. of Octets
INTRADOMAIN ROUTEING PROTOCOL DISCRIMINATOR	1
LENGTH INDICATOR	1
VERSION/PROTOCOL ID EXT	1
RESERVED	1
R R R TYPE	1
VERSION	1
ECO	1
USER ECO	1
RESERVED/CIRCUIT TYPE	1
SOURCE ID	6
HOLDING TIMER	2
PDU LENGTH	2
RES PRIORITY	1
LAN ID	7
VARIABLE LENGTH FIELDS	VARIABLE

FIGURE 3.23 – Détails d'un message "Hello"

	No. of Octets
INTRADOMAIN ROUTEING PROTOCOL DISCRIMINATOR	1
LENGTH INDICATOR	1
VERSION/PROTOCOL ID EXT	1
RESERVED	1
R R R TYPE	1
VERSION	1
ECO	1
USER ECO	1
PDU LENGTH	2
REMAINING LIFETIME	2
LSP ID	8
SEQUENCE NUMBER	4
CHECKSUM	2
P ATT HIPPITY IS TYPE	1
VARIABLE LENGTH FIELDS	VARIABLE

FIGURE 3.24 – Détails d'un LSP

il nous a fallu modifier les LSPs des BRBs. Pour que la PG soit visible et perçue comme l'unique gateway du campus, il est nécessaire que les LSPs émis par les BRBs de cette PG n'utilisent que le nickname et le sysid de la PG. Il en résulte des collisions entre les BRBs d'un même campus qui utilisent le même nickname. Pour parer à ce problème, il est nécessaire que les BRBs connaissent leurs vrais nicknames. À cette fin, nous avons ajouté une option aux messages "Hello" et aux LSPs afin que les BRBs puissent connaître les vraies informations des autres BRBs.

Les messages "Hello", présentés dans la Figure 3.23, utilisés pour découvrir les nœuds adjacents, utilisent le "System Identifier" (sysid) du nœud. Or, chaque BRB remplace son

sysid avec celui de la PG pour que seul la PG ne soit connue des autres nœuds. Nous avons donc ajouté une option de type TLV (Type, Length, Value) comprenant le sysid réel du nœud dans la partie variable des messages "Hello" lorsque ceux-ci sont envoyés par un BRB. Cependant, seuls les BRBs d'un même campus vont récupérer cette information et ainsi éviter des problèmes de découverte de voisinage.

Les LSPs (Figure 3.24) aussi ont été modifiés avec l'ajout d'une option, de type TLV, inclue dans la partie variable du LSP. Cette option contient le vrai nickname du BRB. En effet, comme pour les "Hello", les BRBs, dans le but de n'exposer qu'une seule gateway, utilisent dans leurs LSPs le nickname de la PG. Ceci crée des collisions entre les BRBs d'un campus car le même nickname est utilisé par plusieurs nœuds. Ce problème a été résolu par l'ajout de cette option qui, comme pour celle des "Hello", n'est traitée que par les autres BRBs du campus.

Chaque BRB membre de la PG sait ainsi quels campus sont accessibles par chacun des membres de la PG. Le fait de n'exposer qu'une seule gateway par campus sur le backbone permet ainsi qu'une PG serve de nœud de transit entre deux campus. Le réseau backbone est ainsi unique et contigüe, respectant la contrainte imposée par le protocole IS-IS.

3.4.2.4 Conception du plan de données de MLTP

Les différences du plan de données de MLTP par rapport à SMLTP se situent au niveau des processus d'envoi et de réception des paquets. Dans les sections suivantes, nous décrivons les modifications apportées à ces processus.

3.4.2.4.1 Processus d'envoi

Le processus d'envoi est présenté dans la Figure 3.25 et on a mis en avant les nouvelles étapes en rouge ainsi que les étapes ajoutées par SMLTP en vert. On peut remarquer que les modifications ne concernent que les BRBs. En effet, l'ajout de la PG n'impacte que les BRBs de par la conception de la PG.

Le processus d'envoi de SMLTP est modifié en deux endroits, lors du changement de l'en-tête du paquet avant l'envoi sur les arbres de distribution du campus et du backbone. Ces deux modifications changent le nickname source du paquet par celui de la PG ce qui permet de n'afficher que la PG aux autres nœuds. Cependant, pour maintenir les BRBs cachés en n'affichant que le nickname de la PG, nous sommes obligés de propager un autre paquet multicast au sein du réseau de la PG. En effet, si MLTP ne possédait pas cette distribution au sein de la PG, alors, les autres BRBs du campus recevraient un message avec, comme source, la PG dont ils font partie. Ils seraient alors dans l'incapacité de savoir quel nœud est la source du message.

La propagation du paquet multicast de niveau 3 au sein de la PG implique une nouvelle

"branche" dans le processus d'envoi. Pour réaliser cette propagation au sein de la PG, le BRB vérifie dans un premier temps s'il existe un arbre de distribution pour la PG, puis il vérifie que le prochain nœud sur le chemin est bien de niveau 1&2. Une fois ces deux vérifications effectuées, le BRB copie le paquet et modifie l'en-tête de la copie. Lors de cette modification, le BRB remplace le niveau avec le niveau sur 1&2, il remplace aussi le nickname destination avec celui du DTroot de la PG. Cependant, à l'opposé des modifications réalisées pour les autres niveaux (campus et backbone), le BRB met son propre nickname comme nickname source ce qui permet aux autres BRBs du campus de connaître la vraie source du paquet.

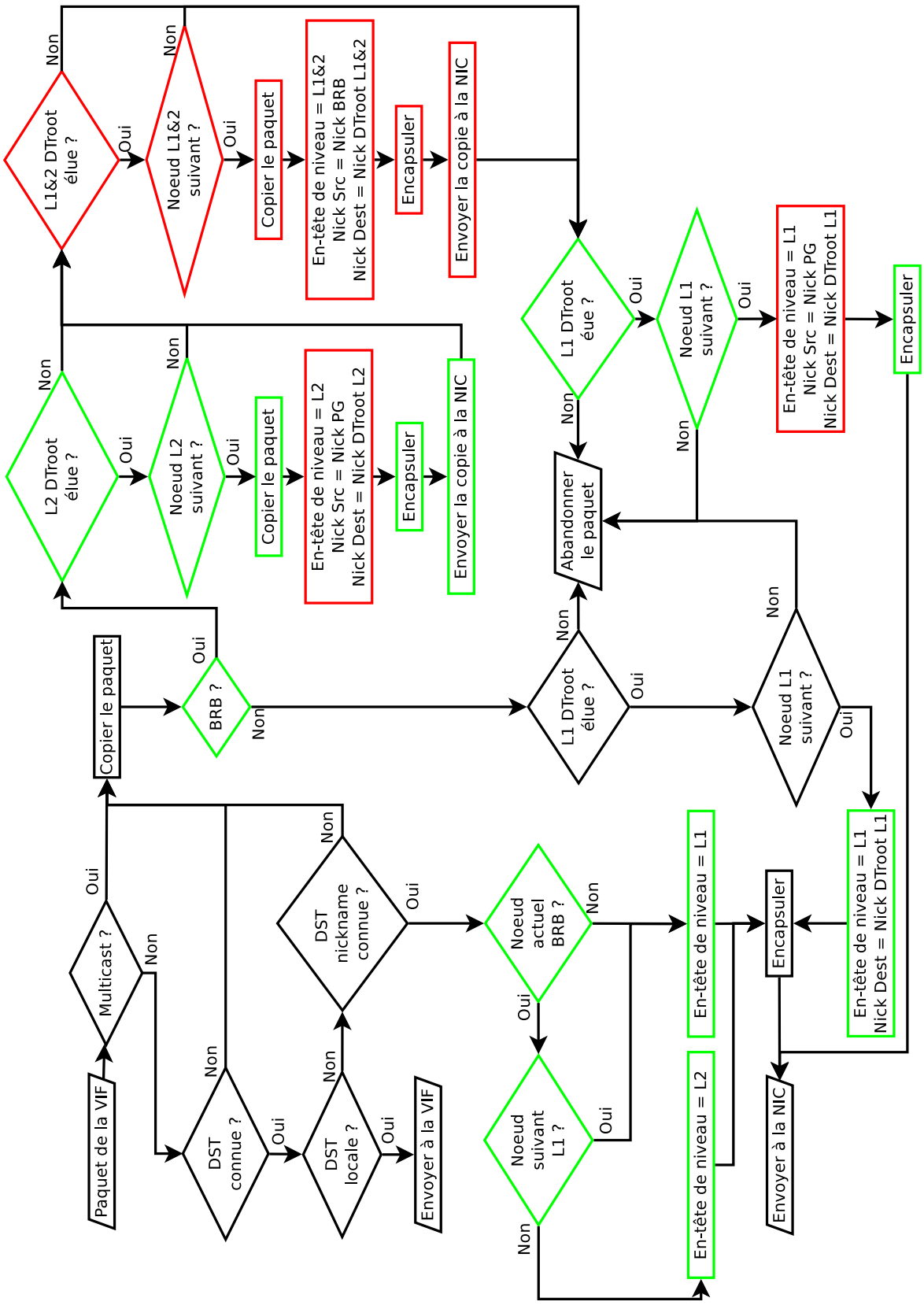


FIGURE 3.25 – Processus d'envoi d'un message dans MLTP

3.4.2.4.2 Processus de réception unicast

Pour la réception de paquet unicast, on remarque, là aussi, que les deux modifications, en rouge sur la Figure 3.26, nécessaires pour MLTP ne concernent que les BRBs. La première modification impacte la vérification du nickname destination qui ne vérifie plus si le nickname destination correspond au nickname du BRB mais de la PG. En effet, le nickname du BRB n'est plus connu par les RBridges qui utilisent le nickname de la PG. Or, lorsqu'un BRB reçoit un paquet destiné à la PG il doit le traiter comme s'il lui était destiné. La seconde modification concerne l'étape du changement de l'en-tête lorsque le paquet transite d'un niveau à l'autre. Pour masquer les BRBs et n'afficher que la PG, le BRB va, au lieu de mettre son nickname comme nickname source du paquet, mettre le nickname de la PG. De cette façon, seule la PG est connue par les autres campus et les RBridges du campus.

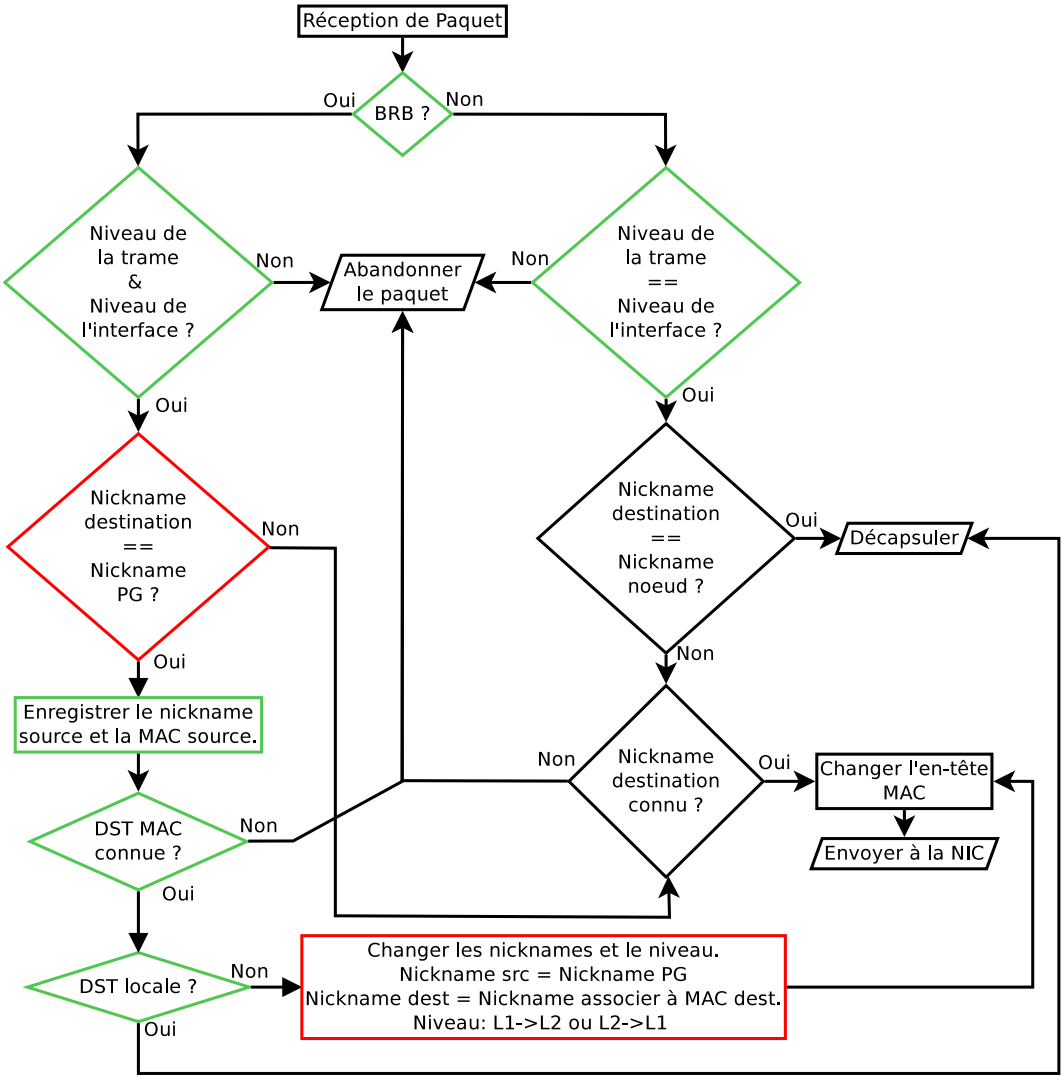


FIGURE 3.26 – Processus de réception d'un message unicast dans MLTP

3.4.2.4.3 Processus de réception multicast

Le processus de réception multicast de MLTP doit prendre en compte les paquets de

niveau 1&2 comme étant des paquets pour le réseau de la PG et non plus comme des paquets multicast à ne pas renvoyer sur le backbone comme c'est le cas dans SMLTP. Nous avons donc ajouté les étapes nécessaires à la nouvelle gestion de ces paquets de niveau 1&2, représentées en rouge dans la Figure 3.27. Pour cela, un BRB vérifie si le paquet est de niveau 1&2 et qu'il ne possède pas, comme nickname source, le nickname de la PG. Si ce n'est pas le cas, alors, le paquet est rejeté car, avec le nickname de la PG comme source, le BRB ne sait pas d'où provient le paquet. Ensuite, le BRB vérifie que le nickname destination correspond au DTroot de la PG qu'il connaît, puis il enregistre le nickname source et les adresses MAC avant de réaliser une copie du paquet qu'il décapsule et traite localement. Finalement, le BRB propage le paquet sur l'arbre de distribution de la PG.

Nous avons aussi modifié la gestion des paquets multicast de niveau 1 et de niveau 2. En effet, nous avons ajouté des étapes qui contrôlent le nickname source du paquet pour éviter de le renvoyer dans le backbone ou dans le camp. Lorsqu'un BRB reçoit un paquet multicast qu'il doit faire transiter du campus vers le backbone, ou vice versa, il ne va plus mettre le niveau 1&2 au nouveau paquet mais simplement, soit le niveau 1 pour le campus, soit le niveau 2 pour le backbone. Par contre, le nickname source du nouveau paquet correspond à celui de la PG et c'est grâce à ce nickname que les autres BRBs savent qu'ils ne doivent pas faire à nouveau transiter le paquet d'un niveau à l'autre.

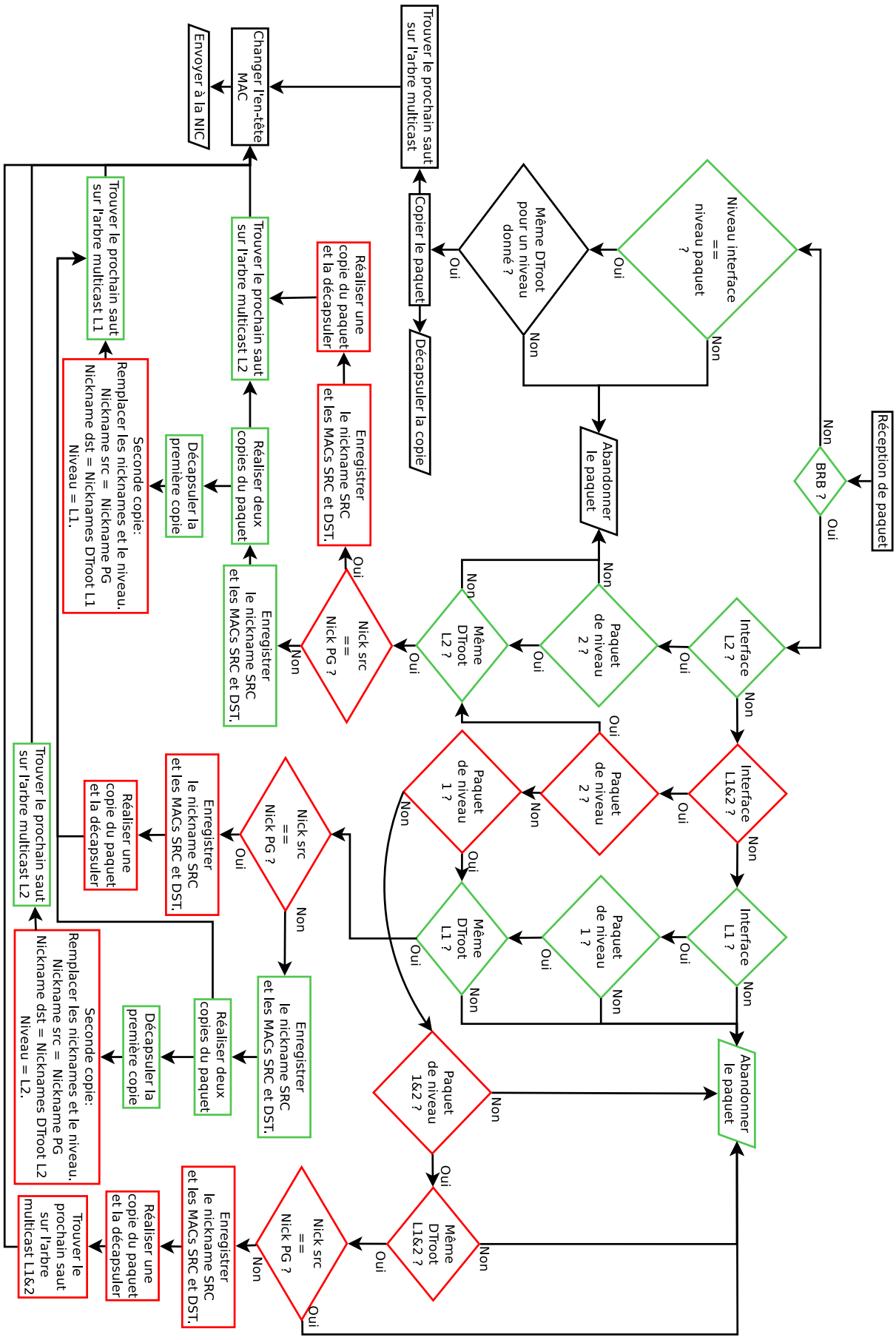


FIGURE 3.27 – Processus de réception d'un message multicast dans MLTP

3.4.2.5 Rétro-compatibilité

Concernant le plan de contrôle de MLTP, nous avons modifié les LSPs et les "Hellos" des BRBs en leur ajoutant des options contenant des informations topologiques nécessaires pour les autres BRBs de la PG. Cependant, ces options ne sont destinées que pour les BRBs, et les RBridges du campus doivent les ignorer. C'est pour cela que nous avons défini de nouvelles options de type TLV que seuls les BRBs connaissent et peuvent traiter. Les RBridges, ne connaissant pas ces options, vont les ignorer et traiter les "Hellos" et les LSPs comme défini dans TRILL. Nous conservons donc la rétro-compatibilité avec TRILL au sein des campus.

Pour le plan de données, nous avons dû modifier les processus d'envoi et de réception des paquets seulement pour les BRBs. En effet, les RBridges du campus ne subissent aucune modification de leur processus d'envoi et de réception par rapport à SMLTP. Or SMLTP est compatible avec TRILL, donc MLTP l'est aussi.

Le seul problème pouvant survenir est un problème de commutation/routage dans le cas où un RBridge au sein d'un campus, utilise un nickname de niveau 2 déjà utilisé par un BRB ou une PG. Pour résoudre ce problème, les BRBs et les PGs associent le nickname en conflit à la PG ou le BRB du fait qu'ils possèdent une priorité supérieure au RBridge TRILL. Néanmoins, le RBridge TRILL ne peut plus communiquer qu'au sein de son site, et, pour résoudre cela, il faut manuellement reconfigurer le RBridge TRILL en lui donnant un nickname de niveau 1.

3.4.2.6 Performances

L'étude des performances de MLTP consiste à étudier sa latence ainsi que son débit et à les comparer aux solutions Ethernet et SMLTP. Nous analysons aussi le temps de convergence du plan de contrôle de MLTP. Les expériences réalisées pour obtenir les résultats présentés ci-après sont identiques aux expériences réalisées pour obtenir les résultats de SMLTP dans la Section 3.4.1.6.

3.4.2.6.1 Implémentation de la solution MLTP

MLTP est une évolution de SMLTP et nous avons utilisé l'implémentation précédemment réalisée de SMLTP (Section 3.4.1.6.1) pour développer MLTP.

3.4.2.6.2 Environnement de test

Cette fois encore, nous avons conservé la même topologie physique que celle utilisée dans l'étude des performances de SMLTP. Cependant, avec l'ajout de la PG, la topologie logique diffère de la topologie physique. La Figure 3.28 montre la nouvelle topologie logique obtenue et les Figures 3.29, 3.30 et 3.31 montrent l'évolution des arbres de distribution

des deux campus et du backbone. Dans la topologie logique, les BRBs sont masqués au sein de la PG et il en est de même dans les arbres de distribution.

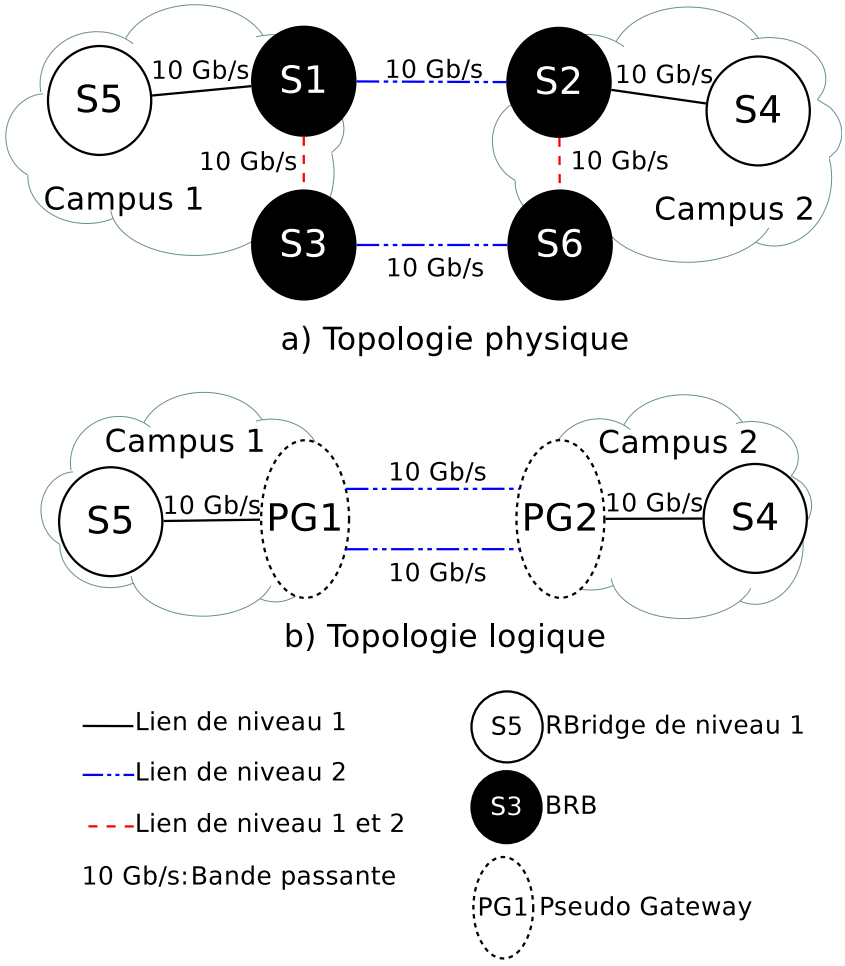


FIGURE 3.28 – Topologie physique de la plate-forme de test

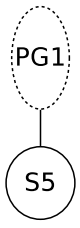


FIGURE 3.29 – Arbre de distribution du campus 1

FIGURE 3.30 – Arbre de distribution du campus 2

FIGURE 3.31 – Arbre de distribution du backbone

3.4.2.6.3 Plan de contrôle

L'étude de la convergence du plan de contrôle de MLTP a été réalisée suivant la même procédure que lors de l'étude de la convergence du plan de contrôle de SMLTP (Section 3.4.1.6). Nous avons mesuré le temps nécessaire pour qu'un nœud nouvellement ajouté soit reconnu par tous les nœuds partageant ce plan de contrôle. Dans la Figure 3.32, on

peut voir que le temps de convergence du plan de contrôle d'un campus est, au maximum, de 30 secondes, l'équivalent du temps nécessaire à l'envoi de deux LSPs. Comme pour SMLTP, seul le temps de rafraichissement des LSPs, 15 secondes entre chaque LSPs, impacte le temps de convergence du plan de contrôle. Ce temps de convergence maximum est identique dans chaque campus et au sein du backbone. Concernant l'ajout d'un BRB au sein d'une pseudo gateway, le temps de convergence est aussi de 15 secondes au sein du réseau de la pseudo gateway et de 15 secondes au sein du campus. Cependant, ce sont les mêmes 15 secondes car la découverte des BRBs au sein de la pseudo gateway utilise les LSPs de niveau 1, utilisés pour la découverte du campus, auxquels une option pour les BRBs a été ajoutée.

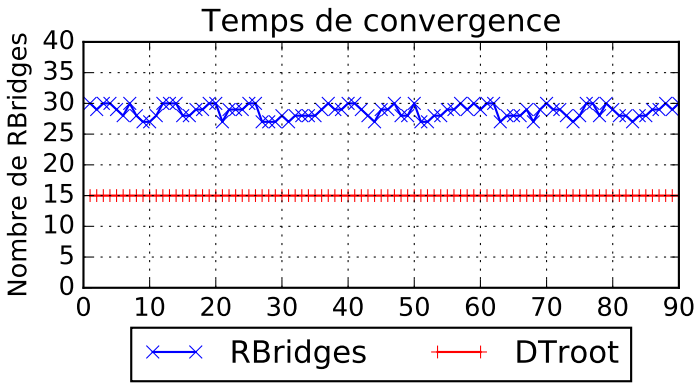


FIGURE 3.32 – Évolution du temps de convergence du plan de contrôle d'un campus de MLTP en fonction du nombre de RBridges.

3.4.2.6.4 Plan de données

Dans les deux Figures 3.33 et 3.34, on peut remarquer que les résultats de MLTP sont identiques à ceux de SMLTP, tant au niveau de la latence que du débit. Ceci s'explique par le fait que les seules modifications que nous avons réalisées au processus de réception des messages unicast n'ajoutent pas d'étapes mais modifient des étapes existantes. En effet, le BRB, suite à la première modification, ne vérifie plus si le nickname destination est le sien mais si c'est celui de la PG et la seconde modification implique de remplacer le nickname source avec celui de la PG au lieu du nickname du BRB. Ces deux modifications n'impactent, ni le temps de traitement des paquets, ni la taille des en-têtes et il était donc attendu que MLTP réalise les mêmes performances que SMLTP.

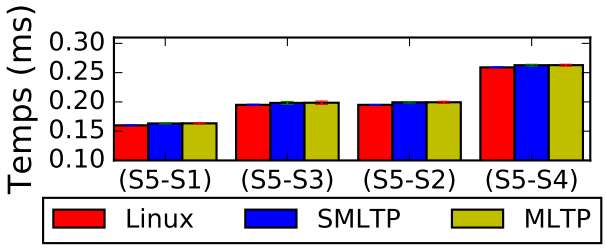


FIGURE 3.33 – Comparaison de la latence de MLTP avec Ethernet et SMLTP en fonction du chemin.

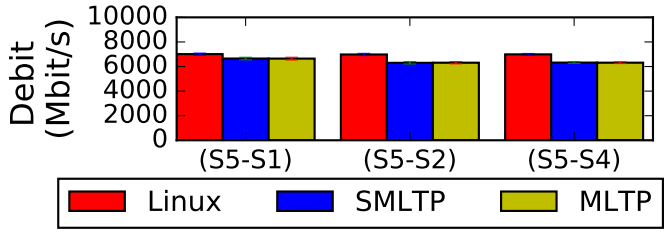


FIGURE 3.34 – Comparaison du débit de MLTP avec Ethernet et SMLTP en fonction du chemin

3.5 Conclusion

Grâce à l'ajout de la Pseudo Gateway au sein de MLTP, nous avons modifié la limite de SMLTP, à savoir l'unicité de la gateway du campus au niveau physique. Nous l'avons remplacée par une unique gateway logique composée de plusieurs BRBs. MLTP peut utiliser l'ensemble des BRBs pour réaliser des communications inter-campus et ainsi profiter des capacités multi-chemins de TRILL. Néanmoins, MLTP possède encore un inconvénient qui est l'envoi systématique des messages de broadcast émis au sein du campus vers le backbone et vers les autres campus même si toutes les destinations se trouvent au sein du campus ou de quelques campus.

Chapitre 4

Isolation des flux inter et intra data centers

Sommaire

4.1 Introduction

L'un des intérêts du cloud public est de proposer des services à des coûts moindres que si l'on utilisait un cloud privé. En effet, le fournisseur de cloud public fournit une infrastructure qu'il va partager entre tous ses clients et ainsi optimiser l'utilisation de son infrastructure. Il a donc tout intérêt à avoir un grand nombre de clients utilisant son infrastructure et il peut donc proposer des tarifs attractifs à ses clients. Dans le même temps, les utilisateurs font des économies sur plusieurs points. Ils n'ont pas à acheter les équipements informatiques réseaux nécessaires à la mise en place de l'infrastructure, ni à mettre en place cette infrastructure. Ils économisent aussi les coûts d'entretien et de mise à niveau de l'infrastructure. De plus, ils n'ont plus à gérer les coûts énergétiques liés à l'infrastructure et à son refroidissement ainsi que les coûts des locaux. Un autre point positif pour l'entreprise est le fait que les dépenses liées au système informatique sont connues d'avance. En effet, avec des offres de type abonnement ou d'achat de crédit pour l'utilisation des VMs, l'entreprise peut établir un budget qui ne souffre pas d'augmentations non souhaitées dû à des remplacements de machines défectueuses ou tout autre problème survenu si elle possédait sa propre infrastructure.

Néanmoins, le fait de partager une infrastructure commune entre différents clients pose le problème de confidentialité des informations de chaque utilisateur. Pour résoudre ce problème il est possible d'isoler les clients à l'aide de réseaux virtuels en utilisant par exemple un VLAN par client. Cependant, pour répondre à la demande croissante de service des utilisateurs, les équipements cloud sont de plus en plus performants et la taille des data centers augmente. Il en résulte que de plus en plus de clients (utilisateurs) utilisent les mêmes équipements et les mêmes réseaux au sein du cloud public. Se pose alors le problème du passage à l'échelle de la solution d'isolation. Comme détaillée dans l'article "A Survey of network isolation solutions for multi-tenant data centers" [13], la solution des VLANs ne passe pas à l'échelle de par sa limitation à 4096 VLANs différents. Il nous faut donc trouver une nouvelle solution d'isolation capable de gérer un nombre important de clients au sein d'un cloud. Toujours dans l'article [13], nous avons réalisé une comparaison de différentes solutions d'isolation adaptées aux infrastructures de type cloud. Cette comparaison a été effectuée sur plusieurs points tels que leurs complexités, l'overhead qu'elles produisent ainsi que leurs capacités. Chacune des solutions étudiées possède ses avantages et ses inconvénients. Néanmoins, la solution VNT, détaillée dans la Section 4.3.1, semblait la plus intéressante pour nos travaux et nous nous sommes donc focalisés sur son implémentation au sein de MLTP.

4.2 Problématique

Comme introduit dans la Section précédente (4.1), l'utilisation de solutions telles que les VLANs n'est pas suffisante pour isoler l'ensemble des clients. Nous avons donc décidé

d'implémenter la solution VNT au sein de MLTP. Cette solution permet d'obtenir plus de 16 millions de réseaux privés et donc d'accommoder un plus grand nombre d' utilisateurs grâce aux identifiants VNI codés sur 24 bits. Cependant, pour intégrer la solution VNT à MLTP, il est nécessaire de modifier le plan de contrôle de ce dernier. En effet, VNT nécessite un automate de décision pour établir si un lien fait, ou non, partie d'un réseau virtuel. Il faut donc ajouter cet automate au plan de contrôle de MLTP.

Le plan de données de MLTP doit aussi être modifié avec l'ajout dans les processus d'envoi et de réception de la gestion du champ optionnel au sein de l'en-tête TRILL contenant le VNI. Ce VNI indique l'appartenance d'un paquet à un client précis, or, si le nœud ne gère pas cette option, alors le paquet pourrait sortir du réseau virtuel associé à ce VNI.

Une autre problématique consiste en l'optimisation de l'utilisation des liens du backbone. Ces liens, interconnectant les différents campus, n'appartiennent pas forcément à l'entité administrant l'infrastructure cloud. En général, ces liens sont loués et leurs coûts augmentent en corrélation avec leurs utilisations. C'est pourquoi, en plus des bénéfices habituels (latence, débit) lors de la réduction de la charge des liens, l'optimisation de l'utilisation de ces liens est importante. Pour réaliser cette optimisation, nous allons faire en sorte d'éviter l'envoi de paquets qui seront rejetés lors de leur arrivée à la pseudo gateway du campus distant. En effet, au sein de VNT, c'est lors de la réception d'un paquet que le VNI est vérifié et non lors de son envoi. Les paquets sont donc envoyés sur les liens du backbone quel que soit leurs VNI. Or, lors de la réception de ces paquets par le campus distant, si le VNI n'est pas supporté par la PG distante, alors les paquets sont rejetés. On a donc une utilisation des liens du backbone pour des paquets inutiles.

Pour réaliser la vérification des VNIs lors de l'envoi, il faut connaître la liste des VNIs supportés dans les différents campus distants. Il nous faut donc stocker ces VNIs au sein de la PG du campus et par extension au sein des BRBs. Sachant que l'on peut avoir plus de 16 millions de VNIs différents, et que l'on doit enregistrer le fait qu'ils soient supportés, ou non, pour chaque campus du réseau, la quantité d'informations à conserver dans chaque BRB peut devenir conséquente. Pour chaque campus, on doit enregistrer $2^{24} = 16777216$ VNIs, chacun codé sur 24 bits, la résultante étant une taille d'approximativement 402 Mbits. Or, si l'on a quatre campus, l'espace nécessaire pour enregistrer l'utilisation des VNIs dans les trois autres campus dépasse le Gigabit, ce qui est trop important. Nous devons donc trouver une structure permettant d'enregistrer ces informations tout en minimisant l'espace mémoire utilisé et en ne dégradant pas le temps de traitement du paquet par une recherche de VNI peu efficace.

4.3 Contributions : MLTP+VNT

Notre seconde contribution consiste à intégrer la solution VNT au sein de MLTP ainsi que de définir une solution pour le stockage des VNIs des différents campus au sein des BRBs. Nous allons commencer par présenter l'intégration de la solution VNT au sein de MLTP (Section 4.3.1) puis nous détaillerons le choix de la structure de stockage des VNIs (Section 4.3.2).

4.3.1 Intégration de VNT dans MLTP

La solution VNT, présentée dans la thèse du Docteur Ahmed Amamou [14] et présentée dans la section 2.7, implique de modifier MLTP, tant au niveau du plan de contrôle que du plan de données.

4.3.1.1 Modification du plan de contrôle de MLTP

Concernant le plan de contrôle de MLTP, nous avons dû y intégrer les éléments de VNT suivants :

— L'automate d'états pour les VNIs.

— Le processus de découverte des VNIs.

— L'algorithme de calcul des VNIs.

Dans les paragraphes suivants, nous détaillons l'intégration et le fonctionnement de chacun de ces éléments.

4.3.1.1.1 Automate d'états des VNIs

L'automate d'états pour les VNIs, présenté dans la Figure 4.1, doit être implémenté dans chaque nœud pour pouvoir délimiter le réseau privé de chaque utilisateur. En effet, en fonction de l'état dans lequel l'automate se trouve, le nœud va supporter, ou non, un VNI et donc rejeter, ou non, les paquets contenant ce VNI. Or, un VNI identifie un utilisateur et, donc, par extension, son réseau privé.

Lors du démarrage d'un nœud, son automate est positionné sur l'état "Inconnu" pour l'ensemble des VNIs. Ceci est l'état initial de l'automate de chaque nœud du réseau. La seconde étape pour le nœud est de vérifier s'il possède des informations sur des VNIs locaux, c'est à dire s'il a reçu des VNIs directement d'une ou plusieurs VMs instanciées localement. Si c'est le cas, alors l'automate passe à l'état "Découvert" pour ces VNIs. À partir de ce moment, ces nouveaux VNIs sont dans l'état "Supporté" au sein du nœud et l'automate va accepter les paquets utilisant ces VNIs ainsi que les faire transiter vers les VMs et vers les autres nœuds du réseau. Lorsqu'un VNI n'est plus utilisé par une VM locale et qu'il se trouve dans l'état "Supporté", celui-ci est alors renvoyé à l'état "Inconnu" et les paquets utilisant ce VNI ne sont plus acceptés par le nœud. Dans le cas où le

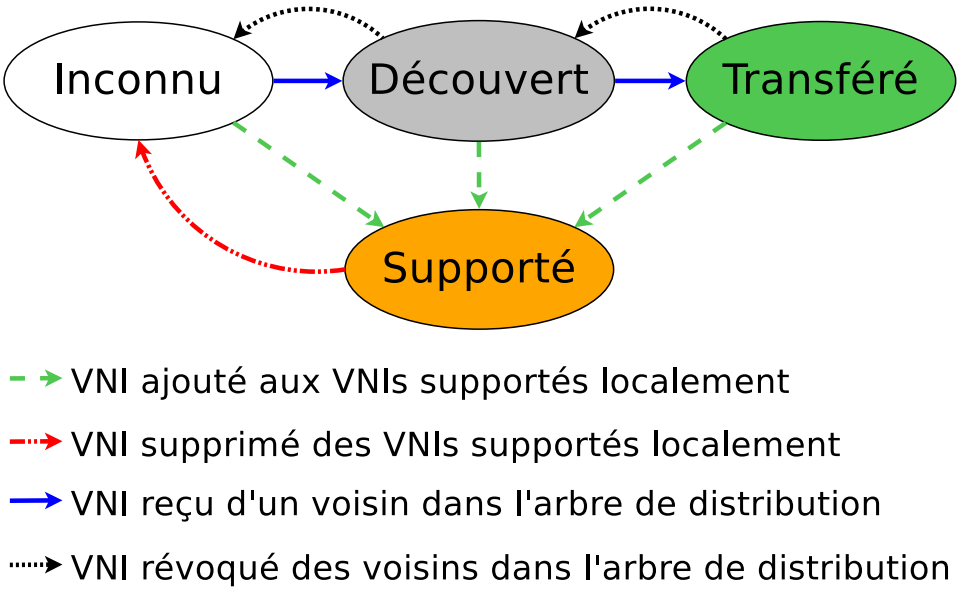


FIGURE 4.1 – Automate d'états pour les VNIs

nœud découvre un VNI via un seul de ses nœuds voisins, il met alors le VNI dans l'état "Découvert". Cependant, dans cet état, le nœud n'accepte pas encore les paquets utilisant ce VNI car il n'a aucune destination pour ces paquets sinon la même que leur origine. Les VNIs dans l'état "Inconnu" et l'état "Découvert" ne sont pas annoncés par le nœud à ses voisins car la seule action qu'il pourrait réaliser serait de rejeter les paquets avec ces VNIs. Si le nœud découvre un deuxième nœud, ou plus, utilisant un VNI qu'il possède à l'état "Découvert", il va alors changer l'état du VNI à "Transféré". Dans cette situation, le nœud va jouer le rôle de transit pour les paquets utilisant ce VNI, mais il ne va pas faire parvenir ces paquets aux VMs instanciées localement.

Cet automate est utilisé au sein des campus et par tous les nœuds des campus (RBridges et BRBs). Cependant, les BRBs n'utilisent cet automate que pour le trafic intra-campus. Nous avons, pour le trafic inter-campus, implémenté un autre procédé qui permet de vérifier, avant l'envoi, si la destination accepte le VNI. Ce procédé repose sur la vérification de la présence, ou non, du VNI dans une structure qui sauvegarde localement (dans chaque BRB) la liste des VNIs utilisés dans les autres campus. Le choix et le fonctionnement de la structure sont indiqués dans la Section 4.3.2.

4.3.1.1.2 Découverte des VNIs

Pour réaliser les changements d'état des VNIs dans un nœud, il est nécessaire d'avoir les informations sur les états des VNIs au sein des VMs locales au nœud et au sein des autres nœuds. Pour découvrir les VNIs utilisés par les VMs, celles-ci doivent envoyer un message indiquant les VNIs qu'elles utilisent à la fin de leur instanciation. Ce message de type broadcast va parcourir l'arbre de distribution du campus et, de cette façon, tous les nœuds du campus vont découvrir le VNI. La Figure 4.2 représente un exemple de découverte de VNI. Nous avons, dans la première phase, le campus dans son état initial

avec son arbre de distribution dont la racine est le nœud 1. Tous les nœuds sont en blanc pour l'état du VNI1 car celui-ci leur est inconnu. Dans la seconde phase, le nœud 2 instancie une VM A utilisant le VNI1 et cette VM l'annonce. Il en résulte que tous les nœuds du réseau sont maintenant passés à l'état "Découvert" pour le VNI1. Finalement, dans la troisième phase, une seconde VM (VM B) utilisant le VNI1 est instanciée dans le nœud 7. Cette VM B va elle aussi annoncer l'utilisation du VNI1 et l'on peut voir sur l'arbre multicast avec état que les nœuds sur le chemin entre les nœuds 2 et 7, c'est à dire les nœuds 8 et 1, sont maintenant dans l'état "Transféré" pour le VNI1. Cependant, par conception de MLTP, ce message restera cantonné au campus et ne sortira pas de celui-ci afin d'éviter d'utiliser les liens du backbone.

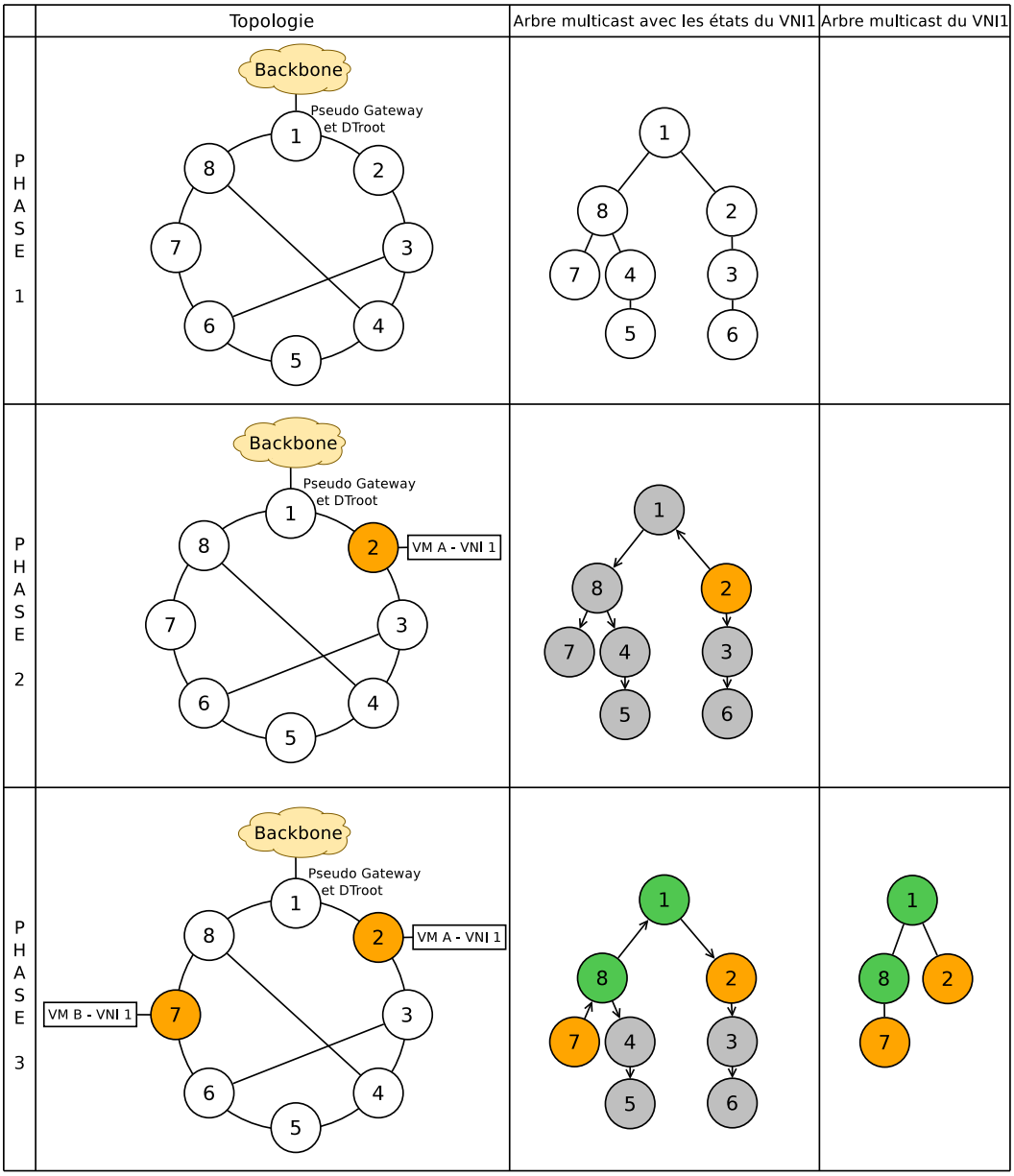


FIGURE 4.2 – Exemple de découverte d'un VNI

Toujours au sein du campus, les nœuds doivent s'échanger la liste des VNIs se trouvant dans l'état "Supporté" ou "Transféré". Ils utilisent, pour ce faire, une nouvelle option de

type TLV, ajoutée aux LSPs L1, qui contient la liste des VNIs que le nœud supporte ou fait transiter. Grâce à cet échange d'informations, les nœuds maintiennent à jour les états de chaque VNI du campus. En effet, si une VM venait à disparaître et que le VNI n'était plus utilisé, alors le nœud local à la VM n'annoncerait plus ce VNI, permettant alors aux autres nœuds de passer ce VNI à l'état "Inconnu". À l'aide de ce procédé de découverte, tous les VNIs du campus sont connus par tous les nœuds du campus, lesquels ont ainsi chaque VNI dans le bon état de l'automate.

Néanmoins, il est nécessaire de modifier ce procédé pour établir des connexions inter-campus utilisant les mêmes VNIs. Lorsqu'un paquet est reçu par un BRB, celui-ci va vérifier, dans un premier temps, si le VNI utilisé par ce paquet est utilisé dans les autres campus. Pour ce faire, il va chercher dans sa base de données locale des VNIs (détaillée dans la Section 4.3.2) quels sont les campus qui utilisent ce VNI. Or il faut, pour cela, que le BRB soit informé des VNIs utilisés dans les autres campus. Nous ajoutons donc l'option TLV contenant la liste des VNIs aux LSP L2 à la différence près que cette liste est composée non seulement des VNIs dans les états "Supporté" ou "Transféré" mais aussi des VNIs dans l'état "Découvert". En effet, l'ajout des VNIs dans l'état "Découvert" permet d'informer les autres campus de l'ensemble des VNIs utilisés dans le campus. Sans cela, nous pourrions avoir des segments d'un réseau privé utilisant le même VNI qui soit disjoint au sein de campus différents.

Nous avons fait l'hypothèse qu'un VNI est unique dans la totalité du réseau. Cette hypothèse se base sur le fait que l'on ne veut pas restreindre le réseau privé d'un client à un seul campus, ce qui lui permet de faire migrer ces VMs au sein de n'importe lequel des campus en fonction de ses besoins.

4.3.1.1.3 Algorithme de calcul des VNIs

Concernant l'algorithme de calcul des VNIs (Figure 4.3), nous avons utilisé celui défini dans [14] pour les nœuds de niveau 1 (i.e. ceux au sein d'un campus). Pour le niveau 2, l'algorithme est légèrement modifié et en particulier l'étape 3. Celle-ci limite les VNIs à annoncer dans les LSPs aux seuls VNIs dans les états "supporté" et "Transféré", or, pour informer les autres campus de la liste des VNIs utilisés au sein du campus, il est nécessaire que les LSP L2 annoncent aussi les VNIs que les BRBs ont découverts. Il en résulte que, pour la création de la liste des VNIs à annoncer dans les LSPs L2, nous ne vérifions pas leurs états et les ajoutons directement à la liste pour les annoncer.

```
while TRUE do
    /* it is unnecessary to compute VNI for DTroot as it has to support all VNI       */
    if LocalHost ≠ Dtroot then
        CreateList(HandledVni) ;
        /* step1: add locally supported VNI                                          */
        foreach Vni in SupportedVni do
            NewElement .Vni ← Vni ;
            NewElement .VniType ← supported ;
            NewElement .SeenIface ← 0 ;
            AddToList (HandledVni, NewElement) ;
        end
        /* step2: compute other known Hosts VNI                                      */
        foreach host in KnownHost do
            if host ∈ PseudoHost and PseudoHost .DR = LocalHost then
                /* DR has to support all VNI that are contained on a pseudo Host     */
                foreach Vni in host .AdvertisedVni do
                    NewElement .Vni ← Vni ;
                    NewElement .VniType ← supported ;
                    NewElement .SeenIface ← host .SeenIface ;
                    AddToList (HandledVni, NewElement) ;
                end
            end
            else
                foreach Vni in host .AdvertisedVni do
                    foreach element in HandledVni do
                        if Vni = element .Vni then
                            if element .VniType = discovered and element .SeenIface ≠ host .SeenIface then
                            element .VniType ← forwarded ;
                        end
                        else
                            NewElement .Vni ← Vni ;
                            NewElement .VniType ← discovered ;
                            NewElement .SeenIface ← host .SeenIface ;
                            AddToList (HandledVni, NewElement) ;
                        end
                    end
                end
            end
        end
        /* step3: create advertised VNI list and remove unnecessary VNI             */
        CreateList(AdvertisedVni) ;
        foreach element in HandledVni do
        if element .VniType = discovered then
        | RemoveFromList(HandledVni, element) ;
        end
        else
        | AddToList (AdvertisedVni,element .Vni) ;
        end
        Send(AdvertisedVni) ;
    end
end
```

FIGURE 4.3 – Algorithme de calcul des VNIs

4.3.1.2 Modification du plan de données de MLTP

En plus du plan de contrôle, il nous a fallu modifier le plan de données et plus précisément les processus d'envoi et de réception des paquets pour prendre en compte le champ optionnel contenant le VNI. Dans les Sections 4.3.1.2.1 et 4.3.1.2.2, nous allons détailler les modifications que nous avons réalisées sur les processus d'envoi et de réception.

4.3.1.2.1 Modification du processus d'envoi des paquets

L'intégration de la solution VNT au sein de MLTP a impliqué des changements au sein du processus d'envoi des paquets. Comme le montre la Figure 4.4, nous avons dû ajouter plusieurs étapes. Ces étapes (en rouge sur la Figure 4.4) consistent en l'ajout de l'option VNT à l'en-tête du paquet TRILL, que ce soit, ou non, pour les paquets multicast, et à inonder le paquet sur le Virtual Switch (VS). Le VS est un nouvel équipement logique introduit par la solution VNT qui a, pour fonctionnalité, de connecter toutes les VMs locales possédant le même VNI. Nous pouvons ainsi, grâce à ces modifications, associer un VNI aux paquets sortants du nœud.

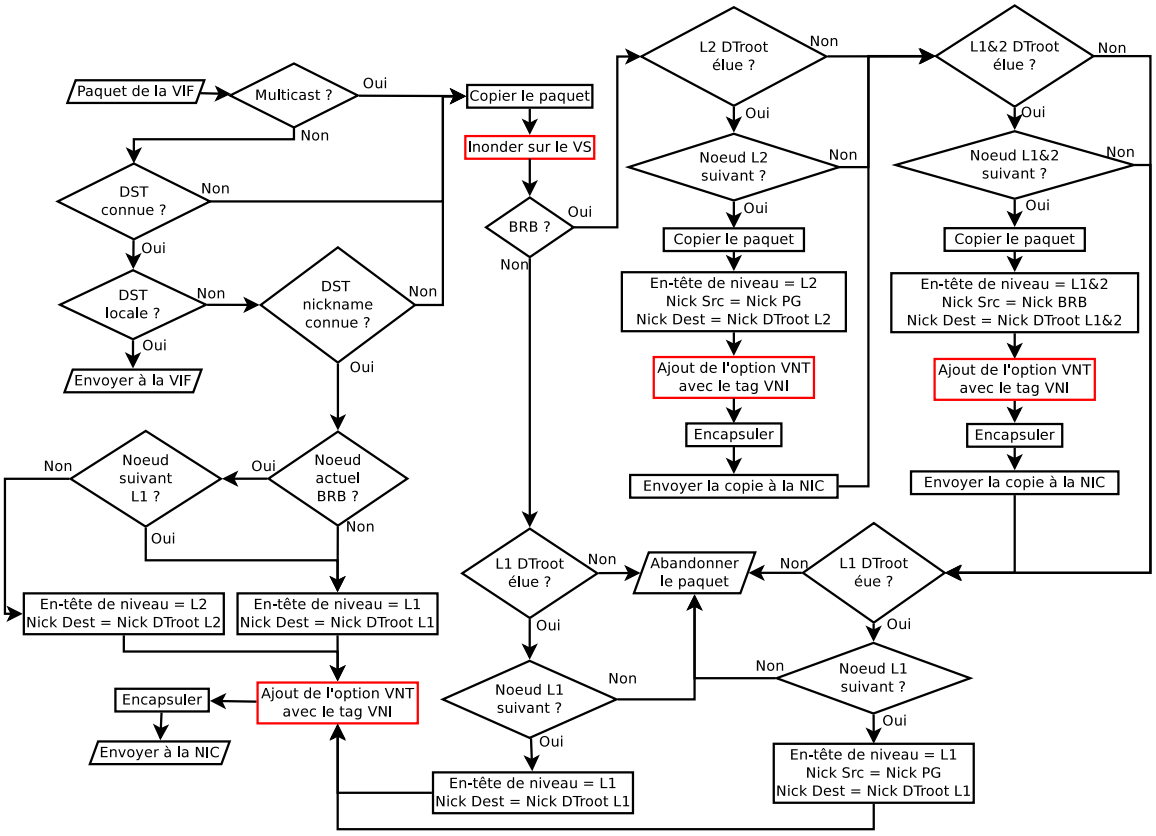


FIGURE 4.4 – Processus d'envoi d'un paquet

4.3.1.2.2 Modification du processus de réception des paquets

Nous avons, au sein de MLTP, deux processus de réception des paquets en fonction du type de ceux-ci. En effet, les paquets peuvent être, de type unicast ou de type multicast.

Dans le cas des paquets de type unicast, les modifications apportées au processus de réception sont directes. La première consiste à vérifier si le nœud supporte le VNI utilisé par le message. Si ce n'est pas le cas, alors le paquet est rejeté. La seconde modification intervient lorsque le paquet est accepté. Une fois celui-ci décapsulé, le nœud envoie le paquet au VS qui, ensuite, acheminera le paquet à la VM. De par la conception et la gestion des VNIs, cette seconde modification est nécessaire. La Figure 4.5 met en évidence (en rouge) les modifications que nous avons effectuées.

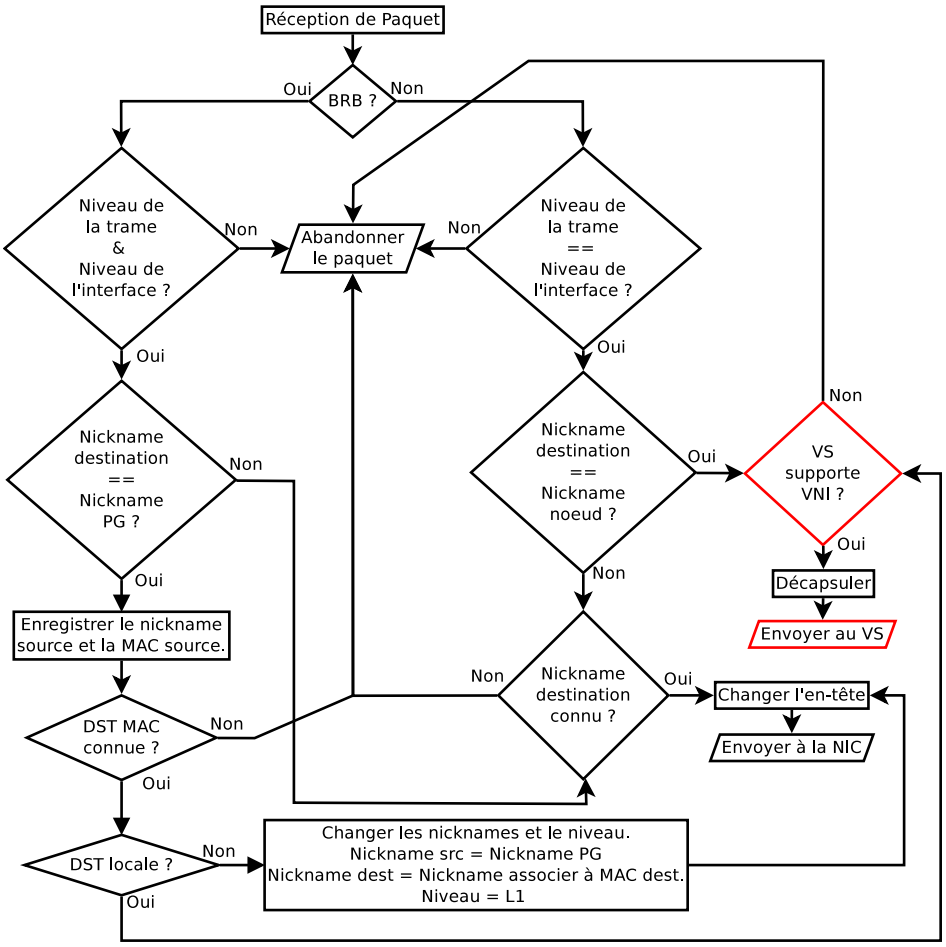


FIGURE 4.5 – Modification du processus de réception d'un paquet unicast

Le processus de réception des paquets multicast a aussi été modifié. On lui a ajouté les deux mêmes étapes que dans le processus unicast, à savoir le contrôle du VNI et l'envoi des paquets au VS. Cependant, comme le processus multicast est plus complexe, ces nouvelles étapes ne se situent pas au même moment dans le processus. La Figure 4.6 montre l'intégration de ces nouvelles étapes (en rouge) dans le processus.

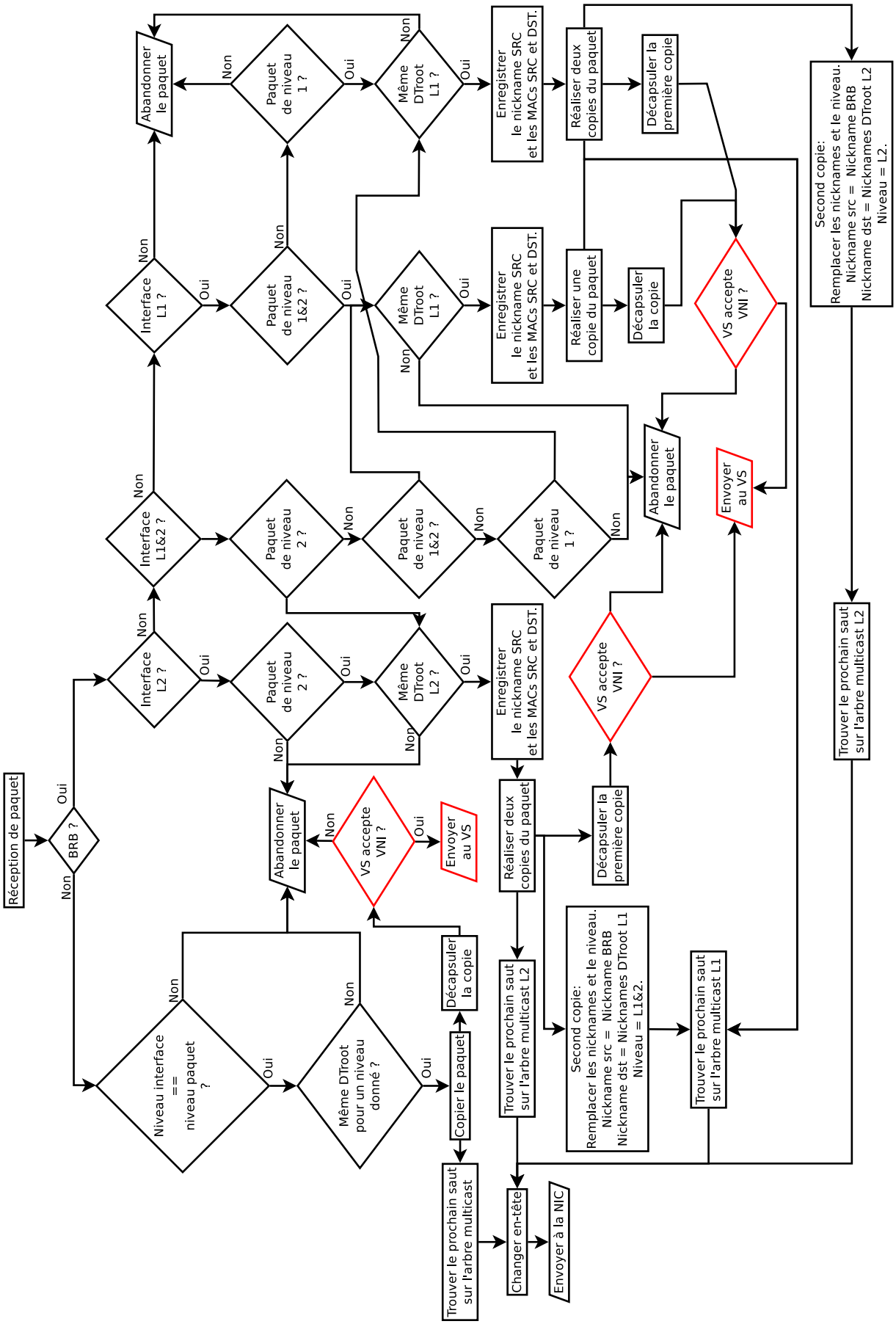


FIGURE 4.6 – Modification du processus de réception d'un paquet multicast

4.3.2 Enregistrement des VNIs

Le fait d'enregistrer les VNIs utilisés dans les autres campus au niveau d'une gateway permet une meilleure utilisation du backbone. En effet, au lieu de vérifier le VNI du paquet à l'arrivée de celui-ci dans la gateway d'un autre campus, on vérifie si le VNI est utilisé, ou non, dans d'autres campus et la gateway ne route le paquet que vers ces campus. Grâce à cette méthode, nous réduisons l'utilisation du backbone et, surtout, les paquets qui transitent via le backbone ne sont plus rejetés une fois arrivés au campus. Néanmoins, pour réaliser la vérification avant l'envoi du paquet, il nous faut enregistrer l'utilisation des VNIs au sein de tous les campus du réseau. Pour cela, nous avons étudié plusieurs solutions qui permettent d'enregistrer un grand nombre de valeurs, au moins 2^{16} VNIs, tout en utilisant le moins d'espace mémoire disponible car on doit enregistrer l'utilisation de ces VNIs pour tous les campus du réseau.

4.3.2.1 Solutions étudiées

Nous avons étudié plusieurs solutions permettant le stockage d'information qui, soit optimise l'espace mémoire nécessaire au stockage, soit réduit le temps de recherche de valeurs enregistrées dans ces structures. La première solution étudiée est un algorithme de compression des entiers appelé l'Elias gamma coding. Il utilise 773674750 bits pour stocker les 2^{24} VNIs. Dans [66] les auteurs proposent un nouvel algorithme qu'ils comparent à certains déjà existants. Dans leurs résultats, on voit que leur algorithme a le meilleur ratio de compression qui, au mieux, est de 6 :1. Néanmoins, ce ratio de 6 :1 n'est pas suffisant pour notre problème. De plus, à chaque fois que l'on a besoin de chercher une valeur, nous devons décompresser l'ensemble des valeurs puis effectuer la recherche parmi ces valeurs. Cette démarche demande un temps de calcul supplémentaire et implique un délai plus important pour le traitement des paquets.

Pour éviter d'avoir à décompresser et recompresser les valeurs à chaque fois qu'elles sont nécessaires, nous avons cherché un "Compressed Self-Index" [67]. Ces structures permettent de conserver la possibilité de chercher une valeur une fois les données compressées. Ceci permet de ne pas avoir à décompresser et recompresser les valeurs quand elles sont nécessaires. Néanmoins, le problème de cette technique est la taille des données. Comme pour la compression, le meilleur ratio de compression est de 6 :1.

Nous nous sommes ensuite orientés vers les arbres tel que le R-tree. Cette solution forme un arbre et, en suivant une branche, on peut retrouver la valeur. Cet arbre possède autant de niveaux qu'il y a de digits dans le nombre à enregistrer (unité, dizaine, centaine, ...). Cela représente donc, pour les VNIs, 24 niveaux avec, pour le premier niveau, 9 nœuds puis, pour le second niveaux, $9 * 9$ nœuds et ainsi de suite ce qui, au final, implique $9^{24} \approx 7,976644308 \times 10^{22}$ nœuds pour le vingt-quatrième niveau, ce qui est trop conséquent.

Nous avons ensuite regardé du côté des tables de hachage et du filtre de Bloom. Cependant, bien que la recherche puisse être effectuée rapidement, l'espace mémoire utilisé

est beaucoup plus important que l'utilisation d'un tableau de bits à 16 millions de bits. Or, l'élément crucial de ce problème est l'espace mémoire utilisé pour stocker les VNIs. Nous nous sommes donc orientés vers des structures plus spécifiques à notre problème de stockage de 16 millions de VNIs. Nous avons défini quatre techniques de stockage différentes que nous allons comparer dans notre simulateur.

La première est un tableau de bits qui en utilisent autant qu'il y a de valeurs à stocker. Chaque case contient un seul bit qui indique si le VNI est présent ou non. La valeur du VNI est obtenue à l'aide de l'index de la case du tableau. L'utilisation de cette structure utilise un espace constant de mémoire qui est de 2^{24} bits. Cependant, un seul tableau n'est pas suffisant car celui-ci ne peut contenir que les informations pour un seul data center alors que notre objectif est d'en inter-connecter plusieurs. Il en résulte que chaque BRB de la Pseudo Gateway utilise 2^{24} bits pour stocker les VNIs de chaque data center. Au final, cela peut devenir conséquent. Par exemple, l'espace mémoire utilisé pour trois data centers est d'environ 50 megabits $3 * (2^{24})$.

La seconde technique consiste à enregistrer les VNIs, de façon ordonnée et adjacente, dans un espace mémoire mono bloc. On peut assimiler cette technique à une liste ordonnée de VNIs. Néanmoins, nous n'avons pas de pointeur et, pour passer à la valeur suivante, nous utilisons le fait que la taille des VNIs est constante (24 bits) et qu'ils sont enregistrés de façon contiguë. Nous avons ainsi un espace mémoire utilisé qui dépend seulement du nombre de VNIs utilisés. Cependant, le fait de maintenir l'ordre des VNIs implique des difficultés de gestion de ces derniers et deux solutions peuvent être utilisées. La première consiste à laisser l'espace nécessaire entre les VNIs utilisés pour ensuite pouvoir insérer les VNIs au fur et à mesure. Cette solution implique donc de bénéficier, dès le départ, d'un bloc mémoire suffisant pour y stocker tous les 2^{24} VNIs de 24 bits, ce qui représente une taille d'environ $2^{24} * 24 = 402653184 bits \approx 402 megabits$.

La troisième structure est une structure hybride qui peut contenir, soit une liste, soit un tableau de bits. Dans un premier temps, les VNIs sont divisés en 64 ensembles. Chaque ensemble contient donc 262 144 VNIs. La représentation de ces ensembles peut se faire, soit par une liste, comme la seconde technique, soit par un tableau de bits, comme la première technique. Cette structure contient donc :

- Un tableau de 64 bits qui sert d'index. Chaque valeur du tableau indique si un ensemble, correspondant à l'indice de la case, existe ou non.
- 64 cellules qui contiennent chacune :
  - Un entier sur 64 bits. Cet entier est utilisé pour coder plusieurs informations.
    - Le premier bit sert à indiquer si l'on a, dans cette cellule, un tableau de bits ou une liste.
    - Le second bit est utilisé pour indiquer si la cellule contient plus de 90 pourcent des VNIs de son ensemble. Si c'est le cas, on traite alors la totalité des VNIs de cet ensemble comme "existant" et on récupère l'espace mémoire

du tableau de bits car il n'est plus nécessaire de stocker, ni la liste, ni le tableau. En effet, la cellule ne contient plus qu'un seul entier de 64 bits.

— Les bits restants possèdent une signification différente en fonction de la valeur du premier bit : Si l'on a une liste, alors on utilise 24 bits pour coder le nombre d'éléments dans la liste et 24 bits pour coder le premier VNI de la liste. Si l'on a un tableau, alors on utilise 24 bits pour coder la valeur du VNI minimal présent dans l'ensemble et 24 bits pour coder la valeur du VNI maximal présent dans l'ensemble. Ces éléments nous permettent de retrouver facilement la taille de la cellule.

— Une liste de VNIs de 24 bits chaque ou alors un tableau de bits de taille $(vni_max - vni_min)$.

Cependant, cette structure, bien que permettant de réduire l'espace mémoire utilisé pour stocker les VNIs, ne peut être retenue dans la réalité en raison de la perte d'informations sur les VNIs. En effet, une fois qu'un VNI est utilisé, celui-ci le sera constamment et la perte d'informations sur les VNIs ne pose, alors, pas de problème. Néanmoins, dans la réalité, les VNIs sont utilisés pendant des durées variables et peuvent donc disparaître. Il nous faut donc être capable de récupérer cette information. Par exemple, dans une cellule où 90% des VNIs sont utilisés, on n'a plus qu'une seule valeur. Or, si des VNIs partent, on aura alors moins de 90% et on ne peut plus savoir quels sont les VNIs restants.

La dernière structure étudiée est le Range Set. Cette structure est une liste de cellules ne contenant que deux VNIs. Il en résulte que chaque cellule possède une taille de $2 * 24$ bits. Cependant, une cellule peut représenter plusieurs VNIs. En effet, les deux VNIs contenus sont le VNI minimum et le VNI maximum de l'ensemble. Ils englobent donc tous les VNIs compris entre ceux-ci. Il est donc possible de représenter l'ensemble des VNIs utilisés avec seulement une cellule et donc deux VNIs si ceux-ci sont continus. Cette structure est donc très intéressante pour optimiser l'espace mémoire utilisé ainsi que le temps de recherche à condition d'avoir un faible nombre de cellules.

4.3.2.2 Résultats et choix d'une structure de stockage

Dans ce paragraphe, nous allons comparer l'espace de stockage utilisé par chaque structure. Parmi les quatre structures étudiées dans la section 4.3.2.1, une structure possède une taille fixe. Cette structure est un tableau de bits avec une taille de 2^{24} bits, ce qui fait 2 097 152 octets soit environ 2 Mo. La liste de VNIs a une taille qui dépend du nombre de VNIs. On en calcule la taille en multipliant le nombre de VNIs par l'espace occupé par un VNI. On a donc une taille égale à $nb_vni * 24$. La troisième et la quatrième structures ont des tailles variant en fonction du nombre de VNI.

Pour étudier l'évolution de ces tailles, nous avons utilisé le langage Python afin de réaliser une simulation de l'utilisation de l'espace mémoire en fonction de la structure utilisée pour enregistrer les VNIs. Afin d'être au plus proche de la réalité, nous avons

simulé un cycle arrivée/départ des clients. Pour les arrivées de clients, nous avons utilisé une loi de Poisson de paramètre $\mu = 230$ représentée dans la Figure 4.7. Nous avons aussi utilisé une loi de Poisson de paramètre $\mu = 23$ pour le départ des clients, représentée dans la Figure 4.8. Les paramètres μ d'arrivée et de départ sont basés sur l'analyse des données mensuelles des clients de Gandi.

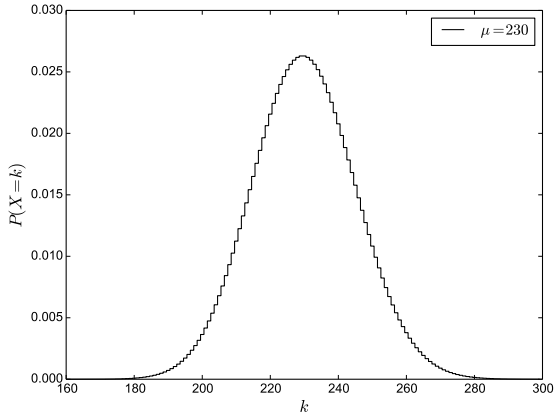


FIGURE 4.7 – Fonction de masse de la loi de poisson de paramètre $\lambda = 230$

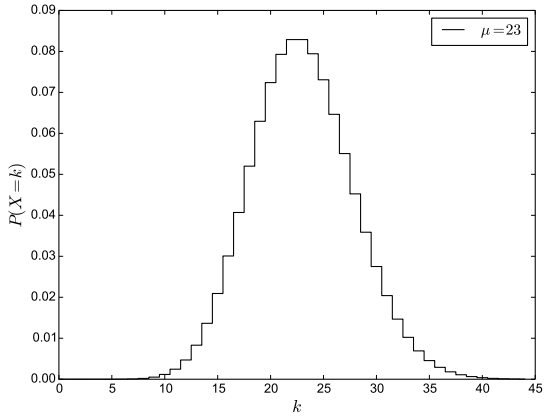


FIGURE 4.8 – Fonction de masse de la loi de poisson de paramètre $\lambda = 23$

À l'aide de ses valeurs, nous avons simulé 12000 cycles arrivées/départs et aussi 12000 cycles départs/arrivées. Dans la Figure 4.9, nous pouvons voir les résultats de la simulation lorsque les arrivées sont traitées avant les départs.

En effet, à chaque cycle, nous simulons l'arrivée de nouveaux clients et donc l'ajout de VNIs, puis le départ de clients et donc la suppression de VNIs. On peut constater que, lors du traitement des arrivées avant les départs, l'espace mémoire utilisé par la structure range set varie autour de 10^3 bits et surtout, qu'elle varie à chaque cycle. Cette variation s'explique en raison de l'ajout des nouveaux VNIs en essayant d'optimiser l'espace mémoire utilisé. C'est à dire que nous ajoutons les VNIs de telle sorte que nous remplissons les espaces vides entre les cellules du range set. Si on a, par exemple, deux cellules [1-10] et [13-15] et que l'on doit ajouter 3 VNIs, on va alors ajouter les VNIs 11,12 et 16 pour obtenir une seule cellule [1-16]. En utilisant cette méthode d'insertion, nous réduisons le nombre de cellules à chaque cycle. De plus, comme nous avons plus d'arrivées que de départs, nous pouvons donc, à chaque cycle, remplir tous les "trous" du range set pour n'obtenir, au final, qu'une seule cellule. Mais comme nous traitons ensuite les départs et que ceux-ci sont aléatoires, on peut, dans le pire des cas, avoir $nb_depart + 1$ cellules dans le range set. En effet, les VNIs supprimés sont obtenus aléatoirement pour simuler le fait que des clients partent, or, comme le fait de partir ne dépend pas du temps mais seulement du client, nous ne pouvons prévoir quels VNIS seront supprimés.

Afin d'optimiser encore plus l'espace mémoire utilisé, il est préférable de traiter les départs avant les arrivées. En effet, comme déjà précisé, nous avons plus d'arrivées que de départs, ce qui permet de remplir les "trous" causés par les départs. La Figure 4.10 montre

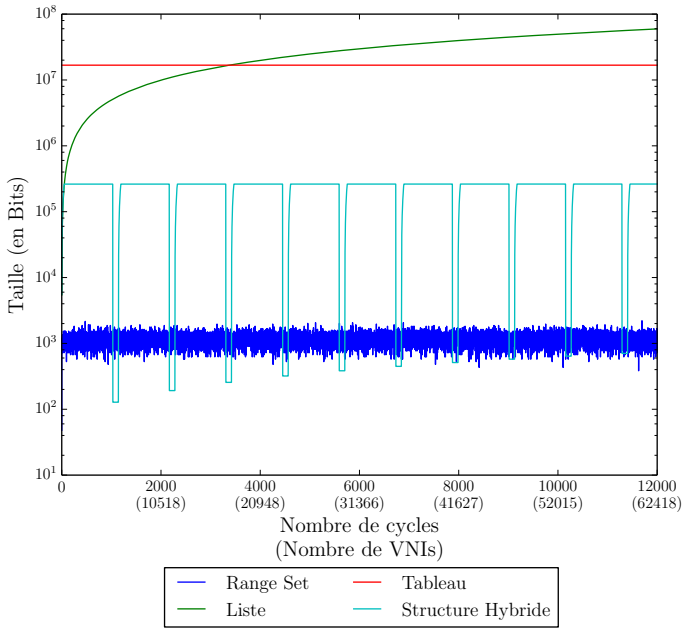


FIGURE 4.9 – Taille en bits des différentes structures en fonction du nombre de VNIs et en suivant le cycle arrivées/départs.

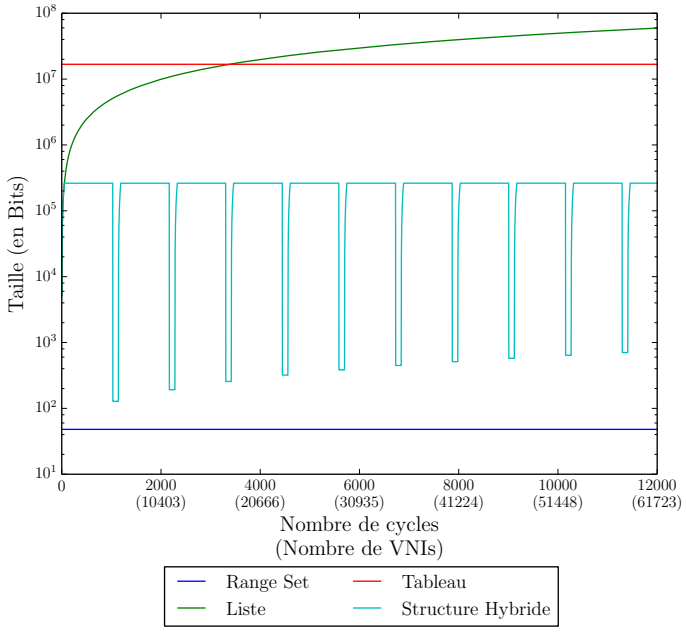


FIGURE 4.10 – Taille en bits des différentes structures en fonction du nombre de VNIs et en suivant le cycle départs/arrivées.

les résultats obtenus lorsque les départs sont traités en premier. On peut constater que la structure range set possède une taille constante tout au long de la simulation. En effet, la structure ne possède qu'une seule cellule à chaque cycle et seule la valeur VNI maximale évolue au cours du temps.

Cependant, les deux simulations précédentes ne permettent pas réellement d'étudier le passage à l'échelle des structures étudiées car le nombre maximal de VNIs enregistrés

n'atteint que 62418, ce qui est faible, comparé aux 2^{24} VNIs possibles. Pour cette raison, nous avons fait une seconde simulation, mais cette fois, nous avons multiplié par 100 les paramètres μ d'arrivées et de départs et nous nous sommes limités à 120 cycles, ce qui nous fait obtenir environ 2 485 649. Nous obtenons alors les résultats présentés dans les Figures 4.11 et 4.12.

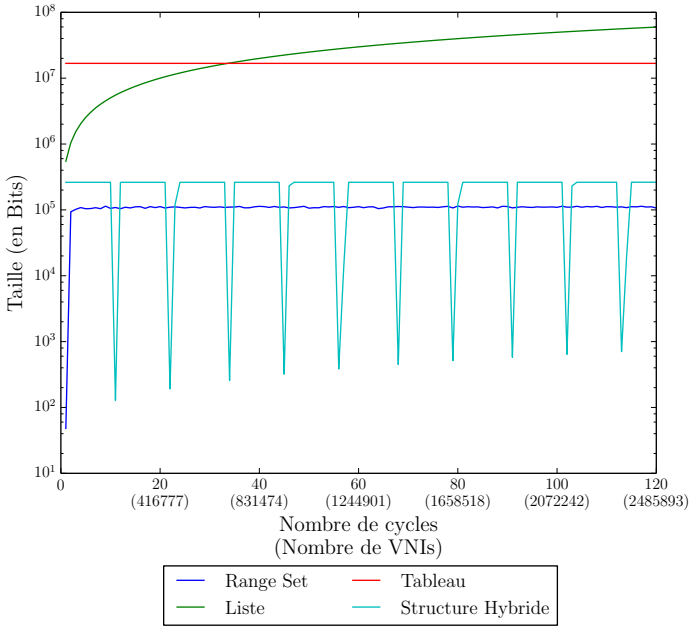


FIGURE 4.11 – Taille en bits des différentes structures en fonction du nombre de VNIs et en suivant le cycle arrivées/départs.

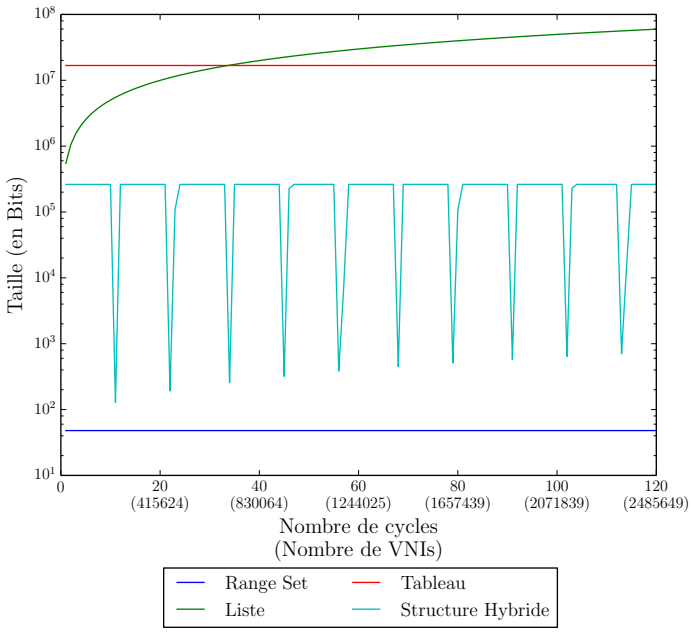


FIGURE 4.12 – Taille en bits des différentes structures en fonction du nombre de VNIs et en suivant le cycle départs/arrivées.

La Figure 4.11 correspond au cas où l'on traite les arrivées avant les départs. Dans

ce cas-là, on peut remarquer alors que le range set consomme beaucoup plus de mémoire que lorsqu'on avait peu de départs. En effet, la taille du range set dépend du nombre de départs. Plus le nombre de départs est important, plus le range set est segmenté. Comme mentionné précédemment, le pire des cas implique $nb_depart + 1$ cellules dans le range set, or, dans cette simulation, nous avons défini le nombre de départs via une loi de poisson de paramètre $\mu = 2300$, ce qui implique un plus grand nombre de départs par rapport au scénario basé sur les données de Gandi. Dans ce cas-ci, l'augmentation de segments à enregistrer occupe donc plus d'espace mémoire.

Comme dans les résultats du scénario précédent, on peut remarquer dans la Figure 4.12 que, si l'on traite les départs puis les arrivées, l'espace mémoire utilisé par le range set est toujours égal à la taille de deux nicknames. La structure range set est donc la plus intéressante sur le plan de l'espace mémoire utilisé, à condition de réaliser d'abord les départs puis de sauvegarder les arrivées. En ce qui concerne le temps de recherche, la structure range set est aussi très intéressante, car la recherche consiste en deux tests ayant pour but de savoir si le VNI recherché est supérieur au VNI minimum et inférieur au VNI maximum.

4.4 Performances de MLTP+VNT

Dans cette section nous étudions les performances de MLTP+VNT tant au niveau du plan de contrôle (Section 4.4.1) que du plan de données (Section 4.4.2).

4.4.1 Évaluation du plan de contrôle

L'évaluation du plan de contrôle de MLTP+VNT se focalise sur l'étude du temps de convergence, comme pour les solutions SMLTP (Section 3.4.1.6.3) et MLTP (Section 3.4.2.6). Nous utilisons le même scénario que lors des deux expériences précédentes et nous obtenons les résultats présentés dans la Figure 4.13. On y voit que, comme pour SMLTP et MLTP, le temps de convergence de MLTP+VNT est au maximum de 30 secondes, ce qui correspond au temps de deux LSPs. Ces résultats identiques étaient attendus car nous n'avons pas modifié la découverte des nœuds au sein d'un campus et nous avons gardé un temps de 15 secondes par LSPs. Il advient donc que le plan de contrôle de MLTP+VNT converge de la même façon que celui de MLTP.

4.4.2 Évaluation du plan de données

Pour évaluer le plan de données de MLTP+VNT, nous nous sommes concentrés sur l'étude du débit (Section 4.4.2.3) et du délai (Section 4.4.2.2) entre deux VMs au sein du réseau. Ces deux VMs peuvent, ou non, être hébergées sur des serveurs dans le même campus. Une fois les résultats obtenus, nous les comparons avec MLTP (Section 3.4.2) et

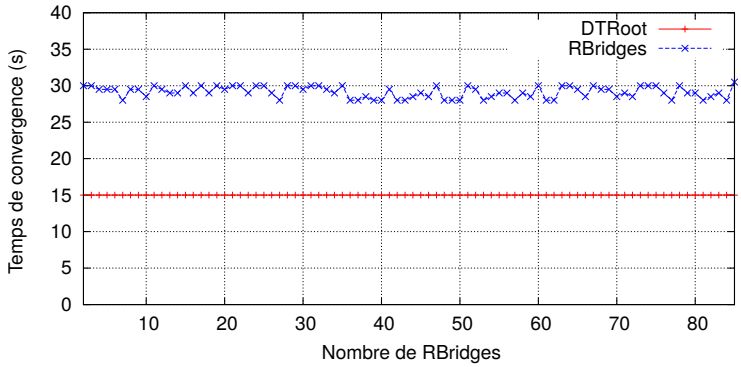


FIGURE 4.13 – Évolution du temps de convergence du plan de contrôle de MLTP+VNT en fonction du nombre de nœuds.

VXLAN [68] ainsi que la solution native de Linux. Une troisième évaluation est réalisée pour étudier le comportement de MLT+VNT concernant le multi-chemins et lorsqu'un de ces chemins est coupé.

4.4.2.1 Plate-forme de test

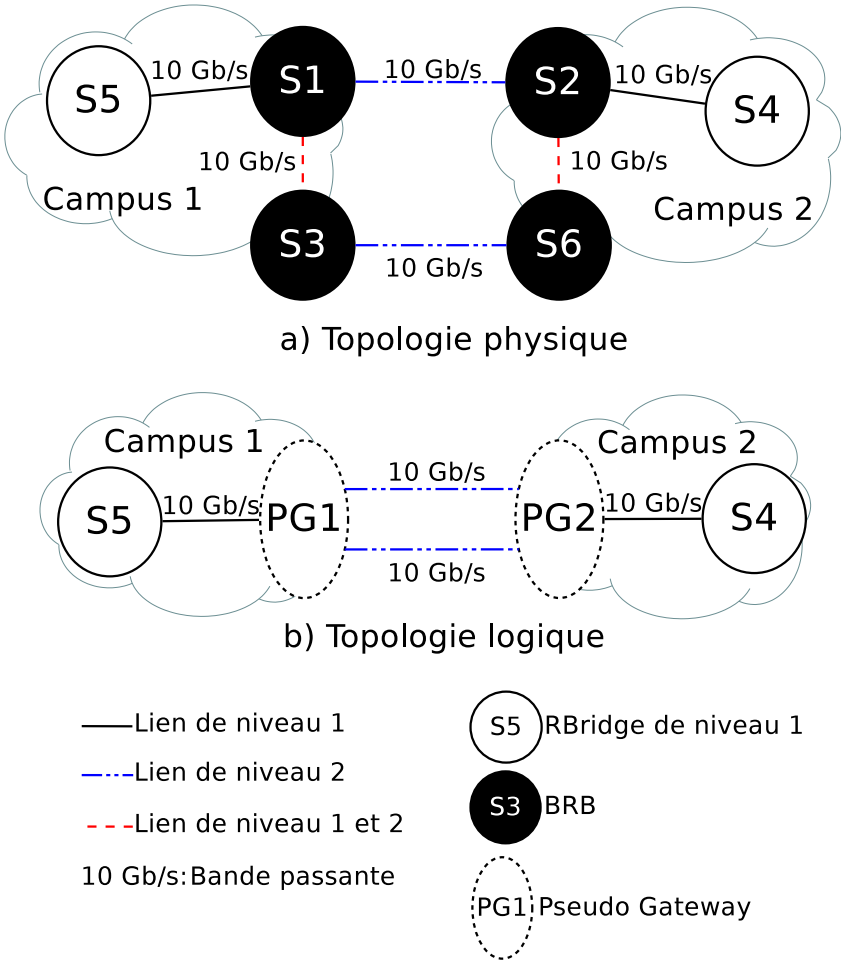


FIGURE 4.14 – Topologie physique (a) et logique (b) de la plate-forme de test

FIGURE 4.15 – Arbre de distribution du campus 1

FIGURE 4.16 – Arbre de distribution du campus 2

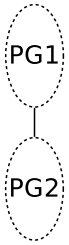


FIGURE 4.17 – Arbre de distribution du backbone

La plate-forme de test, présentée dans la Figure 4.14, est identique à celle utilisée pour étudier les performances de MLTP (Section 3.4.2.6). Les arbres de distribution obtenus (Figures 4.15, 4.16 et 4.17) sont aussi identiques à ceux obtenus lors de l'étude de MLTP. On a toujours deux campus, chacun composé d'un RBridge et de deux BRBs avec les BRBs regroupés au sein des PGs. Dans cette topologie, deux chemins de coût identique entre les nœuds S5 et S4 existent du fait des deux connections entre les PGs des campus et parce que les liens entre BRBs d'un campus ont un coût nul. Les chemins physiques S5-S1-S2-S4 et S5-S1-S3-S6-S2-S4 sont vus comme le même lien logique S5-PG1-PG2-S4.

4.4.2.2 Étude de la latence

Pour pouvoir comparer les résultats de latence de MLTP+VNT avec ceux de MLTP et d'Ethernet, nous avons réalisé les mêmes expériences que dans la Section 3.4.2.6. La Figure 4.18 montre les résultats de latence que nous avons obtenus pour MLTP+VNT. On remarque toujours la corrélation entre le nombre de sauts sur le chemin et la latence car plus le nombre de sauts est important et plus la latence augmente. On remarque aussi que l'intégration de la solution VNT à MLTP cause une très faible augmentation de la latence. Cette augmentation est due au contrôle de l'identifiant VNI du paquet lorsqu'il est reçu par un nœud. Cependant, cette augmentation de la latence diffère en fonction du chemin ; l'augmentation de la latence est plus importante lorsque le paquet quitte le campus et passe par un BRB. En effet, le BRB doit réaliser un contrôle supplémentaire sur le VNI du paquet pour savoir s'il doit l'envoyer sur le backbone. Cette vérification de l'utilisation du VNI dans le campus distant implique un temps de traitement plus long du paquet et donc impacte la latence. Cependant, cet impact est négligeable car il représente environ 3% d'augmentation de la latence par rapport à Ethernet, et moins de 2% par rapport à MLTP.

Nous nous sommes ensuite focalisés sur le chemin S5-S4 pour lequel nous avons étudié la latence en fonction de la technologie choisie. Nous avons choisi le chemin S5-S4 car c'est le seul dans notre plate-forme de test qui utilise toutes les modifications que nous avons réalisées pour MLTP+VNT. Les résultats sont affichés dans la Figure 4.18. On y voit, comme dans la Figure 4.19, que la solution native Linux a la plus faible latence parmi les quatre solutions étudiées, mais elle ne permet d'isoler ni les paquets, ni les campus.

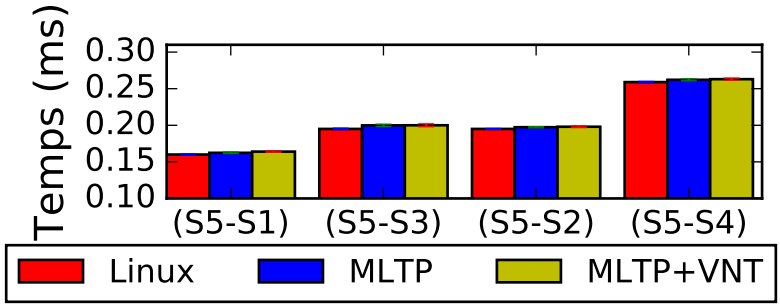


FIGURE 4.18 – Évolution de la latence de MLTP+VNT en fonction du nombre de nœuds sur le chemin.

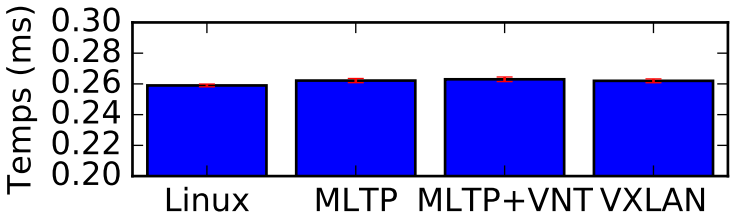


FIGURE 4.19 – Étude de la latence sur le chemin S5-S4 en fonction de la solution.

Les trois autres solutions que sont MLTP, MLTP+VNT et VXLAN sont très proches les unes des autres et ne dégradent que faiblement la latence par rapport à la solution Linux native. En effet, les résultats de latence de MLTP+VNT et VXLAN sont identiques et environ 3% plus importants que ceux de la solution Linux native.

4.4.2.3 Étude du débit

Comme pour l'étude de la latence, nous avons utilisé les mêmes expériences que lors de l'étude de MLTP (Section 3.4.2.6) pour pouvoir comparer les résultats de MLTP+VNT avec ceux de MLTP. La Figure 4.20 montre la moyenne des résultats ainsi que l'écart type que nous avons obtenus lors de l'étude du débit en fonction du chemin. On peut y voir que la solution MLTP+VNT a un débit inférieur à MLTP et Linux. Ceci s'explique du fait que MLTP+VNT nécessite une option supplémentaire dans l'en-tête du paquet et que cette option a une taille de 64 bits. L'ajout de cette option, qui contient le VNI, double la taille de l'en-tête MLTP et réduit la taille du payload du paquet. On peut aussi remarquer que le débit de MLTP+VNT est légèrement diminué lorsque le chemin passe par le backbone. Par exemple, les chemins S5-S2 et S5-S4 ont un débit approximativement 4% moindre que lorsque le chemin reste au sein d'un campus (chemin S5-S1). Ceci s'explique car les BRBs doivent vérifier le VNI puis changer les informations de l'en-tête MLTP+VNT des paquets, et donc, engendre un temps de traitement plus long pour chaque paquet. Néanmoins, comme le montre la Figure 4.21, la solution MLTP+VNT possède un débit qui est supérieur à celui de NVGRE [30] et quasiment identique à celui de VXLAN. Le

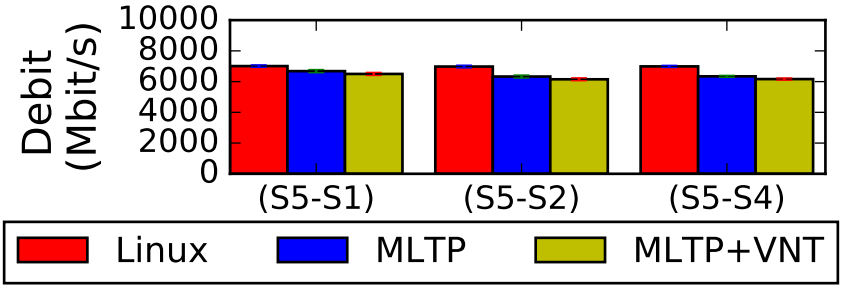


FIGURE 4.20 – Évolution du débit en fonction du nombre de nœuds sur le chemin.

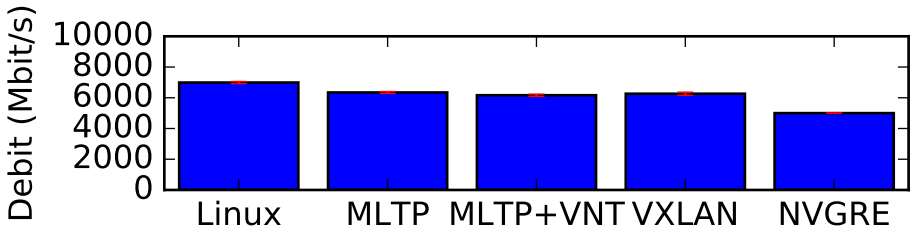


FIGURE 4.21 – Étude du débit sur le chemin S5-S4 en fonction de la solution choisie.

débit présenté pour NVGRE correspond aux meilleurs débits sans offloading présentés dans les white papers [69] et [70].

4.4.2.4 Résilience

Grâce à sa capacité d'utiliser de multiples chemins de même coût, TRILL possède une résistance accrue aux pannes. En effet, si deux chemins de même coût sont disponibles, alors les deux seront utilisés, et si un chemin n'est plus utilisable, alors le trafic sera envoyé sur le second chemin. De cette façon, le trafic n'est pas interrompu.

Comme MLTP+VNT est basé sur le protocole TRILL, il est intéressant de vérifier que MLTP+VNT bénéficie lui aussi du multi-chemins. Pour vérifier cela, nous démarrons deux communications entre quatre VMs hébergées dans les nœuds S5 et S4. Chaque communication possède un débit de 1 Gbit/s, un "Flow id" unique et se fait entre une VM hébergée dans S5 et une VM hébergée dans S4 (Figure 4.14). De cette façon nous sommes capables de séparer les communications pour qu'elles empruntent des chemins différents. Les chemins entre S5 et S4 diffèrent au niveau du backbone grâce aux deux liens d'interconnexion entre la PG1 et la PG2 et chaque communication va utiliser un lien. Le chemin 1 est constitué des nœuds S5-S1-S2-S4 alors que le chemin 2 passe par les nœuds S5-S1-S3-S6-S2-S4 avec les nœuds S1 et S3 qui forment la PG1 et les nœuds S2 et S6 qui forment la PG2. En conséquence, sur chacun des liens entre PG1 et PG2, nous avons un trafic de 1 Gbit/s. Dix secondes après le début de ces communications, nous coupons le lien S1-S2 entre les PGs pour simuler une panne. La Figure 4.22 nous montre le comportement du réseau lors de cette panne. On constate que le trafic qui transitait

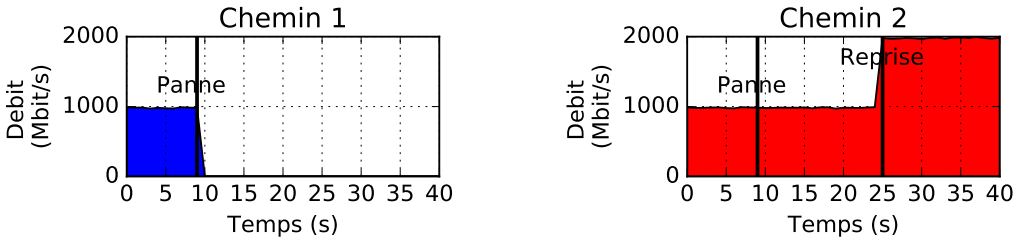


FIGURE 4.22 – Changement de chemin après une panne.

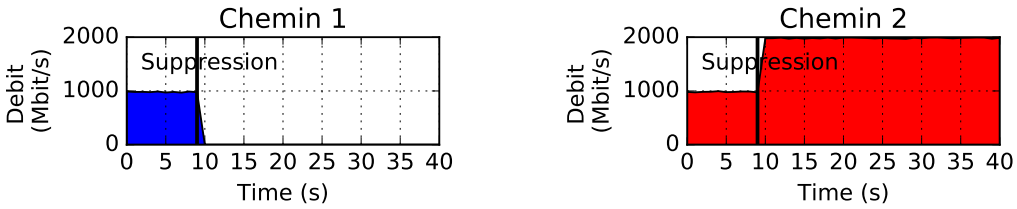


FIGURE 4.23 – Changement de chemin lorsqu'on efface un des chemins de la table de commutation.

par le lien coupé s'arrête au moment de la coupure, soit 10 secondes après le début de la communication. Puis, à la vingt cinquième seconde, donc quinze secondes après la coupure, le trafic sur l'unique chemin restant est doublé pour atteindre 2 Gbit/s. La résultante est que les deux communications utilisent désormais ce chemin. L'interruption du trafic a duré quinze secondes, ce qui correspond au temps d'un LSP et au temps de détection d'une panne.

Nous avons réalisé une seconde expérience dans laquelle nous ne simulons pas une panne, mais nous supprimons directement le chemin de la table de commutation du nœud pour comprendre pourquoi nous avons une interruption de quinze secondes avant le changement de chemin. Dans la Figure 4.23 on voit que le trafic qui empruntait le chemin supprimé est immédiatement transféré sur le chemin encore valide. On en déduit que le temps d'interruption du trafic de quinze secondes, vu lorsqu'une panne se produit, est dû au temps de propagation des informations concernant la panne dans le réseau.

Cependant, dans notre test, tous les BRBs possèdent les informations des deux flux en raison de notre intervention pour séparer les flots sur des chemins différents. Or, dans une situation normale, les informations de routage ne sont pas synchronisées entre les BRBs. Ceci implique que, si une panne à lieu au niveau d'un BRB, les autres BRBs ne connaissent pas les informations de routage des flots routés par celui-ci. Il advient, lorsque les flots sont reroutés vers les BRBs restants, que ces derniers doivent alors, via des messages de broadcasts, redécouvrir les nicknames associés aux adresses MAC de destination. De plus, dans le cas où le chemin aller diffère du chemin retour au niveau des BRBs, il est alors impossible d'établir une communication unicast entre la source et la destination. Ces problèmes sont détaillés dans la section 5 et une solution est présentée dans la section 5.2.

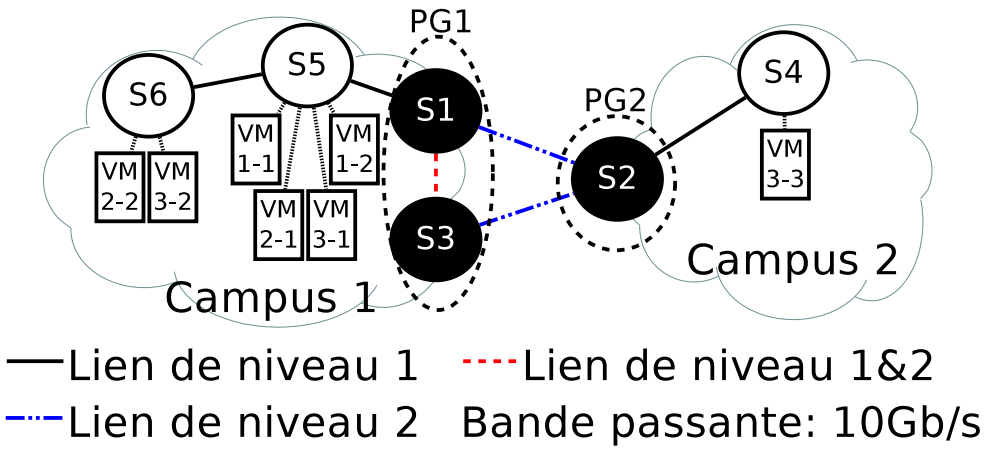


FIGURE 4.24 – Topologie utilisée pour tester l'isolation

4.4.2.5 Isolation des utilisateurs et taille des tables de commutation

Pour évaluer la capacité de MLTP+VNT à isoler les réseaux virtuels, nous avons testé MLTP+VNT dans le réseau présenté Figure 4.24. Nous définissons, dans ce réseau, trois utilisateurs avec chacun leur propre VNI et plusieurs VMs. Pour identifier les VMs, nous utilisons la notation suivante : VM X-Y avec X l'identifiant du réseau virtuel (le VNI) et Y l'identifiant de la VM au sein du réseau virtuel. Il en découle que le premier utilisateur possède le VNI 1 et les VMs : VM 1-1 et VM 1-2.

Pour propager les adresses MAC de toutes les VMs présentes dans le réseau, nous exécutons, à l'intérieur de chacune d'elle, un script qui envoie en permanence des requêtes ARP. Dans la Figure 4.25, nous indiquons le nombre d'adresses MACs enregistrées dans chaque nœud de la topologie en fonction de la solution utilisée. On remarque que la solution standard de niveau 2 (Ethernet) implique que chaque nœud voit l'ensemble des autres nœuds et enregistre toutes les adresses MACs des nœuds physiques et des VMs du réseau. Ceci résulte dans le fait que chaque nœud enregistre les 13 adresses MACs du réseau de test.

Les résultats pour MLTP montrent que les nœuds enregistrent moins d'adresses MAC. Ceci s'explique car les campus sont indépendants dans MLTP et, donc, les nœuds du campus 1 ne voient pas les nœuds du campus 2 mais seulement la PG de leur campus. Par exemple, le nœud S6 qui sauvegarde 13 adresses MAC avec Ethernet, n'en sauvegarde plus que 9 avec MLTP. Ces 8 adresses sont les adresses MAC des nœuds physiques du campus 1 (S5, PG1) et les adresses MAC des VMs (VM 1-1, VM 1-2, VM 2-1, VM 2-2, VM 3-1, VM 3-2, VM 3-3).

La solution MLTP+VNT permet de réduire la taille des tables de commutation en réduisant le nombre d'adresses MAC enregistrées. En effet, dans le pire des cas, MLTP+VNT aura les mêmes résultats que MLTP, mais dans les autres cas MLTP+VNT permet de n'enregistrer que les adresses MACs utiles. Par exemple, le nœud S6 n'enregistre plus que 7 adresses MACs. Les deux adresses MACs des VMs VM 1-1 et VM 1-2 ne sont plus

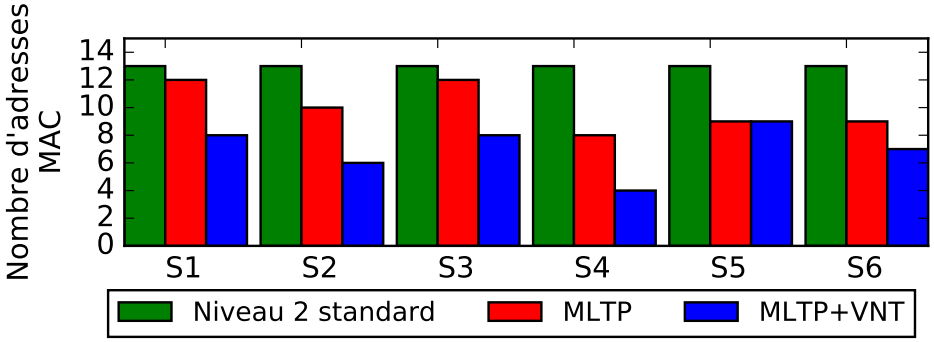


FIGURE 4.25 – Nombre d'adresses MAC dans chaque nœud

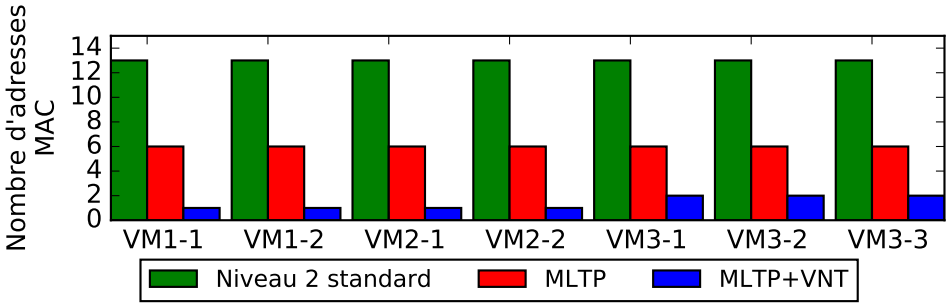


FIGURE 4.26 – Nombre d'adresses MAC dans chaque VM

enregistrées par S6 car il ne possède aucune VM appartenant à ce réseau virtuel.

La Figure 4.26 présente l'évolution de la taille des tables de commutation au niveau des VMs du réseau. Comme dans la Figure 4.25, on remarque qu'avec l'utilisation d'Ethernet les VMs enregistrent toutes les adresses MAC du réseau. Cependant, avec MLTP, les VMs n'enregistrent plus que les adresses MAC des autres VMs du réseau. Ceci s'explique par l'encapsulation des trames dans des trames MLTP, lesquelles masquent tous les nœuds de l'infrastructure. Par exemple, la VM 1-1 qui enregistre les 13 adresses MAC lors de l'utilisation d'Ethernet n'en n'enregistre plus que 6 avec MLTP, celles des 6 autres VMs. L'ajout de la solution VNT à MLTP nous permet de réduire encore plus ce nombre en ne gardant que les adresses MACs qui sont utilisées par la VM. En effet, la solution MLTP+VNT permet d'isoler les réseaux virtuels identifiés par les VNIs et on remarque dans la figure 4.26 que les VMs n'enregistrent plus que les adresses MAC des VMs faisant partie du même réseau virtuel (avec le même VNI). En reprenant l'exemple de la VM 1-1, celle-ci ne voit plus que l'adresse MAC de la VM 1-2.

La solution MLTP+VNT permet d'isoler les réseaux virtuels et donc les utilisateurs, bien qu'ils partagent et utilisent la même infrastructure. De plus, elle permet de réduire la taille des tables de commutation de tous les équipements du réseau.

4.4.3 Rétro-compatibilité

Dans la section 3.4.2.5, nous avons expliqué pourquoi MLTP est rétro-compatible avec TRILL et le standard de niveau 2. Il nous faut ici expliquer pourquoi l'implémentation de VNT au sein de MLTP ne modifie pas la rétrocompatibilité de la solution.

MLTP+VNT est toujours compatible avec le standard de niveau 2 en raison de la réalisation d'une encapsulation MAC-in-MAC. En effet, l'en-tête modifié par MLTP+VNT est l'en-tête TRILL qui est encapsulé par une trame Ethernet standard et, donc, tous les nœuds non MLTP+VNT sur le chemin de la trame vont commuter cette trame en se basant sur les informations contenues dans l'en-tête MAC.

Concernant la rétrocompatibilité du plan de contrôle de MLTP+VNT avec MLTP et TRILL, les modifications que nous avons réalisées au plan de contrôle sont :

— L'ajout de l'automate d'états pour les VNIs dans les RBridges.

— L'algorithme de calcul des VNIs.

— Le processus de découverte des VNIs.

Parmi ces modifications, seul le processus de découverte des VNIs pourrait impliquer des problèmes de compatibilité. En effet, les autres modifications sont propres au nœud et n'influent pas sur les autres nœuds du réseau alors que la découverte des VNIs impacte les LSPs échangés entre les nœuds. Nous avons modifié les LSPs en suivant les recommandations de TRILL, c'est à dire en utilisant le format prédéfini pour leur ajouter un champ optionnel. De ce fait, notre modification des LSPs, qui consiste à ajouter la liste des VNIs supportés par un nœud, est un champ optionnel et les RBridges ne reconnaissant pas cette option l'ignorent. On a donc ainsi conservé la retrocompatibilité avec les plans de contrôle de TRILL et de MLTP.

En ce qui concerne le plan de données de MLTP+VNT, nous avons réalisé les modifications suivantes par rapport à MLTP :

— Ajout du champ contenant le tag VNI à l'en-tête TRILL

— Modification des processus d'envoi et de réception des paquets.

Seul l'ajout du champ VNI à l'en-tête TRILL aurait pu poser des problèmes de rétrocompatibilité. En effet, les processus d'envoi et de réception sont propres aux nœuds alors que l'ajout de l'option modifie les messages entre ces nœuds. Pour fournir la rétrocompatibilité avec MLTP et TRILL, le champ VNI est ajouté à l'en-tête MLTP sous forme d'option et ainsi les nœuds ne supportant pas cette option l'ignorent.

MLTP+VNT est donc rétrocompatible avec MLTP et est rétrocompatible avec TRILL à condition de respecter la même contrainte que pour MLTP ; à savoir que les RBridges TRILL doivent être au sein des campus et ne doivent pas être utilisés comme BRBs.

4.5 Conclusion

L'implémentation de la solution VNT au sein de MLTP a nécessité de modifier la solution VNT ainsi que le protocole MLTP pour les faire fonctionner ensemble. Ceci nous a permis de fournir à MLTP la capacité d'isoler les utilisateurs ainsi que leurs données au sein du réseau, comme montré dans l'étude des performances de MLTP+VNT. De plus, nous avons un procédé qui, à l'aide des structures de stockage de VNIs, permet d'éviter de surcharger le réseau backbone avec du trafic inutile. La nouvelle solution MLTP+VNT propose donc deux nouvelles fonctionnalités d'isolation et d'optimisation tout en n'augmentant la latence que de 3% et en diminuant le débit que de 4%.

Chapitre 5

Amélioration de reprise après panne d'un BRB

Sommaire

5.1 Problématique

La solution MLTP+VNT, présentée dans la section 4.3 permet d'isoler les utilisateurs et leurs trafics tout en interconnectant des campus indépendants. Elle permet aussi d'optimiser l'utilisation du niveau 2 en n'envoyant que les messages appartenant à des réseaux virtuels étendus sur plusieurs campus. Cependant, pour réaliser cela, nous avons conçu un nouvel équipement physique, le BRB, et un nouvel équipement logique, la PG. Le BRB est la gateway entre le campus et le backbone. Il sauvegarde toutes les informations nécessaires au routage des messages entre les campus, à savoir les adresses MAC source et destination ainsi que les nicknames source et destination. Grâce à ces informations, un BRB peut convertir un message unicast de niveau 1 vers un message unicast de niveau 2 et vice versa. Néanmoins, au sein de MLTP+VNT les BRBs formant la PG ne s'échangent pas ces informations de routage et, donc, si un BRB devient inaccessible, alors les informations de routage qu'il a enregistrées sont perdues.

Un autre problème dû au manque de synchronisation entre les BRBs survient lorsque le chemin de retour diffère du chemin aller, comme abordé dans la Section 4.4.2.4. Dans

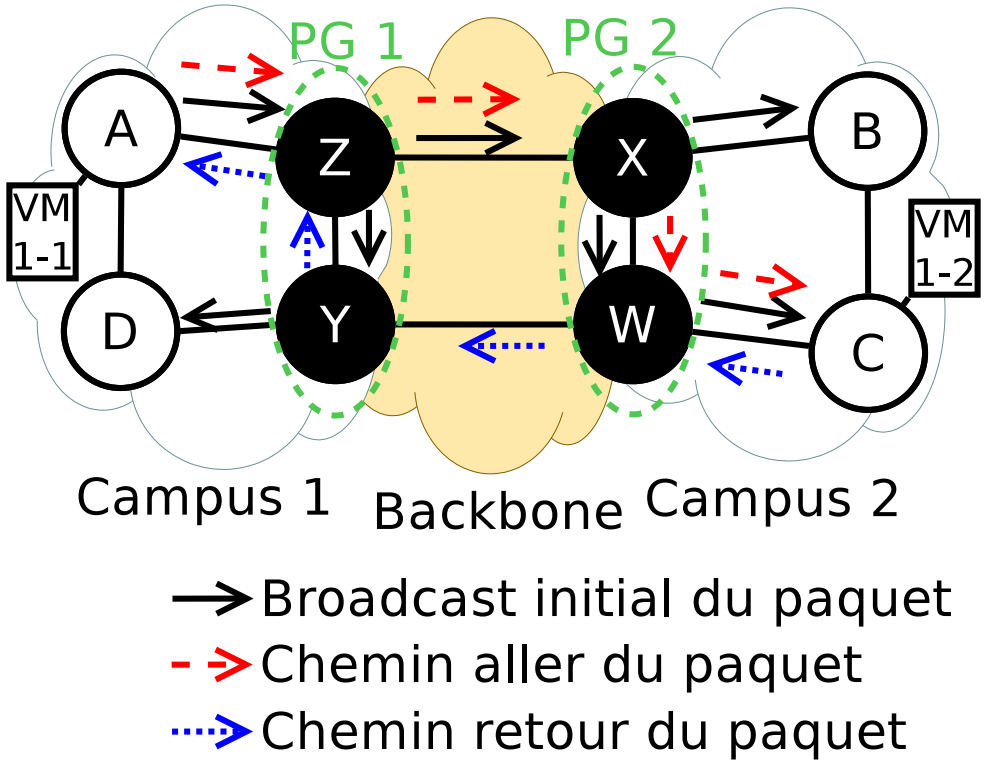


FIGURE 5.1 – Exemple de topologie où les chemins aller et retour diffèrent

ce cas-là, les BRBs sur le chemin aller ne peuvent pas apprendre le nickname associé à l'adresse MAC de destination. Il en résulte que, pour chaque nouveau message envoyé, ces BRBs sont obligés de l'envoyer en broadcast pour inonder le réseau et atteindre la destination. La Figure 5.1 montre une telle situation. La VM 1-1 communique avec la VM 1-2, ce qui est possible car elles possèdent le même VNI, et on remarque que les chemins aller et retour diffèrent. Dans une telle situation, les BRBs Z et X ne connaissent que les informations de la source et non de la destination. Ils sont donc obligés d'inonder le message dans le backbone pour le BRB Z et dans le Campus 2 pour le BRB X, afin d'atteindre la VM 1-2 qui est la destination. De part cette inondation, les BRBs Y et W sont informés et enregistrent l'adresse MAC de la source et le nickname qui lui est associé. Tous les BRBs possèdent alors les informations de la source et sont capables de router la réponse vers celle-ci. Le Tableau 5.1 résume les informations de routage connues par chaque BRB. Comme on peut le voir sur la Figure 5.1 et dans le Tableau 5.1, les BRBs Z et X n'apprennent jamais le nickname associé à l'adresse MAC de la VM 1-2. Ils doivent donc, à chaque fois qu'ils reçoivent un message avec cette adresse MAC, réaliser un broadcast afin d'atteindre la VM 1-2.

5.2 Contribution : Synchronisation des BRBs

Pour résoudre ces problèmes de manque ou de perte d'information de routage nous avons défini une méthode de synchronisation entre les BRBs d'un campus. De cette ma-

BRB	Chemin aller		Chemin retour	
	MAC	Nickname	MAC	Nickname
Z	VM 1-1	A		
X	VM 1-1	PG 1		
W	VM 1-1	PG 1	VM 1-2	C
Y	VM 1-1	A	VM 1-2	PG 2

TABLE 5.1 – Associations adresse MAC - Nickname dans chaque BRB en fonction du chemin

nière, l'ensemble des BRBs possède toutes les informations de routage et tous les BRBs peuvent ainsi router les messages sans recourir au broadcast. On a donc une meilleure gestion du multi-chemin et une meilleure gestion des pannes de BRBs.

Pour réaliser la synchronisation des informations de routage entre les BRBs, nous avons défini un nouveau message de contrôle possédant quatre nouvelles options Type Length Value (TLV). Comme ces messages ne doivent être restreints qu'aux BRBs de la PG, nous avons choisi de créer un nouveau LSP, le LSP de niveau 1&2 (LSP L1&2), qui n'est reconnu que par les BRBs et, donc, les RBridges vont rejeter ce LSP. Le LSP L1&2 est envoyé seulement via les interfaces 1&2 des BRBs pour ne pas propager les informations en dehors de la PG. De plus, les BRBs utilisent leur propre nickname et leur propre identifiant système au sein de ces LSPs L1&2 afin de permettre aux autres BRBs de vérifier que ces informations proviennent bien d'un BRB du même campus. Le LSP L1&2 est composé de la même façon que les LSPs L1 présentés dans la Figure 3.24. Les différences sont la valeur du *IS TYPE* et la partie variable. Le *IS TYPE* contient la nouvelle valeur du L1&2 et la partie variable contient quatre nouvelles TLVs dénommées : *Add-Info*, *Del-Info*, *Syn-Info* et *New-Info*. La TLV *Add-Info* contient la liste des nouveaux couples Adresse MAC / Nickname (MAC/Nickame) à ajouter dans les informations de routage des BRBs. Son format est détaillé dans la Figure 5.2. Elle possède un nouveau type, sur 1 octet, utilisant la valeur 150 pour l'identifier et sa taille, sur 1 octet, exprime le nombre de couples MAC/Nickname. Comme l'adresse MAC possède une taille fixe de 48 bits et que le Nickname possède lui une taille fixe de 16 bits, le couple MAC/Nickname possède alors une taille de 64 bits ce qui correspond à 8 octets. La taille maximale du TLV *Add-Info* est donc de 2048 octets ce qui correspond à 256 couples MAC/Nickname. La seconde TLV *Del-Info* (Figure 5.3) possède le même format que la TLV *Add-Info* et seule la valeur du type est remplacée par 151.

Le LSP L1&2 incluant le TLV *Add-Info* est envoyé par un BRB lorsque celui-ci reçoit un paquet unicast qu'il peut et doit faire transiter d'un niveau à l'autre et dont il ne connaissait ni l'adresse MAC source, ni le nickname source. Dans ce cas-là, le BRB envoie aux autres BRBs de la PG le LSP L1&2 incluant le TLV *Add-Info* contenant l'adresse MAC et le nickname source qu'il vient de découvrir. Dans notre exemple précédent de la Figure 5.1, le BRB W envoie le LSP L1&2 avec le TLV *Add-Info* contenant l'adresse MAC

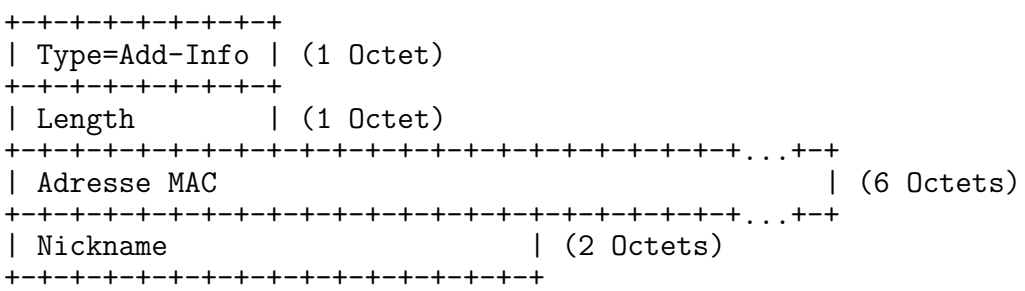


FIGURE 5.2 – Format du TLV Add-Info

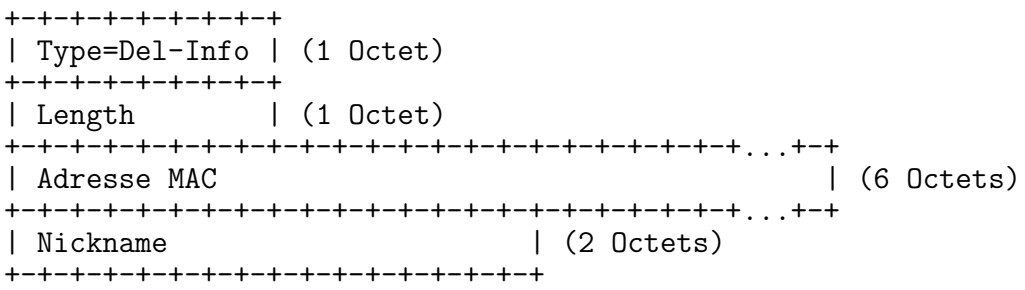


FIGURE 5.3 – Format du TLV Del-Info

de la VM 1-2 et le nickname de C au BRB X lorsqu'il reçoit la première réponse de la VM 1-2 destinée à la VM 1-1. Puis lorsque le BRB Y reçoit la réponse, celui-ci envoie un LSP L1&2, au BRB Z, avec le TLV *Add-Info* contenant l'adresse MAC de la VM 1-2 et le nickname de la PG 2. De cette façon, tous les BRBs du réseau possèdent les informations nécessaires pour router les paquets sans avoir besoin de réaliser de nouveaux broadcasts.

Lorsqu'un BRB reçoit un nouveau couple MAC/Nickname via le TLV *Add-Info*, celui-ci enregistre ces informations ainsi que le RBridge source du LSP L3 sans limite de temps. Seul le BRB qui a envoyé les informations pour ce couple maintient un temps de vie pour celui-ci. Ce temps de vie est réinitialisé à chaque fois que le BRB utilise ces informations. Lorsque le temps de vie expire, alors le BRB envoie un LSP L1&2 avec le TLV *Del-Info* contenant le couple MAC/Nickname à effacer. À la réception de ce LSP, les autres BRBs vérifient localement si le BRB source de ce message de suppression est bien celui qui a informé les autres BRBs à l'origine. Si c'est le cas, alors ils démarrent un temps de vie pour ce couple MAC/Nickname. Si jamais un BRB utilise toujours ce couple MAC/Nickname, il va alors, à son tour, envoyer un LSP L1&2 avec le TLV *Add-Info* pour que les autres BRBs réenregistrent de façon illimitée ces informations.

Dans le cas où un BRB tombe en panne, les autres BRBs s'en aperçoivent après un temps équivalent à un LSP. Dès que les BRBs détectent la panne, ils démarrent tous des temps de vie pour chaque couple MAC/Nickname appris par le BRB tombé en panne. Ils réalisent donc la même action que lorsqu'ils reçoivent le message *Del-Info* sauf qu'ils réalisent cette action pour tous les couples appris via ce BRB. Ensuite, si un autre BRB route le trafic, celui-ci, connaissant les couples MAC/Nickname, va réinitialiser le temps

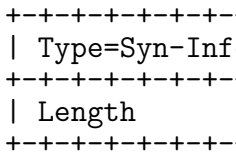


FIGURE 5.4 – Format du TLV Syn-Info

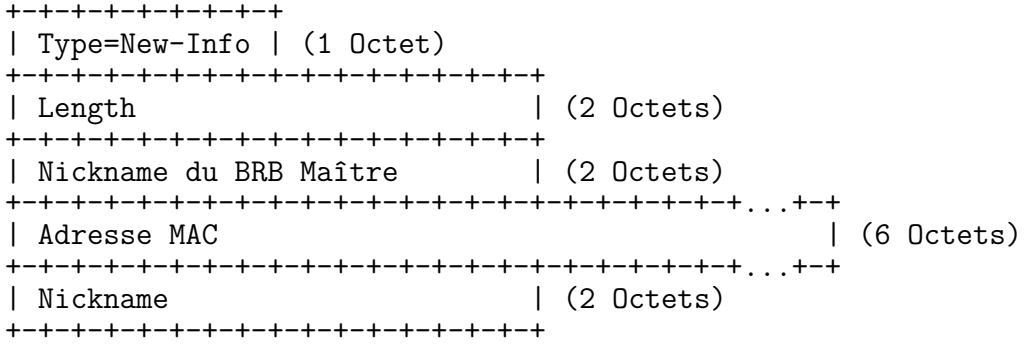


FIGURE 5.5 – Format du TLV New-Info

de vie des couples MAC/Nickname qu'il utilise et envoyer le message *Add-Info* aux autres BRBs pour ces couples.

On a donc défini un système du type Maître/Esclave pour la gestion des couples MAC/Nicknames, où le BRB Maître gère le temps de vie de l'enregistrement et les BRBs Esclaves conservent le couple tant que le BRB Maître ne leur a pas dit de le supprimer. Dans le cas où le BRB Maître d'un couple MAC/Nickname tombe en panne, tous les BRBs Esclaves deviennent des BRBs Maîtres potentiels. Le premier BRB Esclave à recevoir le trafic et à utiliser le couple MAC/Nickname devient le BRB Maître pour ce couple. Il envoie un LSP L1&2 avec le TLV *Add-Info* contenant ce couple MAC/Nickname aux autres BRBs qui redeviennent des BRBs Esclaves pour ce couple.

Les deux autres TLVs : *Syn-Info* et *New-Info* sont utilisés lors du redémarrage ou de l'ajout d'un BRB au campus. Ce BRB doit se synchroniser avec un de ses voisins BRB en lui envoyant un LSP L1&2 contenant le TLV *Syn-Info* dont la valeur type est 152 et dont le format est présenté dans la Figure 5.4. Bien que *Syn-Info* soit un TLV, celui-ci ne possède pas de Valeur car le type du TLV suffit pour que le BRB le recevant comprenne qu'il doit envoyer une copie de l'ensemble des triplets (BRB Maître - adresse MAC - Nickname) qu'il possède. Pour envoyer ces informations, le BRB utilise un LSP L1&2 avec un TLV *New-Info* que nous avons défini (Figure 5.5). La valeur du type de ce TLV *New-Info* est de 153 et sa taille exprime le nombre de triplets qu'il contient, jusqu'à 65536 triplets.

Ce système de BRB Maître et Esclave ainsi que les nouveaux messages de contrôle LSP L1&2 que nous avons définis, nous permettent de fournir une méthode de synchronisation des informations de routage entre les BRBs d'un campus. Ceci permet aux flux inter-

campus de pouvoir utiliser n'importe quel BRB du campus et donc de pouvoir avoir des chemins aller et retour différents. De plus, en cas de panne d'un BRB, les autres BRBs du campus possèdent déjà les informations nécessaires pour router les flux préalablement routés par le BRB tombé en panne et ceci nous permet d'éviter de nombreuses inondations de messages dans le réseau.

5.2.1 Évaluation

Pour évaluer notre solution de synchronisation, nous avons calculé la taille de l'overhead induit par l'ajout d'un BRB et sa synchronisation. Puis nous avons calculé l'overhead dû aux messages de types *Add-Info* et *Del-Info*. Finalement, nous avons réalisé une expérience dans laquelle on teste la reprise du trafic après la panne du BRB qui faisait transiter le trafic entre deux VMs.

5.2.1.1 Plan de contrôle

L'ajout de notre méthode de synchronisation peut poser des problèmes d'overhead lors du démarrage d'un BRB dans une PG qui réalise un grand nombre de communications inter-campus. En effet, pour que le nouveau BRB apprenne l'ensemble des couples MAC/Nickname, il faut qu'un BRB déjà actif lui transmette les informations via des LSP L1&2 avec le TLV *New-Info*. Ces informations sont des triplets (Nickname Maître - MAC - Nickname) d'une taille de 80 bits chacun. On a donc une taille des informations à transmettre égale au nombre de triplets multiplié par 80. Dans la Figure 5.6, on montre l'évolution de cette taille en megabits des informations à échanger lors de la synchronisation d'un BRB en fonction du nombre de triplets. On remarque que, pour 30000 triplets, on doit échanger l'équivalent de 2 Mbits de données, ce qui nécessite un temps de $200\mu s$ sur un lien avec une bande passante de 10 Gbit/s, et représente 0,02% de la capacité du lien pendant 1 seconde. Cependant, comme la taille croit linéairement en fonction du nombre de triplets à échanger, notre solution présentera des problèmes de passage à l'échelle lorsque le nombre de triplets augmentera. Sachant que le nombre maximal de VNIs est de 2^{24} et qu'il n'y a pas de restriction sur le nombre de VMs par VNI, ni sur le placement de ces VMs au sein du réseau, il est possible qu'un campus héberge au moins une VM de chaque VNI et que toutes ces VMs doivent communiquer avec d'autres VMs dans des campus différents. On a donc, au minimum, le double de triplets que de VNIs, à savoir 33554432, à synchroniser lorsqu'un nouveau BRB se connecte. Or, pour un tel nombre de triplets, la synchronisation nécessite l'échange de plus de 2685 Mbits d'informations, ce qui prendrait un temps d'environ $0,27$ seconde et utiliserait environ 27% de la capacité du lien sur une seconde.

Lors du fonctionnement normal du réseau, c'est-à-dire sans la synchronisation d'un nouveau BRB, les BRBs s'échangent, si nécessaire, des messages de type *Add-Info* et *Del-Info* pour synchroniser leurs informations. Dans le pire des cas, chacun de ces messages

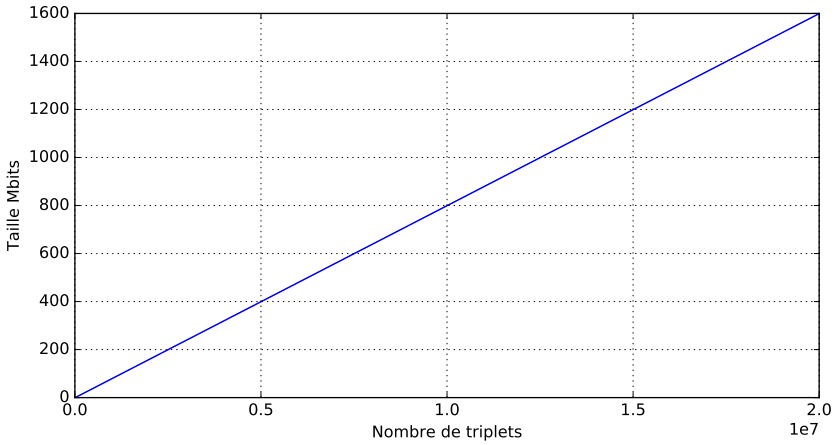


FIGURE 5.6 – Évolution de la taille et du nombre de messages à échanger en fonction du nombre de couples MAC/Nickname

ne contient qu'un seul couple MAC/Nickname et nécessite donc 296 bits en comptant l'en-tête du LSP L1&2. Dans ce cas-là, seulement 338 messages, et donc 338 couples MAC/Nickname, monopolisent 1% de la bande passante, ce qui correspond à un débit de 0,1 Gbit/s. Si les couples MAC/Nickname sont regroupés dans les messages et que l'on en a 256 par message, il est alors possible d'échanger 15360 couples MAC/Nickname toujours en n'utilisant que 1% de la bande passante.

Concernant le temps de convergence des nœuds au sein d'un campus, il n'est pas modifié, et nous obtenons les mêmes résultats que dans la figure 4.13. En effet, notre processus de synchronisation nécessite que le nouveau BRB découvre un BRB adjacent pour lui demander de partager ses informations. Nous n'avons donc pas modifié le processus de découverte du réseau ni la façon dont un BRB s'annonce dans le réseau. De plus, comme précédemment introduit, le temps d'échange des informations ne dépasse pas le temps de convergence du plan de contrôle qui est de 30 secondes, l'équivalent du temps de deux LSP. Ceci permet au BRB de se synchroniser avec un voisin avant que l'ensemble des nœuds ne le voit et, donc, d'être totalement synchronisé avant d'être utilisé pour commuter des paquets.

5.2.1.2 Plan de données

Concernant les performances du plan de données, à savoir la latence et le débit, notre modification n'impacte pas les processus de traitements et de commutations des paquets. Les performances ne sont pas dégradées lors du traitement des paquets par les BRBs. Cependant, comme vu dans la section précédente, l'overhead induit par la synchronisation des BRBs impacte les débits des liens inter-BRB au sein de la PG. Or, si une communication utilise ces liens, son débit sera alors impacté et réduit en conséquence de l'utilisation du lien pour la synchronisation des BRBs.

Nous avons testé les performances de reprise après une panne d'un BRB lorsqu'on

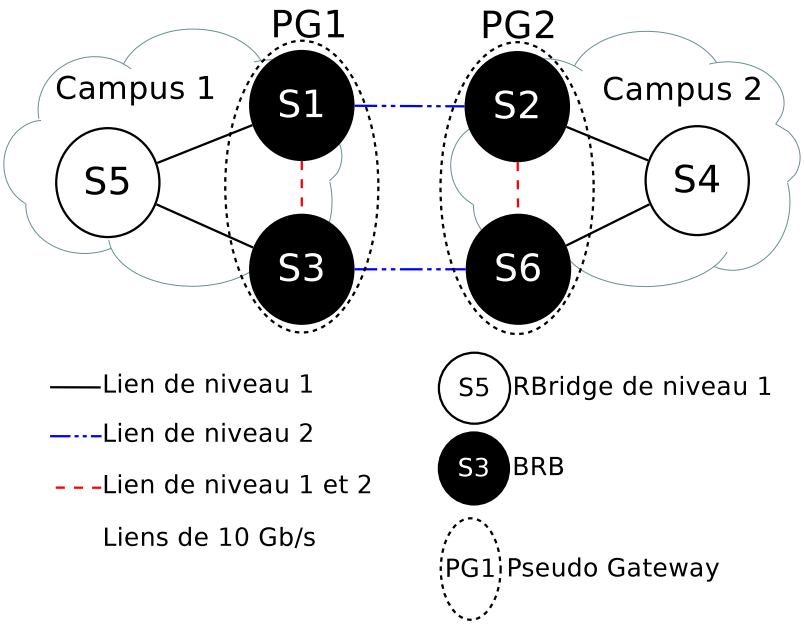


FIGURE 5.7 – Plate-forme de test pour l'étude de la résilience

utilise, ou non, notre solution de synchronisation. À cette fin, nous avons utilisé la plate-forme de test présentée dans la Figure 5.7 et les nœuds S5 et S4 hébergent chacun 30 VMs. Chaque VM de S5 communique avec une VM de S4 et chaque communication possède un débit de 0,1 Gbit/s. Toutes ces communications empruntent le même chemin, le chemin 1 (S5-S1-S2-S4). Le chemin 2 (S5-S3-S6-S4) n'est pour l'instant pas utilisé. On a donc les 30 flux de 0,1 Gbit/s qui empruntent le chemin 1 puis, au bout de 10 secondes, nous simulons la panne du BRB S1 pour étudier le comportement du réseau. Pour cela nous récupérons les traces du trafic dans les BRBs et nous étudions l'évolution des débits à l'aide des résultats de la commande *iperf* que nous avons lancée sur chacune des VMs. Ces résultats sont présentés dans la Figure 5.8 lorsqu'on utilise la synchronisation entre les BRBs et dans la Figure 5.9 lorsque la synchronisation n'est pas utilisée.

La Figure 5.9 montre les résultats obtenus lorsque les BRBs ne sont pas synchronisés. On voit que le trafic commence à être envoyé sur le second chemin après 15 secondes. Cependant, le BRB S3 est obligé de réaliser des inondations pour chacune des 30 communications car il ne possède pas les informations pour les router. Ceci rallonge le temps de reprise après panne et diminue légèrement le débit de chaque communication. Cependant, notre plate-forme de test n'est pas suffisamment grande pour que les dégradations soient importantes.

Dans la Figure 5.8, on remarque que le trafic est envoyé sur le second chemin 15 secondes après la panne du BRB. Le délai de 15 secondes correspond au temps d'un LSP qui est nécessaire pour détecter la panne et ainsi changer le chemin utilisé. De plus, lors du changement de chemin, le BRB S3 possède déjà les informations nécessaires pour router les paquets et n'a pas besoin de réaliser une inondation pour atteindre la destination. Ceci est confirmé dans les traces que nous avons enregistrées où l'on ne retrouve pas de message

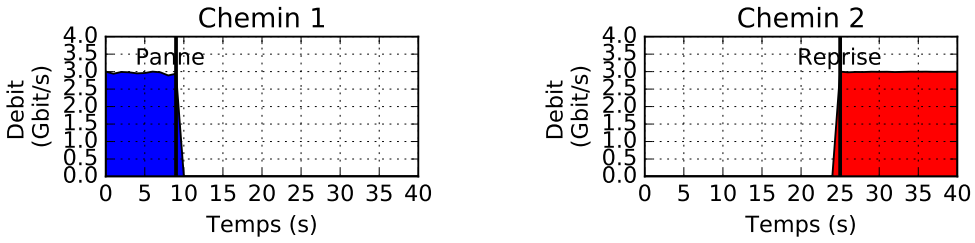


FIGURE 5.8 – Changement de chemin après une panne en utilisant la synchronisation des BRBs

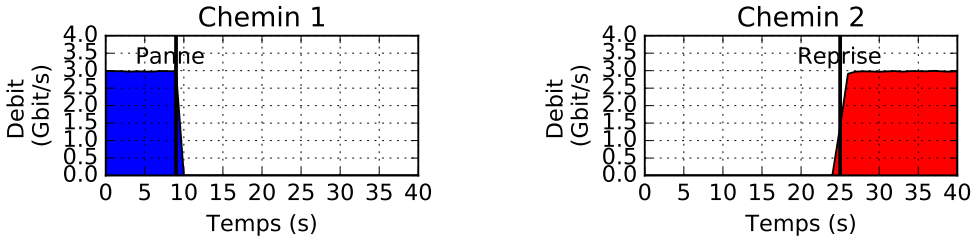


FIGURE 5.9 – Changement de chemin après une panne sans la synchronisation des BRBs

de broadcast mais seulement des messages unicast pour chacune des 30 communications.

5.3 Conclusion

Cette solution de synchronisation nous permet de ne pas avoir de perte d'information de transition entre le campus et le backbone si jamais un BRB tombe en panne. Ceci permet aussi d'éviter que les autres BRBs recevant le trafic qui passait par le BRB tombé en panne ne réalisent autant d'inondations que de paquets qu'ils reçoivent. En effet, un BRB recevant un nouveau flux ne connaît pas les informations MAC/Nickname nécessaires pour faire transiter le paquet et doit donc réaliser une inondation pour atteindre la destination. Notre solution permet d'éviter cette situation en distribuant toutes les informations MAC/Nickname à tous les BRBs de la PG. Chaque BRB annonce les nouveaux couples MAC/Nickname qu'il voit à tous les BRBs de la PG et il prend en charge le temps de vie de ces couples. Lorsque le temps de vie d'un couple expire, le BRB doit alors envoyer un message annonçant aux BRBs de la PG qu'ils doivent supprimer le couple de leurs enregistrements.

En plus de synchroniser les BRBs actifs de la PG entre eux, nous devons partager l'ensemble de ces informations avec un nouveau BRB. Lors du démarrage d'un BRB, celui-ci envoie un message *Syn-Info* à un de ses voisins BRBs qui lui répond en lui envoyant la liste complète des triplets (MAC-Nickname - Nickname Maître). Cependant cette liste peut être conséquente et l'échange de 30000 triplets (MAC - Nickname - Nickname Maître) implique un transfert d'environ 2 Mbits de données. De plus, la synchronisation des BRBs à l'aide des messages que nous avons définis est aussi consommatrice de bande passante.

En effet, dans le pire des cas, pour synchroniser 338 couples MAC/Nicknames, nous avons besoin de 338 LSP L1&2, ce qui implique une taille d'environ 100 Mbits. On remarque que notre solution de synchronisation des BRBs nécessite une infrastructure avec de grandes capacités pour la PG due à l'overhead qu'elle induit. De plus, nous sommes obligés d'attendre 15 secondes, le temps d'un LSP, pour détecter la panne et changer le chemin utilisé. C'est pour cela que nous nous sommes tournés vers le protocole I2RS [71] qui permet de définir un nouveau chemin sans avoir à attendre le temps d'un LSP pour détecter la panne. Nous avons commencé son étude ainsi que le prototypage de son intégration dans une solution TRILL [72].

Chapitre 6

Cloud Hybrid

Sommaire

6.1 Introduction

Comme présenté au début de ce manuscrit dans la section 1, le cloud computing est de plus en plus adopté par les entreprises. Cette adoption est néanmoins focalisée sur le cloud hybride au détriment du cloud public [73] car des craintes subsistent quant à la sécurité des données. L'article [74] nous indique que la sécurité est le principal obstacle à l'adoption du cloud public. Suivent ensuite les craintes sur la perte de propriété des informations envoyées sur le cloud et les craintes que le service ne soit pas à la hauteur des standards de l'entreprise. En raison de ces craintes, le cloud hybride devient l'infrastructure la plus adéquate car il permet à l'entreprise de traiter ses données non vitales au sein d'un cloud public tandis que les données importantes et secrètes restent dans le cloud privé. Le cloud

privé est, soit un data center situé au sein du réseau du client, soit un réseau hébergé par un tiers, comme indiqué dans la définition du cloud privé (Section 2.2). Ce réseau est plus sécurisé que celui d'un cloud public étant donné que les infrastructures du cloud privé ne sont utilisées que par un seul client contrairement au cloud public où elles sont partagées par plusieurs clients. De plus, d'après [73, 75], l'utilisation d'un cloud hybride permet aux clients de conserver le contrôle de leurs données, d'avoir une fiabilité accrue, d'avoir des coûts moindres et de meilleures performances que s'ils n'utilisaient qu'un cloud public ou une infrastructure propre. C'est pour cela que, d'après Gartner [76], la moitié des grandes entreprises va utiliser le cloud hybride en 2017.

6.2 Problématique

La solution MLTP a été développée avec, comme objectif, d'interconnecter des data centers utilisant MLTP et donc de former un cloud. De plus, avec l'intégration de la solution VNT, MLTP+VNT est capable de réaliser l'isolation des utilisateurs. La solution MLTP+VNT permet d'optimiser l'utilisation de l'infrastructure tout en isolant les différents utilisateurs, c'est donc une solution qui a été conçue pour un cloud public. Néanmoins, il est possible d'utiliser MLTP+VNT au sein d'un cloud privé et ainsi de former un cloud hybride avec un campus, formant le cloud privé, interconnecté au cloud public. Dans ce cas de figure, l'utilisateur doit utiliser la solution MLTP+VNT au sein de son cloud privé.

Cependant, nous ne pouvons pas obliger un utilisateur à utiliser la solution MLTP+VNT au sein de son cloud privé. Il nous faut donc trouver une solution pour permettre à cet utilisateur d'interconnecter son cloud privé à notre réseau cloud public.

Cette problématique d'interconnexion de cloud est aussi présente dans les environnements de type *Cloud federation* et *Inter-Cloud* [77]. En effet, dans un cloud fédéré, les fournisseurs de cloud mettent en commun leurs ressources, suivant des règles communes, pour satisfaire les besoins des clients alors que dans l'environnement *Inter-Cloud*, les opérateurs interconnectent leurs clouds sans règle commune. Ces opérateurs n'utilisent pas forcément les mêmes technologies et, comme dans le cas du réseau privé de l'utilisateur, l'opérateur a toujours le choix d'utiliser une solution plutôt qu'une autre.

Pour réaliser l'interconnexion entre le cloud public, utilisant la solution MLTP+VNT, le cloud privé ou encore un autre cloud public, nous avons conçu et développé une passerelle. Nous nous sommes focalisés, dans un premier temps, sur l'interconnexion de MLTP+VNT avec un réseau utilisant le protocole OpenFlow. Le protocole OpenFlow nous a été "imposé" par le second partenaire du projet RAVIR, à savoir Thalès, qui possédait une plate-forme Cloud privé. Nous étions donc dans la situation du cloud hybride et, évidemment, il nous était impossible d'imposer à Thalès de changer de protocole. Le protocole OpenFlow est, par ailleurs, une solution centralisée alors que MLTP+VNT est

décentralisé. La passerelle que nous avons conçue doit donc être capable de communiquer avec les deux protocoles en utilisant les messages de contrôle propres à chacun d'eux sachant qu'elle doit aussi avoir les fonctionnalités nécessaires pour chacun des plans de contrôle.

6.3 Les approches pour interconnecter des clouds

Il existe deux approches pour interconnecter les clouds et dans chacune de ces approches il existe deux solutions [77].

La première approche est l'approche dite *Provider-centric* et c'est le fournisseur de cloud qui initie l'interconnexion des clouds. Dans ce cas, il existe deux solutions qui sont : *Federated Cloud* (la fédération de cloud) et *Hybrid Cloud* (le cloud hybride). La fédération de cloud implique que plusieurs fournisseurs de cloud s'associent, s'accordent sur des règles communes et sur un standard de communication entre leurs infrastructures clouds pour permettre l'interaction entre les clouds. De cette façon, ils peuvent s'échanger des ressources, et ainsi ils pourront être plus à même de satisfaire les demandes de leurs utilisateurs. Cet échange de ressources est réalisé sans que le client ne soit perturbé dans son utilisation du service.

La seconde solution, le cloud hybride, est classée dans l'approche *Provider-centric* car on considère que c'est le fournisseur d'un service cloud privé qui est à l'initiation de l'interconnexion des clouds. En effet, celui-ci à plus d'intérêt à s'interconnecter avec un cloud public tiers plutôt que d'accroître les capacités de sa propre infrastructure. Dans ce cas, le fournisseur de cloud privé réalise des économies au niveau financier et n'a pas à modifier la taille de son infrastructure.

La seconde approche, appelée *Client-centric*, implique que le client est l'initiateur de l'interconnexion des clouds. Il existe aussi deux solutions dans cette approche. La première : le *Multi-Cloud* repose sur le fait qu'un client, utilisateur de solutions cloud public, souhaite utiliser les services de multiples fournisseurs clouds pour de multiples raisons (performance, coûts, cloud souverain, ...). Dans ce cas, le client va donc s'abonner aux services de plusieurs fournisseurs clouds indépendants et il va devoir réaliser lui-même la gestion des accès aux services clouds en utilisant les APIs et commandes de chaque service. Les fournisseurs cloud ne sont pas informés du fait que ce client utilise les services de plusieurs clouds publics et que le client a peut-être réalisé une interconnexion entre ses réseaux virtuels dans chaque cloud.

La dernière solution est l'agrégation de services par un broker (*Aggregated service by Broker*). Le Broker qui s'occupe de réaliser la gestion entre les différentes solutions de chaque fournisseur de cloud est une entité tierce indépendante, ou non, des fournisseurs de cloud. En général, le Broker développe une API qui définit un seul jeu de commandes et va traduire celles-ci dans les formats attendus par chaque cloud. Idem que dans la

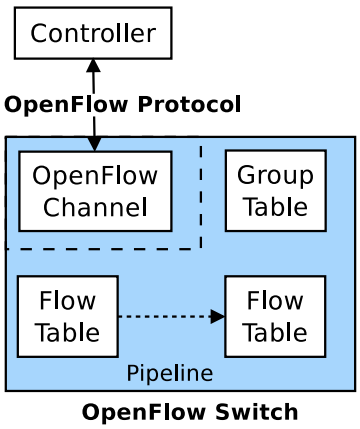


FIGURE 6.1 – Architecture OpenFlow

solution précédente, les fournisseurs de cloud n'ont aucune information quant au fait qu'un utilisateur puisse utiliser plusieurs clouds.

6.4 Contribution : Passerelle MLTP-OpenFlow

Dans un premier temps, nous introduisons le protocole OpenFlow (Section 6.4.1) puis nous présenterons la passerelle MLTP-OpenFlow [15] (Section 6.4.2) ainsi que les résultats expérimentaux que nous avons obtenus (Section 6.4.3).

6.4.1 OpenFlow

OpenFlow est une solution *Software Defined Network (SDN)* centralisée. OpenFlow utilise donc un plan de contrôle centralisé, un unique contrôleur et des commutateurs compatibles OpenFlow. Le contrôleur possède une vue globale de la topologie ainsi que des réseaux virtuels des utilisateurs. Il est responsable de chaque décision de routage et de commutation à l'intérieur du réseau et en informe les équipements compatibles OpenFlow pour qu'ils enregistrent ces décisions. Les commutateurs OpenFlow doivent ensuite appliquer ces décisions à chaque flot, ce qui permet d'avoir des règles pour chaque flot. Si jamais il n'y a pas de règle pour un flot, le commutateur envoie celui-ci au contrôleur qui est le seul élément du réseau à pouvoir décider de la ou des règles à appliquer au flot.

Les commutateurs OpenFlow sont, soit matériel, soit logiciel. Par exemple, Open vSwitch est un commutateur logiciel et virtuel qui est prévu pour les hyperviseurs logiciels. Il permet de fournir la connectivité réseau aux machines virtuelles. Les Open vSwitch utilisent des tables de flots dans lesquelles ils enregistrent les règles qu'ils ont reçues du contrôleur, pour chaque flot. De plus, pour isoler les flots de chaque utilisateur, les commutateurs OpenFlow utilisent des protocoles de tunneling comme les protocoles VLAN ou GRE. Cependant, les Open vSwitch peuvent aussi utiliser le protocole Stateless Transport Tunneling (STT) hérité du développement de Nicira.

L'architecture d'OpenFlow, introduite dans la Figure 6.1, montre le contrôleur connecté à un commutateur OpenFlow. Le contrôleur envoie ses règles à tous les commutateurs OpenFlow au moyen d'un canal sécurisé et le commutateur enregistre ces règles. Cette centralisation du procédé de décision dans un seul équipement logique permet une meilleure gestion de bout-en-bout de chaque flot. Elle permet aussi d'éviter de possibles inconsistances dans le traitement des flots et autorise une rapide mise à jour des règles pour les flots.

6.4.2 Conception de la passerelle

La communication avec un data center peut se faire via deux types de communications : la communication publique et la communication privée. La communication publique regroupe l'ensemble des communications venant du réseau public et n'importe qui peut utiliser ce type de communication pour accéder à un data center. En effet, les clients d'un fournisseur cloud se situent à l'extérieur du réseau du fournisseur cloud et pour qu'ils puissent accéder à leurs VMs, le fournisseur cloud doit leur fournir un accès public à son infrastructure cloud. De plus, cet accès public au cloud est essentiel, pour les clients qui utilisent leurs VMs, afin de fournir des services publics à d'autres usagers qui ne sont pas forcément des clients du fournisseur de cloud. Ces services peuvent être des portails web, des serveurs emails, des forums, des blogs ou alors des espaces proposant les mises à jour pour divers logiciels ou OSs. Ces communications publiques peuvent se dérouler sur des réseaux publics entre les data centers ou alors à l'intérieur d'un data center si les deux clients utilisent le même fournisseur. Cependant, elles nécessitent des routeurs de niveau 3 et des adresses IP routables car elles s'effectuent majoritairement entre les data centers et Internet.

La communication privée n'est accessible que par un seul client. Il est cependant possible d'avoir plusieurs utilisateurs, préalablement autorisés, possédant une communication privée pour un seul client. Cette communication privée permet au client d'administrer ses VMs et de gérer son réseau privé qui interconnecte toutes ses VMs. Ce réseau privé ne peut être vu que par le client qui est isolé des autres clients et de leur réseau privé. Ceci permet à un client de ne voir et de n'accéder qu'à ses propres VMs et de ne pas être vu par les autres clients. Comme le client est l'administrateur de son réseau, il est libre de configurer ses VMs et son réseau comme il le souhaite. Au sein de ce réseau, une partie du trafic est générée par des mécanismes de supervision des VMs, des mécanismes de partage de charge et des mécanismes de distribution du calcul. Cependant, les VMs communiquent aussi entre elles et doivent alors posséder un certain niveau de confiance pour échanger des informations sensibles au sein du réseau privé. Ces échanges peuvent être réalisés au sein d'un data center si toutes les VMs partagent le même data center, ou bien ils peuvent s'opérer entre plusieurs data centers en fonction de la répartition des VMs et du réseau privé de l'utilisateur.

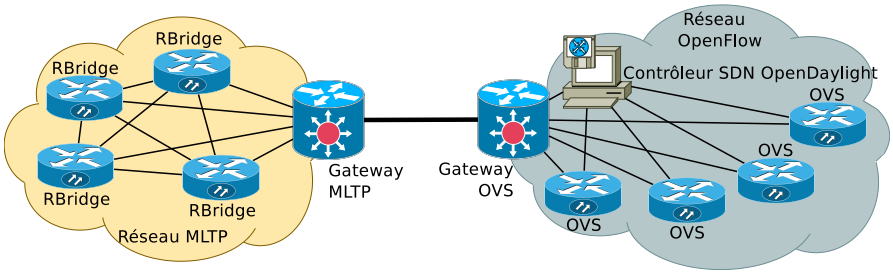


FIGURE 6.2 – Architecture MLTP/OpenFlow

Dans les deux types de communications, il existe la possibilité d'avoir des communications entre data centers. Néanmoins, tous les data centers n'utilisent pas forcément la même technologie et, dans notre cas, nous avons deux technologies qui sont MLTP (Section 4) et OpenFlow (Section 6.4.1). La Figure 6.2 montre l'architecture MLTP/OpenFlow que nous obtenons. Comme il y a deux types de communications, publique et privée, nous devons les gérer lors de l'interconnexion des data centers. Les communications privées entre les deux data centers utilisant des technologies différentes seront réalisées en utilisant un pont de niveau 2, ce qui permet de conserver la compatibilité des trames. Or, dans notre architecture, la passerelle MLTPOpenFlow doit être présente dans chaque gateway MLTP et OpenFlow, comme le montre la Figure 6.2.

Les réseaux MLTP et OpenFlow possèdent des architectures différentes. MLTP est un protocole dont le plan de contrôle est décentralisé et présent dans tous les RBridges du réseau. Au contraire, OpenFlow possède un plan de contrôle centralisé dans un unique contrôleur qui gère l'ensemble des commutateurs OpenFlow du réseau. La gateway OpenFlow est donc connectée au contrôleur et aux Open vSwitch du réseau OpenFlow ainsi qu'à la gateway MLTP qui est connectée aux RBridges. Les deux gateways sont de niveau 2 et permettent de réaliser des communications privées entre data centers sans la nécessité d'avoir des adresses IP publiques pour chaque VM. La topologie du réseau OpenFlow n'est connue qu'au sein de celui-ci. En effet, au niveau du plan de contrôle, les gateways ne s'échangent que les adresses MACs présentes dans les data centers pour pouvoir faire transiter le trafic d'un data center à l'autre.

6.4.2.1 Gateway MLTP

La gateway MLTP est un RBridge comme les autres RBridges au sein du campus MLTP. En effet, celle-ci doit participer, à l'aide des messages de contrôle de niveau 1 et 2, à la découverte de la topologie du campus et du backbone. Cependant, elle a une fonction supplémentaire qui est de réaliser l'interconnexion avec le réseau OpenFlow.

Pour réaliser cette interconnexion, la gateway MLTP sauvegarde les adresses MAC qu'elle a apprises via la gateway OpenFlow et les annonce dans le campus MLTP avec son nickname associé à ces adresses MAC. De cette façon, la gateway MLTP est vue par les RBridges du campus MLTP comme la destination à joindre pour atteindre ces

adresses MACs. Elle joue le rôle de proxy pour l'ensemble des adresses MACs du data center OpenFlow.

La gateway MLTP doit aussi changer l'en-tête des trames MLTP qui sortent du campus à destination du data center OpenFlow. Cela consiste à décapsuler l'en-tête MLTP et à le remplacer par un en-tête Ethernet avec les bonnes informations. Ce nouvel en-tête Ethernet contient, comme adresse MAC source, l'adresse MAC de la gateway MLTP, comme adresse MAC destination celle de la gateway OpenFlow et comme valeur de VLAN la valeur du VNI. L'utilisation du VNI et du VLAN permet d'isoler le flux. Cependant, le nombre de VLAN étant limité, on ne peut réaliser la correspondance VLAN-VNI que pour les 4000 VLANs disponibles. Néanmoins, comme le VNI est codé sur 2^{24} bits comme le VXLAN, il est possible d'augmenter le nombre total de flux isolés à $\approx 16 milions$ tout en conservant la correspondance VNI-VXLAN.

Dans le sens inverse, lorsque les trames rentrent dans le campus MLTP, la gateway doit réaliser le changement d'en-tête inverse, à savoir remplacer l'en-tête Ethernet par un en-tête MLTP. Le changement des informations de routage résulte en un en-tête MLTP avec les informations suivantes :

— Le nickname source est celui de la gateway MLTP.

— Le nickname destination correspond au nickname du RBridge associé à l'adresse MAC destination.

— Le VNI est égale au VLAN.

La Figure 6.3 montre l'évolution des en-têtes du paquet pour sortir et pour rentrer dans le campus MLTP.

6.4.2.2 Gateway OpenFlow

La gateway OpenFlow possède un fonctionnement plus "simple" car elle n'a pas besoin de changer le type de l'en-tête mais seulement de changer les adresses MACs source et destination de l'en-tête Ethernet externe de la trame. Elle n'a pas non plus besoin de sauvegarder les adresses MACs présentes dans le campus MLTP car c'est au contrôleur OpenFlow que revient cette tâche. Elle se doit d'appliquer les règles que le contrôleur lui envoie, à savoir d'envoyer les paquets à destination de certaines adresses MACs vers la gateway MLTP. Pour cela, lorsque les trames sont à destination du campus MLTP, la gateway OpenFlow remplace l'adresse MAC source externe de la trame par la sienne et l'adresse MAC destination externe par celle de la gateway MLTP. Dans l'autre sens, elle remplace l'adresse MAC source par la sienne et l'adresse MAC destination par celle de la destination au sein du réseau OpenFlow. Il en résulte que le contrôleur OpenFlow configure alors l'ensemble du réseau OpenFlow afin de diriger les trames à destination du campus MLTP vers la gateway OpenFlow qui est connectée avec la gateway MLTP.

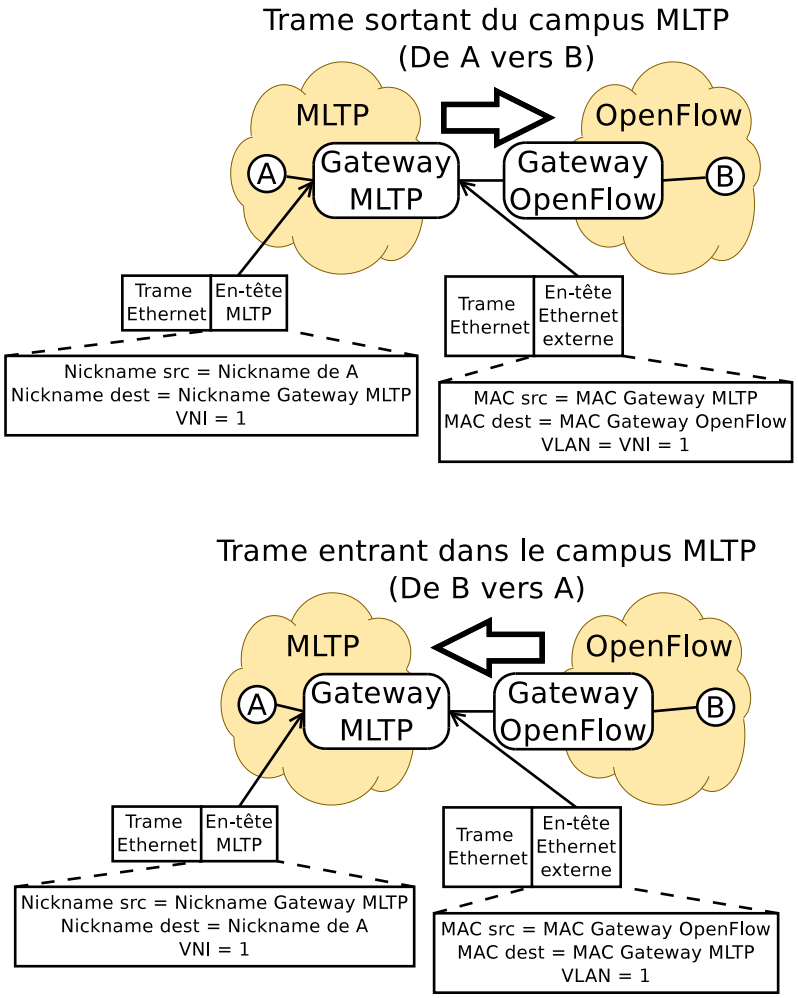


FIGURE 6.3 – Évolution de l'en-tête de la trame

6.4.3 Évaluation de la passerelle

Nous avons développé les deux gateways pour OpenFlow et MLTP afin de réaliser la passerelle MLTP/OpenFlow. Puis nous étudions les performances de la passerelle et présentons les résultats de débit et de latence dans les Sections 6.4.3.2 et 6.4.3.3.

6.4.3.1 Plate-forme de test

La plate-forme de test que nous utilisons est composée de trois serveurs dont deux (Serveurs 1 et 2) sont des DELL C6100 avec un processeur Intel Xeon L5640 @ 2.27GHz et 46GB de RAM. Le troisième serveur (Serveur 3) est un DELL PowerEdge R310 avec un Intel Xeon L3426 @ 1.87GHz et 16GB de RAM. Ces trois serveurs utilisent l'hyperviseur KVM [78] avec comme DOM0 une version modifiée de Linux 3.13.0 et ils sont interconnectés avec des liens de 10 Gbit/s. La plate-forme de test est présentée dans la Figure 6.4.

Le premier serveur héberge une VM ainsi qu'un RBridge logiciel. Ce RBridge est connecté à la gateway MLTP qui est hébergée dans le serveur 2. Ces deux serveurs re-

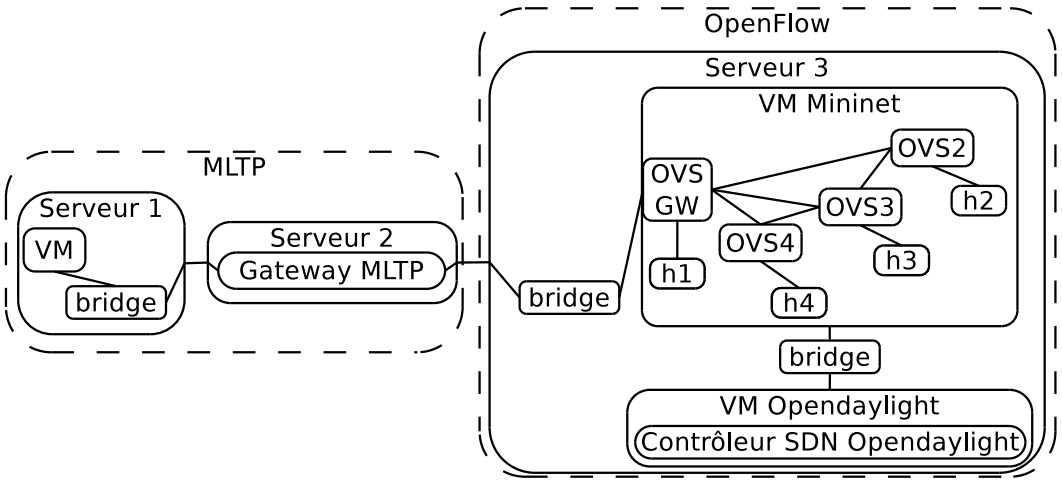


FIGURE 6.4 – Plate-forme de test pour la passerelle MLTP/OpenFlow

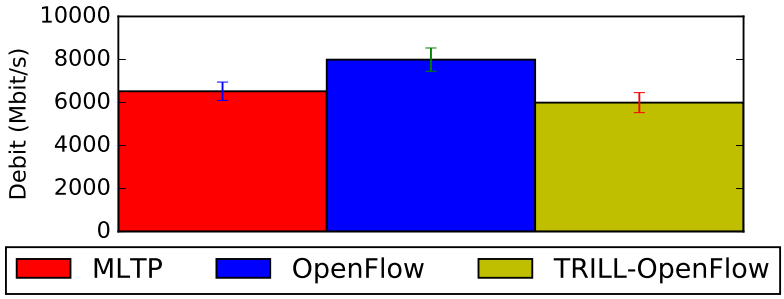


FIGURE 6.5 – Résultats des tests de débit

présentent un data center MLTP. Le serveur 3 héberge l'intégralité du réseau OpenFlow qui est répartie dans deux VMs : la VM Mininet et la VM Opendaylight. La VM Mininet simule, à l'aide de l'émulateur Mininet en version 2.2.1 [79], un réseau OpenFlow avec quatre hôtes (hx), trois Open vSwitch (OVSx) et la gateway OpenFlow (OVS GW). Cette VM est connectée à la VM Opendaylight qui contient le contrôleur SDN Opendaylight [80]. Nous utilisons le contrôleur fourni dans la distribution karaf-0.2.3-Helium-SR3 et dans sa version 3.0.1. Nous utilisons aussi, au sein du contrôleur, les modules suivants : odl-base-all, odl-aaa-authn, odl-restconf, odl-nsf-all, odl-adsal-northbound, odl-mdsal-apidocs, odl-l2switch-switch, odl-dlux-core.

6.4.3.2 Évaluation du débit

Pour tester le débit, nous avons utilisé l'outil *iperf* afin de générer un trafic pendant 50 secondes entre la VM du serveur 1 et h1. Nous avons réalisé ce test 100 fois et avons calculé la moyenne des débits. Notre passerelle MLTP/OpenFlow impacte le débit du trafic inter data centers. En effet, les résultats des tests de débit présentés dans la Figure 6.5 montrent que le débit moyen de la passerelle est approximativement de 6 Gbit/s alors que pour MLTP il est de 6,5 Gbit/s et que pour OpenFlow il est de 8 Gbit/s. Cette dégradation du débit était prévue car la gateway MLTP doit non seulement encapsuler et

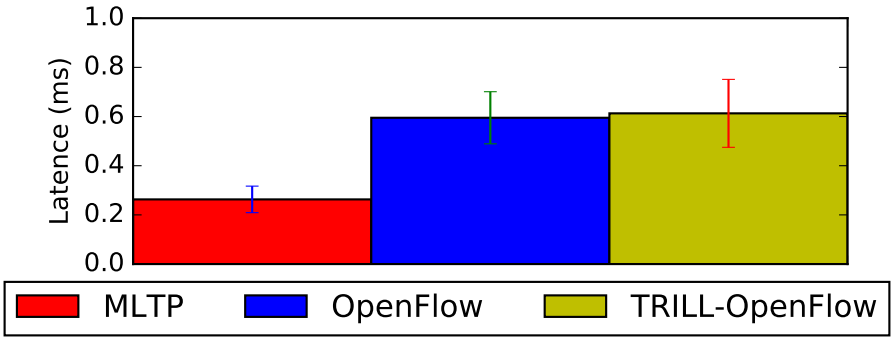


FIGURE 6.6 – Résultats des tests de latence

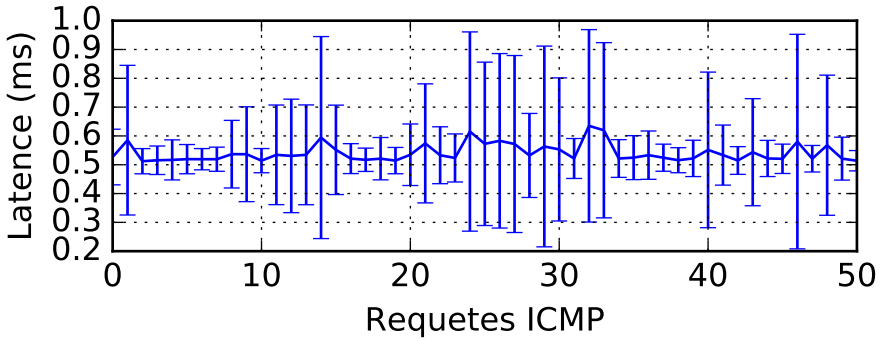


FIGURE 6.7 – Test de latence avec un trafic aléatoire

décapsuler les paquets mais aussi récupérer le nickname correspondant à l'adresse MAC destination. De plus l'ajout de l'en-tête MLTP réduit la taille du payload du paquet et donc réduit le débit. La gateway OpenFlow doit vérifier qu'elle possède les règles nécessaires pour cette communication, les récupérer auprès du contrôleur, si nécessaire, et ensuite les appliquer au flot de données.

6.4.3.3 Évaluation de la latence

Nous étudions la latence au sein de notre plate-forme dans le but de voir l'impact de notre passerelle MLTP/OpenFlow. Pour cela, nous utilisons la commande *ping* entre la VM du serveur 1 et l'hôte h1 dans la VM Mininet hébergée dans le serveur 3. L'expérience se déroule pendant 50 secondes avec une requête ICMP toutes les secondes. Nous réalisons cette expérience 100 fois et calculons la latence moyenne que nous présentons dans la Figure 6.6. La latence moyenne que nous obtenons est d'environ 0,6 ms pour notre passerelle MLTP/OpenFlow. Cette latence est proche de celle d'OpenFlow qui est d'environ 0,59 ms. Cependant, en comparant la latence de la passerelle avec celle de MLTP on remarque que la dégradation est plus importante. En effet, la latence de MLTP est d'environ 0,3 ms. Cette dégradation s'explique car la passerelle doit décapsuler et encapsuler les trames sortantes et entrantes dans le réseau MLTP.

Dans la Figure 6.7, on présente les résultats de la latence obtenue lorsque l'on ajoute un

trafic aléatoire entre la VM et h1. On remarque que la latence n'excède pas 0,65 ms tout au long de l'expérience. Grâce à ces résultats, on peut conclure que la passerelle impacte modérément la latence comparativement aux latences observées pour les solutions MLTP et OpenFlow.

6.5 Conclusion

Avec l'augmentation de la demande pour le cloud hybride, il est nécessaire de concevoir des équipements qui peuvent réaliser l'interconnexion entre un cloud privé et un cloud public qui n'utilisent pas la même technologie. Cette problématique s'étend aussi à l'interconnexion de clouds lors de la réalisation de clouds fédérés ou d'Inter-clouds. C'est dans cette optique que nous avons conçu la passerelle MLTP/OpenFlow. Nous avons réalisé un prototype de la passerelle et nous l'avons testé en utilisant trois serveurs pour réaliser une plate-forme de test. Les tests ont porté sur l'étude de la latence et du débit du trafic lorsque celui-ci utilise la passerelle. Pour la latence, la passerelle implique une dégradation de celle-ci d'environ 1% par rapport à la latence que l'on a obtenue avec OpenFlow. Cependant, la dégradation du débit est plus importante car la passerelle cause une perte d'environ 8% par rapport au débit de MLTP. Ces dégradations s'expliquent par le fait que la passerelle doit modifier les en-têtes des trames en y ajoutant et en y supprimant l'en-tête de MLTP qui possède une taille de 192 bits. De plus, le débit est limité au niveau de MLTP car il n'existe pas, pour l'instant, de *fast path* pour traiter les trames MLTP. L'implémentation d'un tel *fast path* permettrait d'améliorer grandement les performances de la passerelle. Une autre amélioration possible de la passerelle serait de la rendre capable d'interconnecter d'autres technologies telles que VXLAN [68] et NVGRE [81]. Cependant, grâce à notre passerelle MLTP/OpenFlow on peut concevoir la création d'un cloud hybride avec un cloud public MLTP et un cloud privé OpenFlow ou alors interconnecter deux clouds utilisant MLTP et OpenFlow.

Chapitre 7

Conclusion

Sommaire

Le Cloud computing est défini par le NIST (National Institute of Standards and Technology) [16] comme un modèle permettant l'accès simplifié à un ensemble de ressources informatiques configurables telles que des serveurs, des réseaux et des services. Ces ressources doivent pouvoir être rapidement attribuées et libérées avec un minimum d'effort de la part du fournisseur de service. Quatre types de Clouds ont été définis : le cloud privé, le cloud public, le cloud communautaire et le cloud hybride. Le cloud privé permet à un utilisateur d'être le seul utilisateur des ressources et des infrastructures du cloud. Par opposition, le cloud public maximise l'utilisation des infrastructures et des ressources en accueillant de multiples utilisateurs. Le cloud communautaire se situe entre le cloud public et le cloud privé car il partage ses ressources, comme le cloud public, mais restreint l'accès à celles-ci comme le cloud privé. Cependant, à la différence du cloud privé, un cloud communautaire possède plusieurs utilisateurs qui doivent être membres d'une même communauté. Finalement, le dernier type de cloud est le cloud hybride. Le cloud hybride consiste en l'utilisation complémentaire de deux des trois types de cloud présentés précédemment. Cependant, dans la littérature, la majorité des clouds hybrides résulte de l'association d'un cloud public avec un cloud privé. En effet, le cloud communautaire partage la même philosophie d'équilibre entre la confidentialité des données et l'optimisation des coûts, que le cloud hybride. Néanmoins, il possède un désavantage par rapport au cloud hybride qui est le manque de zone privée et donc l'obligation de faire confiance aux autres membres de la communauté qui utilisent la même infrastructure. Or, comme la sécurité est une préoccupation majeure des utilisateurs de cloud et constitue un frein à l'adoption du cloud en général [4], cela explique en partie pourquoi le cloud hybride est le type de cloud qui possède la plus grande adoption parmi les quatre types de clouds.

Le cloud hybride permet aux utilisateurs de sauvegarder leurs données sensibles au sein de leurs clouds privés et de réaliser les opérations nécessitant d'importantes ressources au sein du cloud public dans lequel ils peuvent louer des ressources. Néanmoins, bien que les données envoyées dans le cloud public ne soient pas des données sensibles, elles sont tout de même confidentielles et ne doivent pas être divulguées aux autres utilisateurs du cloud public. C'est pourquoi, il est nécessaire de concevoir une solution qui isole le trafic de chaque utilisateur dans le cloud public tout en maximisant l'utilisation de l'infrastructure de celui-ci.

Nous avons débuté cette thèse CIFRE par l'étude de l'environnement déjà existant au sein de l'infrastructure cloud public de la société GANDI et avons analysé le fonctionnement du protocole TRILL [50] utilisé au sein des data centers. Cette analyse du protocole TRILL nous a permis de relever un manque de fonctionnalités ne permettant pas d'obtenir une isolation réseau dans un environnement Cloud Public/Hybride. Or ces fonctionnalités étaient aussi nécessaires pour l'avancement de deux projets FUI (Fonds Uniques Interministériels) : RAVIR et CARP. Afin de réaliser ces fonctionnalités et pouvoir contribuer à ces deux projets, nous proposons les quatre contributions suivantes.

7.1 Contributions

Interconnexion des data centers

Le protocole TRILL est un protocole de la couche 2 qui permet d'accroître l'utilisation de l'infrastructure du cloud grâce à sa capacité à toujours utiliser le lien le plus court entre chaque nœud. De plus, n'utilisant pas le Spanning Tree Protocol (STP), TRILL conserve et utilise la totalité des liens du réseau tout en évitant la formation de boucles au sein de celui-ci et permet de tirer parti des avantages offerts par des réseaux complètement maillés (full mesh). Le protocole TRILL est donc intéressant pour le réseau d'un data center. Cependant, il ne permet pas d'interconnecter plusieurs réseaux car le protocole TRILL, ayant été conçu pour des réseaux LAN, fusionne tous les réseaux TRILL dès qu'ils sont interconnectés. Or, dans le cas de l'interconnexion de data centers, il est préférable d'éviter d'avoir un seul réseau et un seul domaine de broadcast qui s'étendent sur l'ensemble des data centers et sur le réseau d'interconnexion. Il est donc nécessaire de modifier TRILL pour ne pas réaliser cette fusion et conserver les plans de contrôle de chaque data center indépendant tout en leur permettant de communiquer en utilisant TRILL. L'ajout de cette capacité d'interconnexion de réseau TRILL sans réaliser la fusion de ceux-ci a été le travail de notre première contribution.

Nous avons commencé par étudier l'ensemble des solutions ayant pour objectif d'interconnecter des réseaux TRILL, appelés campus, sans les fusionner. Cependant, aucune des solutions étudiées, lors de la réalisation de l'état de l'art de cette thèse, ne permettait de réaliser cette interconnexion sans perdre l'indépendance des campus. Nous avons donc

modifié le protocole TRILL afin de réaliser l'interconnexion des campus sans les fusionner et, ce, tout en conservant l'indépendance de chaque campus. Cette modification, qui constitue notre première contribution, a été réalisée en deux étapes.

La première étape est la conception du protocole Simple Multi Level Trill Protocol (SMLTP) qui permet d'interconnecter des campus SMLTP tout en conservant le plan de contrôle de chaque campus indépendant. Pour cela, nous avons défini une hiérarchie à deux niveaux dans laquelle le niveau 1 est utilisé par les campus et le niveau 2 par le réseau d'interconnexion des campus, le backbone. Cette hiérarchie s'apparente à la hiérarchie à deux niveaux de IS-IS. Cependant, bien que TRILL soit basé sur IS-IS, il ne possède pas cette notion de niveau ou de hiérarchie. Nous avons donc modifié le protocole TRILL afin d'y ajouter cette notion de hiérarchie qui nous permet d'établir des campus de niveau 1 et un backbone de niveau 2. À l'aide de cette séparation, les messages de contrôle des campus sont des messages de contrôle de niveau 1 et ne peuvent pas accéder au backbone et, par extension, ne peuvent pas accéder aux autres campus. Grâce à ce backbone entre les campus, SMLTP permet d'interconnecter les campus tout en conservant les plans de contrôle de chaque campus indépendant.

Pour réaliser la connexion entre le backbone et un campus nous avons défini un nouvel équipement appelé Border RBridge (BRB). Ce BRB appartient aux deux niveaux et s'occupe de réaliser la séparation entre le niveau 1, le campus, et le niveau 2, le backbone. Il a aussi pour tâche de faire transiter les paquets de données entre le campus et le backbone et vice versa.

SMLTP permet aussi d'augmenter le nombre total de RBridges (nœuds TRILL) au sein du réseau. Dans TRILL, chaque RBridge possède un nickname (un identifiant) unique dans le réseau. Ce nickname est codé sur 16 bits et $2^{16} = 65536$ nicknames sont disponibles. Dans SMLTP, nous avons divisé cet ensemble de nickname en deux groupes en utilisant le bit de poids fort. Chaque groupe contient $2^{15} = 32768$ nicknames. Sachant que le premier groupe contient les nicknames de niveau 1 utilisés uniquement par les RBridges de niveau 1 qui se situent dans les campus, on a donc 32768 nicknames réutilisables dans tous les campus du réseau. En effet, grâce à l'indépendance des campus, les nicknames utilisés dans un campus peuvent être réutilisés dans tous les autres campus. Par opposition, les nicknames du second groupe, les nicknames de niveau 2, ne sont attribués qu'aux BRBs et doivent être uniques dans le réseau.

La solution SMLTP possède cependant un inconvénient qui est l'impossibilité d'avoir plus d'un BRB par campus. Pour remédier à cela nous avons développé Multi Level Trill Protocol (MLTP) qui est la seconde étape de notre première contribution. MLTP définit un nouvel équipement logique appelé Pseudo Gateway (PG) et un nouveau niveau hiérarchique, le niveau 1&2. Cette PG regroupe l'ensemble des BRBs d'un campus et tous ces BRBs utilisent les informations de la PG pour communiquer avec les RBridges. De cette façon, il est possible d'utiliser plusieurs BRBs comme gateway par campus et chacun

de ces BRBs peut faire transiter les paquets de données entre le campus et le backbone. Le nouveau niveau 1&2 est utilisé uniquement par les BRBs au sein de la PG afin que les BRBs puissent se découvrir en utilisant leurs propres informations (adresse MAC et nickname). MLTP permet donc de ne "montrer" qu'une seule gateway par campus bien que celle-ci soit, en réalité, constituée de tous les BRBs du campus.

Isolation des flux inter et intra data centers

Après avoir modifié le protocole TRILL en MLTP (Multi Level Trill Protocol) pour pouvoir interconnecter plusieurs data centers et, donc, concevoir un cloud public formé de plusieurs data centers indépendants, nous nous sommes focalisés sur l'ajout à MLTP d'une méthode d'isolation du trafic des utilisateurs. Cette isolation du trafic est nécessaire afin de répondre à l'une des préoccupations majeures des utilisateurs du cloud, à savoir la sécurité de leurs informations au sein du cloud [4]. Afin de concevoir la solution d'isolation la mieux adaptée à notre besoin, à savoir un grand nombre d'utilisateurs à isoler, nous avons réalisé une veille technologique sur les solutions d'isolation les plus importantes (Annexe A). À la suite de cette veille technologique, nous avons déterminé que la solution Virtual Network over Trill (VNT) [14, 43] était la solution la mieux adaptée à nos besoins. Elle permet d'avoir un grand nombre (2^{24}) de Virtual Network Identifier (VNI), des identifiants uniques et elle s'adapte au protocole TRILL tout en conservant la compatibilité avec les équipements TRILL. Notre seconde contribution a consisté en l'adaptation de la solution VNT au sein de MLTP.

Afin d'intégrer la solution VNT dans MLTP, nous avons modifié le plan de contrôle de MLTP en y ajoutant les trois éléments suivants :

— Un automate à états pour les VNIs.

— Le processus de découverte des VNIs.

— L'algorithme de calcul des VNIs.

De plus, les processus de réception des paquets unicast et multicast ont été modifiés pour prendre en compte le contrôle du VNI lors du traitement du paquet. L'ensemble de ces éléments permet à chaque RBridge de sélectionner l'action adéquate lorsqu'il reçoit un paquet avec un VNI. Le RBridge est ainsi capable pour un VNI donné de savoir s'il doit faire transiter le paquet, l'accepter ou alors le rejeter.

L'implémentation de la solution VNT au sein de MLTP a permis d'isoler le trafic de chaque utilisateur dans le réseau en attribuant un VNI à chacun. De cette façon, chaque utilisateur possède son propre réseau virtuel au sein du réseau cloud. Cependant, cette implémentation de MLTP+VNT implique un contrôle du VNI s'effectuant à la réception du paquet et implique que le trafic multicast de chaque VNI est envoyé sur le backbone à destination de tous les campus, même s'il n'y a pas de destinataire dans le campus. Pour éviter de charger inutilement le backbone, nous avons ajouté un contrôle sur le VNI avant

l'envoi du paquet dans le backbone. À l'aide de cette vérification sur le VNI, le BRB peut savoir si d'autres campus acceptent le VNI et il enverra le flux de ce VNI seulement à ces campus. Ceci permet de réduire la charge du backbone en supprimant du trafic inutile.

Amélioration de reprise après panne d'un BRB

De par la conception de MLTP+VNT, les BRBs sont les seuls à posséder les informations de routage pour les flux inter-campus. De ce fait, la panne d'un BRB interrompt l'ensemble du trafic inter-campus qui l'utilisait comme gateway durant le temps nécessaire pour la détection de la panne ; lequel correspond au temps d'un LSP. Suite à la détection de la panne, chaque flux qui utilisait ce BRB comme gateway est envoyé vers un autre BRB qui est le BRB le plus proche de la source des flux. Cependant, lorsque le trafic d'un flux atteint un nouveau BRB, ce BRB ne possède pas les informations nécessaires pour réaliser son routage. Le BRB est obligé de réaliser une inondation dans le réseau, restreint aux campus acceptant le VNI du flux, pour trouver la destination et récupérer le nickname associé à cette destination.

Pour éviter de perdre les informations de routage enregistrées par un BRB lorsque celui-ci tombe en panne et pour éviter des inondations massives de la part des autres BRBs de la PG, nous avons conçu une méthode de synchronisation entre les BRBs d'une PG. La synchronisation des informations de routage inter-campus, à savoir des couples adresse MAC/Nickname, entre les BRBs d'une PG repose sur un procédé du type Maître-Esclave. Un BRB, qui découvre un nouveau couple adresse MAC/Nickname, est le maître de ce couple et il doit l'annoncer aux autres BRBs. Il est aussi responsable du temps de vie de ce couple et devra annoncer aux autres BRBs la suppression du couple lorsque son temps de vie expirera. Lorsque le BRB maître envoie le message de suppression du couple MAC/Nickname, les autres BRBs ne suppriment pas immédiatement le couple mais démarrent un temps de vie pour ce couple. Cela est nécessaire pour éviter de supprimer des informations qui pourraient être en cours d'utilisation. En effet, le premier BRB à avoir vu le couple peut ne plus être le BRB utilisé comme gateway par le flux. Il faut, pour cela, laisser du temps à ce second BRB afin qu'il puisse détecter qu'il est le nouveau maître pour ce couple et informer les autres BRBs sans avoir à réaliser une inondation pour retrouver le nickname associé à l'adresse MAC.

Dans le cas où un BRB tombe en panne, les autres BRBs possèdent les informations de routage qu'il utilisait et peuvent donc router les nouveaux flux qui passeront par eux. Cependant, au moment de la détection de la panne du BRB, tous les autres BRBs réagissent comme s'ils avaient reçu un message de suppression pour tous les couples dont le maître était le BRB tombé en panne. Cela permet à chaque BRB de lancer un temps de vie sur chaque couple MAC/Nickname et, donc, le premier BRB à voir un couple devient le maître de ce couple. De cette façon les couples qui n'étaient pas utilisés seront supprimés lorsque leur temps de vie expirera et les couples encore utilisés seront distribués sur les

autre BRBs de la PG.

Grâce à la synchronisation des BRBs d'une PG, on évite la perte d'informations de routage lorsqu'un BRB tombe en panne. On évite aussi des inondations de la part des autres BRBs pour retrouver ces informations. Cependant, la détection d'une panne s'effectue dans un temps équivalent à un LSP, c'est à dire qu'il faut attendre 15 secondes pour détecter la panne et donc rediriger le trafic sur un autre chemin.

Cloud Hybrid

Notre solution MLTP+VNT permet de fournir l'isolation dans un cloud public et dans un cloud hybride. En effet, il est possible d'utiliser MLTP+VNT dans un cloud privé et de connecter ce campus au cloud public qui lui aussi utilise MLTP+VNT. Cependant, on ne peut pas imposer à un client d'utiliser notre solution MLTP+VNT dans son réseau cloud privé. Il nous faut donc proposer une solution pour pouvoir interconnecter des réseaux utilisant d'autres technologies à notre réseau MLTP+VNT. Pour cela nous proposons dans notre quatrième contribution une gateway permettant d'interconnecter un réseau OpenFlow à notre réseau MLTP+VNT. Cette passerelle gère les deux types de communications que sont les communications publiques et les communications privées. Pour cela, la passerelle MLTP-OpenFlow se situe sur les gateways du réseau OpenFlow et sur les BRBs du réseau MLTP+VNT.

La partie qui se situe sur les BRBs doit enregistrer l'ensemble des adresses MAC atteignables dans le réseau OpenFlow ainsi que décapsuler les trames MLTP+VNT avant de les envoyer comme paquets IP à la gateway OpenFlow qui doit appliquer les règles adéquates à ces paquets pour les acheminer à destination. Lorsque le paquet quitte le réseau OpenFlow et atteint la gateway MLTP+VNT, celle-ci doit encapsuler le paquet dans une trame MLTP+VNT en lui ajoutant l'en-tête avec les informations nécessaires pour l'acheminement de la trame, à savoir le nickname et le VNI.

Bien que la passerelle MLTP-OpenFlow que nous avons développée ait des performances limitées par la technologie la moins performante, elle nous permet de réaliser l'interconnexion d'un réseau MLTP+VNT et d'un réseau OpenFlow, tant pour réaliser un cloud hybride que pour établir un environnement Inter-Cloud.

Intégration des contributions dans l'infrastructure de GANDI

La société GANDI est une PME française spécialisée dans la Gestion et l'Administration des Noms de Domaines Internet et agréée par l'ICANN [82]. Elle possède aussi plusieurs data centers en Europe, dont 2 en France, 1 au Luxembourg et 1 aux États-Unis d'Amérique, afin de pouvoir proposer à ses clients des services IaaS et PaaS. Ces services comptabilisent plusieurs dizaines de milliers de clients et utilisent environs 300 serveurs ainsi que 120 RBridges. Dans le cadre de cette thèse, l'ensemble des contributions

a été testé et déployé dans un environnement de production au sein des infrastructures de GANDI en suivant le processus de production de la société.

7.2 Perspectives

Les travaux de cette thèse fournissent de nouvelles fonctionnalités au protocole TRILL. Ils permettent également de définir de nouvelles orientations pour de futurs travaux. Une partie de ceux-ci est présentée ci-après.

L'évaluation des performances de notre solution MLTP+VNT montre qu'elle pâtit de l'absence d'un fast path pour traiter les paquets de données. En effet, le fait de passer par le noyau Linux pour analyser et modifier les en-têtes de MLTP+VNT impacte les performances en terme de débit et de latence. Cet impact est plus particulièrement dû aux limitations sur la fréquence mémoire [83] des nœuds physiques. Afin de remédier à cela, nous avons réalisé une veille technologique sur les différentes solutions de traitement des paquets. Suite à cette veille et au sein du projet CARP, nous nous sommes orientés vers le développement, sur une carte réseaux programmable, d'un fast path pour des solutions TRILL [84].

Concernant la synchronisation des BRBs, nous avons remarqué que le temps de reprise après une panne pouvait être amélioré grâce à notre solution. Néanmoins, la partie la plus importante de l'interruption du trafic n'est pas résolue par notre solution. En effet, notre synchronisation permet de router correctement le trafic dès que celui-ci emprunte un nouveau BRB. Cependant, le temps nécessaire à la détection d'une panne n'est pas réduit et correspond au temps d'un LSP, à savoir 15 secondes. Il nous faut donc trouver une solution pour réduire ce temps d'interruption du trafic. Pour cela, nous avons commencé à étudier la solution I2RS [71] qui permet de calculer un nouveau chemin et d'en informer les nœuds avant d'attendre le temps d'un LSP et nous avons commencé à la tester avec une solution TRILL [72].

Finalement, notre passerelle MLTP-OpenFlow permet d'interconnecter deux clouds utilisant OpenFlow et MLTP+VNT mais, au vu de l'essor du cloud hybride, de nombreuses autres technologies seront utilisées. Il nous est donc impératif de penser à l'évolution de cette passerelle pour pouvoir accepter ces nouvelles technologies comme VXLAN et NVGRE. Cependant, nous ne devons pas nous limiter à ces deux technologies et continuer le développement de notre passerelle pour accommoder de futures technologies d'isolation réseau quel que soit leur niveau de fonctionnement (L2 ou L3).

Publications

Journaux internationaux

[1] V. Del Piccolo, A. Amamou, K. Haddadou and G. Pujolle, "A Survey of Network Isolation Solutions for Multi-Tenant Data Centers," in IEEE Communications Surveys & Tutorials, vol. 18, no. 4, pp. 2787-2821, Fourthquarter 2016. doi : 10.1109/COMST.2016.2556979

Conférences internationales

[2] V. Del Piccolo, A. Amamou, W. Dauchy and K. Haddadou, "Multi-tenant isolation in a TRILL based multi-campus network," 2015 IEEE 4th International Conference on Cloud Networking (CloudNet), Niagara Falls, ON, 2015, pp. 51-57. doi : 10.1109/CloudNet.2015.7335279

[3] V. Del Piccolo, A. Amamou, B. Vidalenc, M. Bouet and K. Hadaddou, "Design and analysis of a TRILL - OpenFlow bridge," 2016 IEEE Global Communications Conference (GLOBECOM), Washington (DC), 2016.

[4] M. Bah, V. Del Piccolo, M. Bourguiba and K. Haddadou, "A centralized controller to improve fault tolerance in TRILL-based fabric networks," 2016 IEEE 3rd Smart Cloud Networks & Systems Conference (SCNS), Dubai, UAE, 2016.

[5] D. Cerovic, V. Del Piccolo, A. Amamou and K. Haddadou, "Offloading TRILL on a programmable card," 2016 IEEE 3rd Smart Cloud Networks & Systems Conference (SCNS), Dubai, UAE, 2016.

---

[6] V. Del Piccolo, A. Amamou, K. Haddadou and G. Pujolle, "Tenant Isolation in a TRILL-Multi-Campus Network," Under review for IEEE Transactions on Network and Service Management.

Bibliographie

[1] Enterprise cloud adoption survey 2014 : Summary of results. Technical report, Everest Global, Inc., 2014. (Cité en page 1.)

[2] Kim Weins. Cloud computing trends : 2014 state of the cloud survey. Technical report, RigthScale, 2014. (Cité en pages 1 et 2.)

[3] Kim Weins. Cloud computing trends : 2015 state of the cloud survey. Technical report, RigthScale, 2015. (Cité en page 2.)

[4] Kim Weins. Cloud computing trends : 2016 state of the cloud survey. Technical report, RigthScale, 2016. (Cité en pages 2, 123 et 126.)

[5] Chris Bateman. Survey results : The rise in hybrid cloud adoption speaks volumes on public cloud advancements. Technical report, Datapipe, Inc, 2016. (Cité en page 2.)

[6] Saroj Kar. In five years, the market for hybrid cloud computing is worth $84.67 billion. Technical report, CloudTimes, 2015. (Cité en page 2.)

[7] Hybrid cloud market by solution (cloud management and orchestration, disaster recovery, security and compliance, and hybrid hosting), by service (professional services and managed services), by service model (iaas, paas, and saas) - global forecast to 202. Technical report, marketsandmarkets, 2016. (Cité en page 2.)

[8] Cisco global cloud index : Forecast and methodology, 2014-2019. Technical report, Cisco Systems, Inc, 2016. (Cité en page 2.)

[9] Dino Farinacci, Tony Li, Stan Hanks, David Meyer, and Paul Traina. Generic routing encapsulation (gre). RFC 2784, March 2000. (Cité en pages 2, 11 et 12.)

[10] Institute of Electrical and Electronics Engineers. Virtual bridged local area networks. IEEE Standard 802.1Q, 2005 Edition, May 2006. (Cité en pages 2, 11 et 13.)

[11] Radia Perlman, Donald E. Eastlake 3rd, Dinesh G. Dutt, Silvano Gai, and Anoop Ghanwani. Routing bridges (rbridges) : Base protocol specification. RFC 6325, July 2011. (Cité en pages 4, 17, 26, 33 et 55.)

[12] V. Del Piccolo, A. Amamou, W. Dauchy, and K. Haddadou. Multi-tenant isolation in a trill based multi-campus network. In *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*, pages 51–57, Oct 2015. (Cité en pages 4 et 60.)

[13] V. Del Piccolo, A. Amamou, K. Haddadou, and G. Pujolle. A survey of network isolation solutions for multi-tenant data centers. *IEEE Communications Surveys Tutorials*, 2016. (Cité en pages 4, 10 et 74.)

[14] Ahmed Amamou. Isolation réseau dans un datacenter virtualisée, 2013. Thèse, Université Pierre et Marie Curie - Paris 6 - EDITE de Paris. (Cité en pages 5, 8, 27, 55, 76, 79 et 126.)

[15] V. Del Piccolo, A. Amamou, B. Vidalenc, M. Bouet, and K. Haddadou. Design and analysis of a trill - openflow bridge. In *Global Telecommunications Conference (GLOBECOM 2016), 2016 IEEE*, 2016. (Cité en pages 5 et 114.)

[16] NIST. Nist cloud computing program. `http://www.nist.gov/itl/cloud/index.cfm`. (Cité en pages 8 et 123.)

[17] Gartner. Gartner it glossary - data center. `http://www.gartner.com/it-glossary/data-center/`. (Cité en page 10.)

[18] Michael Bullock. Data center definition and solutions. `http://www.cio.com/article/499671/Data_Center_Definition_and_Solutions`, Août 2009. (Cité en page 10.)

[19] Cisco virtualized multi-tenant data center, version 2.2 design guide. Technical report, Cisco Systems, Inc, 2012. (Cité en page 10.)

[20] Securing multi-tenancy and cloud computing. Technical report, Juniper Networks, Inc, 2012. (Cité en page 10.)

[21] Steve Bobrowski. The force.com multitenant architecture. Technical report, salesforce.com, inc, 2013. (Cité en page 10.)

[22] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn. Point-to-point tunneling protocol (pptp). RFC 2637, July 1999. (Cité en pages 11 et 12.)

[23] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter. Layer two tunneling protocol "l2tp". RFC 2661, August 1999. (Cité en pages 11 et 12.)

[24] J. Lau, M. Townsley, and I. Goyret. Layer two tunneling protocol - version 3 (l2tpv3). RFC 3931, March 2005. (Cité en pages 11 et 12.)

[25] S. Bryant and P. Pate. Pseudo wire emulation edge-to-edge (pwe3) architecture. RFC 3985, March 2005. (Cité en pages 11 et 12.)

[26] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. RFC 3031, January 2001. (Cité en pages 11 et 13.)

[27] L. Berger. Generalized multi-protocol label switching (gmpls) signaling functional description. RFC 3471, January 2003. (Cité en pages 11 et 13.)

[28] E. Rosen and Y. Rekhter. Bgp/mpls ip virtual private networks (vpns). RFC 4364, February 2006. (Cité en pages 11 et 13.)

[29] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. The locator/id separation protocol (lisp). RFC 6830, January 2013. (Cité en pages 11 et 12.)

[30] Murari Sridharan, Yu-Shun Wang, Albert Greenberg, Pankaj Garg, Narasimhan Venkataramiah, Kenneth Duda, Ilango Ganga, Geng Lin, Mark Pearson, Patricia Thaler, and Chait Tumuluri. Nvgre : Network virtualization using generic routing encapsulation. Work in progress, draft-sridharan-virtualization-nvgre-04, February 2014. (Cité en pages 11, 12 et 93.)

[31] Bruce Davie and Jesse Gross. A stateless transport tunneling protocol for network virtualization (stt). Work in progress, draft-davie-stt-06, April 2014. (Cité en pages 11 et 12.)

[32] Institute of Electrical and Electronics Engineers. Ieee 802.1ad-2005. 802.1ad - Virtual Bridged Local Area Networks, 2005. (Cité en pages 11 et 13.)

[33] Institute of Electrical and Electronics Engineers. Ieee 802.1ah-2008. 802.1ah - Provider Backbone Bridges, 2008. (Cité en pages 11 et 13.)

[34] Marco Foschiano and Sanjib HomChaudhuri. Cisco systems's private vlans : Scalable security in a multi-client environment. RFC 5517, February 2010. (Cité en pages 11 et 13.)

[35] Cisco. *Cisco Active Network Abstraction Reference Guide, 3.7*, June 2010. Part 2 - Technology Support and Information Model Objects : Virtual Routing and Forwarding. (Cité en pages 11 et 13.)

[36] Mallik Mahalingam, Dinesh G. Dutt, Kenneth Duda, Puneet Agarwal, Lawrence Kreeger, T. Sridhar, Mike Bursell, and Chris Wright. Vxlan : A framework for overlaying virtualized layer 2 networks over layer 3 networks. Work in progress, draft-mahalingam-dutt-dcops-vxlan-09, April 2014. (Cité en pages 11 et 13.)

[37] Aled Edwards, Anna Fischer, and Antonio Lain. Diverter : A new approach to networking within virtualized infrastructures. In *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, WREN '09, pages 103–110, New York, NY, USA, 2009. ACM. (Cité en page 11.)

[38] Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. Portland : A scalable fault-tolerant layer 2 data center network fabric. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pages 39–50, New York, NY, USA, 2009. ACM. (Cité en page 11.)

[39] Fang Hao, T. V. Lakshman, Sarit Mukherjee, and Haoyu Song. Secure cloud computing with a virtualized network infrastructure. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 16–16, Berkeley, CA, USA, 2010. USENIX Association. (Cité en page 11.)

[40] Saurabh Barjatiya and Prasad Saripalli. Blueshield : A layer 2 appliance for enhanced isolation and security hardening among multi-tenant cloud workloads. In *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*, UCC '12, pages 195–198, Washington, DC, USA, 2012. IEEE Computer Society. (Cité en page 11.)

[41] Li Li and Thomas Woo. Vsite : A scalable and secure architecture for seamless l2 enterprise extension in the cloud. In *Secure Network Protocols (NPSec), 2010 6th IEEE Workshop on*, pages 31–36. IEEE, 2010. (Cité en page 11.)

[42] Jayaram Mudigonda, Praveen Yalagandula, Jeff Mogul, Bryan Stiekes, and Yanick Pouffary. Netlord : A scalable multi-tenant network architecture for virtualized datacenters. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pages 62–73, New York, NY, USA, 2011. ACM. (Cité en page 11.)

[43] Ahmed Amamou, Kamel Haddadou, and Guy Pujolle. A trill-based multi-tenant data center network. *Computer Networks*, 2014. (Cité en pages 11, 27 et 126.)

[44] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. Vl2 : A scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pages 51–62, New York, NY, USA, 2009. ACM. (Cité en page 11.)

[45] Liane Lewin-Eytan, Katherine Barabash, Rami Cohen, Vinit Jain, and Anna Levin. Designing modular overlay solutions for network virtualization. Technical report, IBM, 2011. (Cité en page 11.)

[46] Radia Perlman. Rbridges : Transparent routing. In *Proceedings of the IEEE INFOCOMM 2004*, INFOCOMM '04, 2004. (Cité en page 17.)

[47] David R. Oran. Osi is-is intra-domain routing protocol. RFC 1142, February 1990. (Cité en page 21.)

[48] *Information technology – Telecommunications and information exchange between systems – Intermediate system to Intermediate system intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service*. (ISO 8475), ISO/IEC 10589, 1992. (Cité en page 21.)

[49] Ayan Banerjee and David Ward. Extensions to is-is for layer-2 systems. RFC 6165, April 2011. (Cité en pages 25, 26 et 27.)

[50] Donald E. Eastlake 3rd, Tissa Senevirathne, Anoop Ghanwani, Dinesh Dutt, and Ayan Banerjee. Transparent interconnection of lots of links (trill) use of is-is. RFC 7176, May 2014. (Cité en pages 26, 47 et 124.)

[51] Gandi. Gandi.net. `http://www.gandi.net/`. (Cité en page 27.)

[52] Donald Eastlake, Anoop Ghanwani, Vishwas Manral, and Caitlin Bestler. Rbridges : Further trill header extensions. Work in progress, draft-ietf-trill-rbridge-options-07, Juin 2012. (Cité en page 27.)

[53] Sam Aldrin, Donald Eastlake, Tissa Senevirathne, Ayan Banerjee, and Santiago Alvarez. Trill data center interconnect. Work in progress, draft-aldrin-trill-data-center-interconnect-00, March 2012. (Cité en page 34.)

[54] Tissa Senevirathne, Les Ginsberg, Sam Aldrin, and Ayan Banerjee. Default nickname based approach for multilevel trill. Work in progress, draft-tissa-trill-multilevel-02, Mars 2013. (Cité en pages 34 et 38.)

[55] Radia Perlman, Bhargav Bhikkaji, Balaji Venkat Venkataswami, Ramasubramani Mahadevan, Shivakumar Sundaram, and Narayana Perumal Swamy. Connecting disparate trill-based data center/pbb/campus sites using bgp. Work in progress, draft-balaji-l2vpn-trill-over-ip-multi-level-03, February 2013. (Cité en pages 35 et 36.)

[56] Balaji Venkat Venkataswami, Bhargav Bhikkaji, and Narayana Perumal Swamy. Interconnecting multiple trill sites deploying traffic engineering. Work in progress, draft-balaji-trill-te-multi-site-interconnect-00, Mars 2012. (Cité en page 36.)

[57] Ali Sajassi, Samer Salam, Aldrin Isaac, Nabil Bitar, and Sam Aldrin. Trill-evpn. Work in progress, draft-ietf-l2vpn-trill-evpn-01, Octobre 2013. (Cité en page 36.)

[58] Radia Perlman, Donald Eastlake, Anoop Ghanwani, and Hongjun Zhai. Flexible multilevel trill (transparent interconnection of lots of links). Work in progress, draft-perlman-trill-rbridge-multilevel-07, January 2014. (Cité en pages 39, 44 et 45.)

[59] Xiaolan Wan, Xiaopeng Yang, Vishwas Manral, and Alvaro Retana. Extending trill over wan. Work in progress, draft-xl-trill-over-wan-01, Juin 2012. (Cité en page 42.)

[60] Radia Perlman, Anil Rijhsinghani, Donald Eastlake, Ayan Banerjee, and Dinesh Dutt. Trill : Campus label and priority regions. Work in progress, draft-ietf-trill-rbridge-vlan-mapping-10, Janvier 2014. (Cité en page 43.)

[61] Hongjun Zhai, Tissa Senevirathne, Radia Perlman, Donald 3rd Eastlake, Mingui Zhang, and Yizhou Li. Rbridge : Pseudo-nickname for active-active access. Work in progress, draft-hu-trill-pseudonode-nickname-08, Juin 2014. (Cité en page 44.)

[62] Donald E. Eastlake 3rd, Radia Perlman, Anoop Ghanwani, Howard Yang, and Vishwas Manral. Transparent interconnection of lots of links (trill) : Adjacency. RFC 7177, May 2014. (Cité en page 47.)

[63] K. Ishiguro. Quagga software routing suite. `http://www.nongnu.org/quagga/`. (Cité en page 55.)

[64] Ktrill. `https://github.com/Gandi/ktrill/tree/devel`. (Cité en page 55.)

[65] Xen. `http://www.xenproject.org/`. (Cité en page 55.)

[66] Daniel Lemire and Leonid Boytsov. Decoding billions of integers per second through vectorization. *Software : Practice & Experience*, 2013. (Cité en page 84.)

[67] Gonzalo Navarro and Veli Mäkinen. Compressed full-text indexes. *ACM Comput. Surv.*, 39(1), April 2007. (Cité en page 84.)

[68] Mallik Mahalingam, T. Sridhar, Mike Bursell, Lawrence Kreeger, Chris Wright, Kenneth Duda, Puneet Agarwal, and Dinesh Dutt. Virtual extensible local area network (vxlan) : A framework for overlaying virtualized layer 2 networks over layer 3 networks. RFC 7348, October 2015. (Cité en pages 91 et 121.)

[69] Nvgre offload for windows server 2012 r2. Technical report, Chelsio Communications, Inc., 2014. (Cité en page 94.)

[70] Connectx®-3 pro : Solving the nvgre performance challenge. Technical report, Mellanox Technologies, 2013. (Cité en page 94.)

[71] Thomas Nadeau, Alia Atlas, Joel M. Halpern, Susan Hares, and David Ward. An Architecture for the Interface to the Routing System. RFC 7921, June 2016. (Cité en pages 110 et 129.)

[72] M. Bah, V. Del Piccolo, M. Bourguiba, and K. Haddadou. A centralized controller to improve fault tolerance in trill-based fabric networks. In *Smart Cloud Networks & Systems (SCNS), 2016 IEEE 3rd International Conference on*, 2016. (Cité en pages 110 et 129.)

[73] Tony Kontzer. Private, hybrid cloud interest spurred by security and control. http://www.networkcomputing.com/data-centers/private-hybrid-cloud-interest-spurred-by-security-and-control/d/d-id/1234367, Août 2013. (Cité en pages 111 et 112.)

[74] Tim Wilson. Study : Security fears continue to block cloud deployment. http://www.darkreading.com/informationweek-home/study-security-fears-continue-to-block-cloud-deployment/d/d-id/1174103?, Avril 2014. (Cité en page 111.)

[75] Rackspace. Rackspace 2013 hybrid cloud survey results. http://www.rackspace.com/knowledge_center/article/rackspace-2013-hybrid-cloud-survey-results, Août 2013. (Cité en page 112.)

[76] Gartner. Gartner says nearly half of large enterprises will have hybrid cloud deployments by the end of 2017. https://www.gartner.com/newsroom/id/2599315, Octobre 2013. (Cité en page 112.)

[77] Adel Nadjaran Toosi, Rodrigo N. Calheiros, and Rajkumar Buyya. Interconnected cloud computing environments : Challenges, taxonomy, and survey. *ACM Comput. Surv.*, 47(1) :7 :1–7 :47, May 2014. (Cité en pages 112 et 113.)

[78] Kvm. http://www.linuxkvm.org/page/MainPage. (Cité en page 118.)

[79] Mininet team. mininet : An instant virtual network on your laptop (or other pc). http://mininet.org/. (Cité en page 119.)

[80] Linux foundation. the opendayligth plateform ? opendaylight. https://www.opendaylight.org/. (Cité en page 119.)

[81] Yu-Shun Wang and Pankaj Garg. NVGRE : Network Virtualization Using Generic Routing Encapsulation. RFC 7637, September 2015. (Cité en page 121.)

[82] J. Hofmann. Internet corporation for assigned names and numbers (icann). *Global Information Society Watch*, 2007 :39–47, 2007. (Cité en page 128.)

[83] M. Bourguiba, K. Haddadou, I. E. Korbi, and G. Pujolle. Improving network i/o virtualization for cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 25(3) :673–681, March 2014. (Cité en page 129.)

[84] D. Cerovic, V. Del Piccolo, A. Amamou, and K. Haddadou. Offloading trill on a programmable card. In *Smart Cloud Networks & Systems (SCNS), 2016 IEEE 3rd International Conference on*, 2016. (Cité en page 129.)

Annexe A

Veille technologique sur les techniques d'isolation

A Survey of network isolation solutions for multi-tenant data centers

Valentin Del Piccolo*, Ahmed Amamou†, Kamel Haddadou†, and Guy Pujolle‡

Abstract

The Infrastructure-as-a-Service (IaaS) model is one of the fastest growing opportunities for cloud-based service providers. It provides an environment that reduces operating and capital expenses while increasing agility and reliability of critical information systems. In this multitenancy environment, cloud-based service providers are challenged with providing a secure isolation service combining different vertical segments, such as financial or public services, while nevertheless meeting industry standards and legal compliance requirements within their data centers. In order to achieve this, new solutions are being designed and proposed to provide traffic isolation for a large numbers of tenants and their resulting traffic volumes.

This paper highlights key challenges that cloud-based service providers might encounter while providing multi-tenant environments. It also succinctly describes some key solutions for providing simultaneous tenant and network isolation, as well as highlights their respective advantages and disadvantages. We begin with Generic Routing Encapsulation (GRE) introduced in 1994 in "RFC 1701", and will conclude with today's latest solutions. We detail fifteen of the newest architectures and then compare their complexities, the overhead they induce, their VM migration abilities, their resilience, their scalability, and their multi data center capacities. This paper is intended for, but not limited to, cloud-based service providers who want to deploy the most appropriate isolation solution for their needs, taking into consideration their existing network infrastructure. This survey provides details and comparisons of various proposals while also highlighting possible guidelines for future research on issues pertaining to the design of new network isolation architectures.

1 Introduction

Data centers are being increasingly used by both corporations and individuals. For example, in Cisco's forecast [1], personal content locker services, like Amazon Cloud Drive, Microsoft SkyDrive, and Google Drive, are expected to increase their total traffic by 57% in Cumulative Annual GRowth (CAGR) between 2013 (2 Exabytes) and 2018 (19 Exabytes). In addition to a growing use of data centers made by consumers (individuals), the use of data centers made by corporations will also increase.

As stated in [1], in 2013 global data center traffic reached 3.1 zettabytes for the year and is expected to grow to 8.6 zettabytes in 2018, representing a 3-fold increase. However, it is important to make a distinction between two data center types.

The first type of data center is the traditional one, which possesses specialized servers. On the contrary, the second type of data center is the cloud data center, which possesses non-specialized servers. These different data center types will not see the same increase in traffic. The traffic from traditional data centers will "only" increase by 8% CAGR between 2013 and 2018, while cloud data center traffic will see an increase of 32% CAGR during the same period , as predicted in [1]. In other words, in 2013 cloud data center workloads represented 54% of total data center workloads, and in 2018 it will represent 76% of the total data center workloads. We can therefore see a shift in favor of cloud data center.

This can be explained by one major advantage of a cloud data center over a traditional data center. A cloud data center is more prone to virtualization than a traditional data center. Indeed, cloud data centers are data centers with virtualized devices. With hardware improvement of data center nodes, it is possible to run several virtual machines (VMs) on one physical node.

Using several VMs on a physical node allows using it to the fullest of its capabilities. It therefore spends less time in an idle state and wastes less energy. Consequently, it is therefore cost-effective for both the infrastructure provider and their customers. The infrastructure provider, owner of the data center, has nodes actively processing the data of its clients instead of being held in an idle state. This increases the load time of the nodes, which in turn increases their cost-effectiveness. The infrastructure provider therefore needs fewer physical devices for a fixed number of clients. Instead of having multiple physical devices for one client, the provider has multiple VMs for this client. All these VMs can be on one physical device (Figure 1).

Virtualization also adds functionalities such as :

- Remote OS install
- Access to server console
- Reboot of frozen server.

*Valentin Del Piccolo is with the Research and Development Department of GANDI SAS, Paris, France, and a Phd student at the University Pierre et Marie Curie (UPMC), Paris, France. e-mail: valentin.d.p@gandi.net

†Dr Ahmed Amamou and Dr Kamel Haddadou are with the Research and Development Department of GANDI SAS, Paris, France, e-mail: ahmed@gandi.net, kamel@gandi.net

‡Pr Guy Pujolle is a Professor at the University Pierre et Marie Curie, Paris, France. e-mail: Guy.Pujolle@lip6.fr

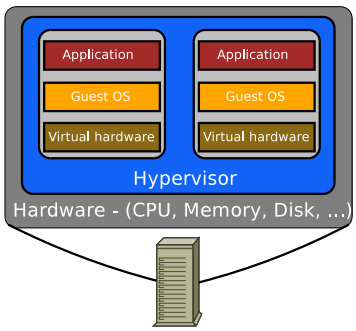


Figure 1: Two VMs on one node

- Guest OS choice.
- Possibility of server snapshots for backups.
- Hardware upgrade without shutting down the VM.
- Possibility of VM migration on a newer server with the backup image of the VM.

However, sharing a physical node among several clients implies that there is no device or data isolation. Nevertheless, clients do not want their data exposed to other clients, who might even be competitors. In order to solve this problem, it is necessary to deploy techniques that will provide client isolation. This results in clients only seeing VMs and traffic that they own, and make them believe that they are alone on the network.

The remainder of the survey is organized as follows. After quickly reviewing terminology and definitions that pertain to multitenancy isolation in the cloud (Section 2) and explaining tunneling and virtual network notions (Section 3), we detail in Section 4 some network isolation solutions developed before the cloud era in order to show why new solutions are needed for cloud data center with multi-tenant issues. In Section 5 we present those new solutions which provide multi-tenant isolation in cloud data centers. Then we focus on fifteen solutions that provide tenants traffic isolation as follows: The Locator/Identifier Separation Protocol (LISP) [2], Network Virtualization using Generic Routing Encapsulation (NVGRE) [3], Stateless Transport Tunneling Protocol (STT) [4], 802.1ad or QinQ [5], 802.1ah or mac-in-mac [6], Virtual eXtensible Local Area Network (VXLAN) [7] Diverter [8], Portland [9], Secure Elastic Cloud Computing (SEC2) [10], BlueShield [11], VSITE [12], NetLord [13], Virtual Network over TRILL (VNT) [14], VL2 [15], Distributed Overlay Virtual nEtwork (DOVE) [16, 17]. We compare them using six criteria in Section 7. We then discuss the future of tenant isolation (Section 8) and, finally, present our conclusions (Section 9).

2 Terminology

In this section we define both terms Tenant and Multitenancy. To do so, we use the definitions given in [18, 19, 20, 21].

2.1 Tenant

In Cisco Virtual Multi-Tenant Data Center 2.0 [19] a tenant has two definitions. In the private cloud model a tenant is defined as "a department or business unit, such as engineering or human resources". In the public cloud model a tenant is "an individual consumer, an organization within an enterprise, or an enterprise subscribing to the public cloud services". In version 2.2 of Cisco Virtual Multi-Tenant Data Center [20] the difference between public or private cloud has been removed and a tenant is "an user community with some level of shared affinity". To explain this definition, examples are provided in which a tenant may be a business unit, department, or work group.

Juniper gives a different definition, they state in their white paper [21] that "a cloud service tenant share a resource with a community". In order to express this definition more clearly, the example of building tenants is given. In this metaphor, a building tenant has to share the building's infrastructure just like a cloud service tenant. However a tenant can also have tenants such as stated in [21]. The given example is the case of Second Life which is a tenant of Amazon Web Services and which has tenants of its own, who could also have tenants and so on. Wider than Cisco's definition of a tenant, Juniper defines a tenant as a cloud service user. This user can be a person or a company or, as in Cisco's definition, a business unit from a company.

In this paper we use the term tenant as defined by Juniper.

2.2 Multitenancy

For Cisco [20], "virtualized multi-tenancy" is a key concept which refers to "the logical isolation of shared virtual compute, storage, and network resources".

In continuation with the building metaphor, most of the time there is not only one tenant in a building. Therefore the building is a multitenancy environment. Each tenant wants privacy so they are isolated in apartments. This metaphor is well presented in "The Force.com Multitenant Architecture" [22] and is reproduced below :

> "Multitenancy is the fundamental technology that clouds use to share IT resources cost-efficiently and securely. Just like in an apartment building - in which many tenants cost-efficiently share the common infrastructure of the building but have walls and doors that give them privacy from other tenants - a cloud uses multitenancy technology to share IT resources securely among multiple applications and tenants (businesses, organizations, etc.) that use the cloud."

For Juniper [21], multitenancy is the idea of many tenants sharing resources. It is also a key element for cloud computing. However multitenancy also depends on the service provided. For example, in an IaaS environment, the provider provides infrastructure resources

like hardware and data storage to the tenants who in turn must share them. In a SaaS environment, tenants use the same applications, so there is a chance that their data is stored in a single database by the service provider. There are security constraints to apply at each layer.

In this survey we focus on data center architectures providing tenants' traffic isolation.

3 Background

This section explains the notions of virtual networks (Section 3.1) and tunneling (Section 3.2). We also detail the relation between multitenancy, virtual networks, and tunneling in Section 3.3.

3.1 Virtual network

Microsoft defines, in [23], a virtual network as a configurable network overlay. The devices from one virtual network can reach each other but those outside of it can not, thus providing isolation to the devices inside the virtual network.

A more concise definition of a virtual network is given in [24]: "[...] a virtual network is a subset of the underlying physical network resources."

Using both definitions we see that a virtual network is a configurable overlay network that uses resources, virtual nodes, and virtual links of a physical infrastructure while at the same time keeping them isolated. A virtual node is a logical node using at most all the resources of a physical node. A virtual link works the same way as a virtual node but using resources of a physical link. This being said, a virtual network does not use all the resources of a physical infrastructure, and so it is possible to have several virtual networks over a physical infrastructure. In order to achieve this, it must allocate resources from physical nodes and physical links to each virtual network. Information about resource allocation algorithms and technology can be found at [24] and [25].

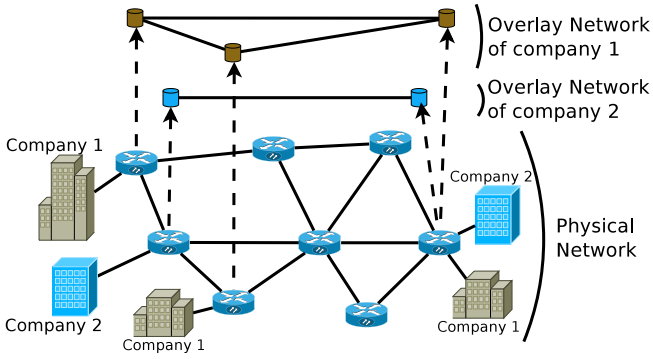


Figure 2: Example of overlay networks

Figure 2 shows an example of a physical infrastructure hosting two overlay networks. We can see that one physical node belongs to both overlay networks as it hosts one virtual node from each overlay network and that some links must transit data from both overlay networks. Each overlay network must be isolated when sharing the same infrastructure. Tunneling is therefore mandatory to keep data from exiting a virtual network.

3.2 Tunneling

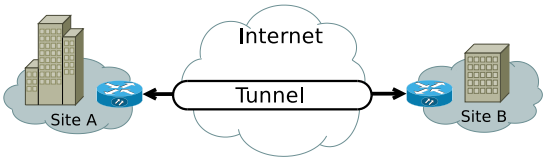


Figure 3: Tunnel

Figure 3 shows the concept of tunneling the data through a tunnel from one point to another. Data using this tunnel is isolated from the rest of the network.

In [26] Cisco defines Tunneling as "a technique that enables remote access users to connect to a variety of network resources (Corporate Home Gateways or an Internet Service Provider) through a public data network." This definition is represented in Figure 3 as we have two sites (A and B) interconnected through the Internet.

In [27], "A tunneling protocol is one that encloses in its datagram another complete data packet that uses a different communications protocol. They essentially create a tunnel between two points on a network that can securely transmit any kind of data between them." Additionally, in [28], "Tunneling enables the encapsulation of a packet from one type of protocol within the datagram of a different protocol." These two definitions add the notion of a packet being encapsulated in another packet, thus the tunneling protocol creates a new packet from the original packet. For example GRE adds a new header (Figure 5) to the packet.

3.3 Multitenancy via virtual networks and tunneling

As stated in Section 3.1 a virtual network only uses a portion of the physical infrastructure's resources. This implies that the rest of the resources can be used for other virtual networks in order to maximize infrastructure usage. Doing so means that the physical resources are shared among virtual networks, however a virtual network belongs to a tenant, and thus the infrastructure provides resources for multiple tenants. This is what we call multitenancy (Section 2).

To grasp the notion of isolation in a data center, it is necessary to understand that the goal of the data center operator is to maximize the use of its infrastructure. To achieve this, it uses virtual networks and tunneling in order to accommodate the maximum number of tenants possible on its infrastructure. Several hundreds or thousands tenants can share the same infrastructure

inside a data center, thus the main challenge when providing isolation in this environment is to be able to provide it for a very large number of tenants. Each tenant wants to have its network isolated from other tenants, therefore scalability is a concern. Another challenge is to provide an isolation solution that can sustain misbehavior or misconfiguration inside tenants' networks without impacting other tenants. Therefore the solution must be resilient. Additionally, isolation inside a data center must be assured inside the whole data center therefore all the devices composing the infrastructure must manage the chosen isolation solution. A fourth challenge is to maintain availability of the data center even when updating the infrastructure by adding new devices or tenants. Moreover, each tenant has its own rules and policies, thus the isolation solution must enforce those rules only for the right tenant.

To summarize, the main issue with multitenancy in a data center is caused by the huge numbers of tenants, policies, servers and links. It is mostly a scalability issue. However, the isolation solution must cope with this issue without degrading, too much, the performances of the infrastructure. Therefore, another challenge for multitenancy is to have an isolation solution with a low overhead.

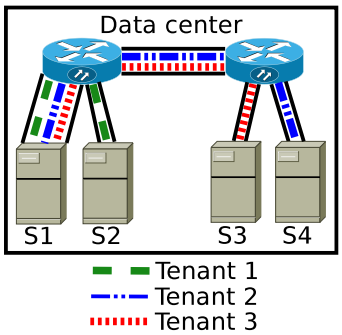


Figure 4: Multitenancy

An example of a multi-tenant data center is shown in Figure 4. In this example, either the three tenants have data stored on Server 1 (S1) and each tenant possesses a virtual network therefore their traffics are tunneled from end to end. The various flow representations indicate all path long isolation for each tenant's traffic while using a common infrastructure.

In order to enforce multitenancy we need both isolation in the network (Section 3.3.3), via the tunneling protocol, and isolation in the nodes, via Hypervisor level Isolation (Section 3.3.1) or Database level Isolation (Section 3.3.2). However, by achieving isolation there is some performance degradation (Section 3.4.1) as well as risks (Section 3.4.2) when the isolation solution is violated.

3.3.1 Hypervisor level isolation

A hypervisor is a software mapping of the physical machine commands to a virtualized machine (VM) running a regular OS. The hypervisor intercepts the OS system calls of the VM and maps these calls to the underlying hardware. This implies that the hypervisor induces a certain percentage of overhead. However, a hypervisor allows the partition of the hardware, thus having several tenants on the same physical node. Since the hypervisor is between the VM and the node, it intercepts all the traffic, thus it can isolate each virtual machine and their resources. With this isolation, a hypervisor allows the protection of tenant resources thereby enabling multitenancy.

3.3.2 Database level isolation

While hypervisor-level isolation is used in Infrastructure-as-a-Service (IaaS), database-level isolation is used in Software-as-a-Service (SaaS). In SaaS, tenants share a database. In [29] the authors describe the three main approaches for managing multi-tenant data in database. The first approach is to have separate databases for each clients. The second one is to have a shared database but with separate schemas, therefore multiple tenants use the same database but each tenant possesses its own set of tables. The last approach is to share both the database and the schemas. In this case an id is append to each record in order to indicate which tenant is the owner of the record. In [30] a new schema-mapping technique for multi-tenancy called "Chunk Folding" is proposed in order to improve the performance of database sharing among multiple tenants. Additionally, in [31], the authors propose a solution called SQLVM which focuses on allocating resources for tenant dynamically while ensuring low overheads. Other solutions like [32, 33, 34] focuses on improving database performances when the number of user increases.

This type of isolation has more security concerns than hypervisor-level isolation. If the modification of the request is mis-configured or if there is an error in an access control list then tenants' information is at risk.

3.3.3 Network level isolation

Tunneling is a key element for isolation inside a data center, however data centers have different constraints than typical LAN networks. The most important one is the number of different tenants whose isolation must be provided. Therefore the scalability of the tunneling protocol and the maximum number of tenants it can manage, is a criterion to take into account when choosing a tunneling protocol. If the tunneling protocol can not isolate all the tenants of a data center then there is no interest in using it, therefore we performed a scalability comparison in Section 7.5. Another criterion is the overhead induced by the tunneling protocol. The challenge is to have the lowest overhead possible. In Section 7.2 we compare the overhead induced by the tunneling protocol. A third criterion, which influences overhead, is the security provided by the tunneling protocol (Section 7.1.4). Then the resilience criterion is

also important in order to quickly mitigate any link or node failure (Section 7.4). As we focus on data center networks we also choose as criterion, the ease of multi data center interconnection which we describe in Section 7.6.

Another element of choice when deciding which tunneling protocol to choose is how it enforces its tunnel. There are two possibilities as indicated in [35]. The first one is the Host isolation technique described at the beginning of Section 4.1. In this technique, the ingress and egress nodes are the ones enforcing the isolation. The second technique, called the Core isolation technique (Section 4.2), enforces network isolation at each switch on the path.

However, while these criteria must be taken into account, the goal is to have the greatest scalability possible with the lowest overhead and complexity.

3.4 Impact of isolation

Providing multitenancy is a key function for cloud data center. However, this functionality implies overhead (Section 3.4.1) and also risks (Section 3.4.2) in case the tenants isolation fails.

3.4.1 Isolation performances overhead

Hypervisor performances have already been studied in several papers [36, 37, 38, 39]. They induce overhead thus performance is decreased. However, not all hypervisors have the same impact on performance. In [36] four hypervisors (Hyper-V, KVM, vSphere, Xen) are compared over four criteria (CPU, Memory, Disk, Network). Their results show that hypervisor overhead is globally low, therefore they do not deteriorate performances too much.

Network isolation solutions are the main subject of this paper, thus in Section 7 a comparison is done between several of them.

First we study their complexity based on six criteria. The first is the control plane design of the solution. The second is network restrictions imposed by each solution, some of them only work with Layer 3 (L3) networks, other only with Layer 2 (L2) networks, and some of them need specific architecture. The third criterion focuses on tunnel configuration and establishment. We analyze if there are messages needed to establish a tunnel, or if it must be allocated before hand on each node. The fourth criterion is tunnel management and maintenance in order to determine the quantity of messages needed by each protocol to keep alive those tunnels. The fifth criterion is the capacity of those tunnels to handle multiple protocols. The sixth, and last, criterion of this complexity study focuses on their security mechanisms.

Then we study the overhead induced by each solution, followed by their capability to migrate VM and a comparison of their resilience. The last criterion for their comparison is their scalability and their capacity to be managed among multiple data centers.

3.4.2 Isolation violation risks

Multitenancy is based on sharing the underlying physical infrastructure, thus tenants' data is stored on the same devices while being isolated via tunneling protocols and hypervisor- or database-level isolation. However, if one of these isolation mechanisms fails, then the data can be seen by other tenants. Having the hypervisor- or the database-level isolation fail implies that all the tenants sharing the same hypervisor or database can access all the data managed by the hypervisor or that is inside the database. This is an issue, however it is restricted to one node and can be resolved without shutting down the whole data center. The worst case is when the tunneling protocol fails. In this case all tenants' traffic is visible, thus data can be stolen or misused by other tenants. To resolve this situation, the data center must be stop in order to reconfigure or change the tunneling protocol, thus the choice of a tunneling protocol is an important decision.

4 Network isolation in traditional data center

The network solutions introduce in this Section were designed before the development of cloud data center. They possess capabilities for isolating flows in a network but are either not scalable enough, or were not designed for cloud data center topologies. As such they can not cope with the increasing number of flows and Virtual Machines (VMs) to isolate. Additionally they can not manage VM live migration, which is not necessary for traditional data centers in which there is no VM, but is mandatory for cloud data centers. Therefore, we present those solutions because they can be used in some traditional data center and mostly to show that there is a need for new multi-tenant network isolation solutions.

4.1 Host isolation

Host isolation is an isolation method which selects flows once they arrive at the destination host or the egress node of the tunnel. This means that all along the path no switching or routing device checked the packets or messages of the flow. The switching or routing is done normally using the information of the transport header. It also means that there is no explicit need for a tunneling protocol in this kind of isolation. For example the Ethernet protocol is a Host isolation protocol. When a packet reaches a host, this host checks the MAC address and accept or not the packet if the MAC address of the packet matches the MAC address of the host. It is only once the packet arrives at the destination host or the egress node of the tunnel that checks are done. Either the destination host or the egress node verifies if both the destination Virtual Machine (VM) and the flow, to which the data belongs, are from the same virtual network. If they do belong to the same virtual network then the data is either delivered to the

VM or dropped depending on policies. This Host isolation advantage is that it does not require the node to know the whole topology. Indeed it is not necessary to know the location of the others VMs in order to distribute the flows. However this techniques has drawbacks. The lack of information about the VMs belonging to the same network imposes that the flows of each VM be propagated in the whole data center. This creates useless traffic, overloading the data center, which is dropped when received by a physical node not belonging to the same network. Additionally, such isolation technique security is weak against a "man in the middle" attack. An attacker who is able to put a listening device in the network could see the traffic from all the clients.

4.1.1 Host isolation for both Layer 2 and Layer 3 networks

The protocol presented in this section can be used over both Layer 2 and Layer 3 network.

4.1.1.1 GRE: Generic Routing Encapsulation

The Generic Routing Encapsulation protocol was proposed in "RFC 1701" [40]. In this RFC, the goal of GRE is to encapsulate any other protocol with a simple and lightweight mechanism. To do that GRE adds a header to the packet (Figure 5). The original packet is called the payload packet and the GRE header is added to it. Then, if the packet needs to be forwarded, the delivery protocol header is also added to the packet. The GRE header is composed of nine mandatory fields, for a total of 4 bytes, and of five optional fields with a variable total length. In these optional fields, there is a routing field which contains a list of Source Route Entries (SRE). Thanks to this field, it is possible to use the GRE header to forward the packet without adding a delivery header.

In the second RFC, "RFC 2784" [41] derived from the original "RFC 1701" without superseding it, the GRE header has been simplified. The header is now made of 4 mandatory fields (4 bytes) and of 2 optional fields (4 bytes). The header length is now limited to 8 bytes whereas in the first RFC there was no length limit. The new header needs the delivery header because there is no information to forward or route the packet in it anymore. As it is a lightweight mechanism, some functionalities are not managed such as the discovery of the MTU along the path. This could be an issue if the source sends a packet with the "don't fragment bit" set in the delivery header and the packet is too big for the MTU. In this case the packet is dropped along the path. As the error message is not required to be sent back to the source, the source could keep sending packets too big for the MTU. Those packets would always be dropped and would never reach their destination.

GRE allows for an easy deployment of IP VPN and can tunnel almost any protocol through those VPN. Additionally it is possible to authenticate the encapsulator by checking the Key field. However, the provisioning is not scalable. In addition, GRE does not protect the payload of its packets because of a lack of integrity check and encryption. In order to resolve this last issue, it is possible to use GRE over IPSec.

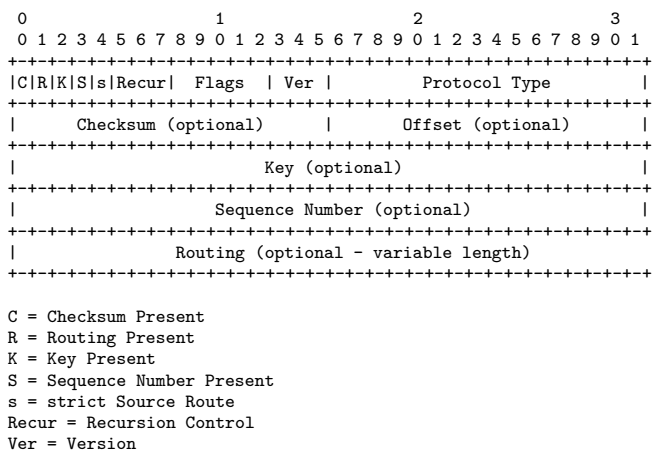


Figure 5: GRE Header

4.1.2 Host isolation for Layer 3 networks

In this section we introduce four Host isolation protocols which impose that the underneath network be a Layer 3 network.

4.1.2.1 PPTP: Point-to-Point tunneling protocol

The Point-to-Point Tunneling Protocol (PPTP), from "RFC 2637" [42], was introduced 5 years after the first version of GRE [40]. This can explain why GRE is used to do the tunneling in PPTP. However, the GRE header is modified in PPTP (Figure 6). The Routing field is replaced by an acknowledgment number field of 4 bytes. This way, the header has a maximal length. The new acknowledgment number field is used to regulate the traffic of a session. As the PPTP tunnel multiplexes sessions from different clients, this acknowledgement number allows traffic policing for each session. The tunnel formed by PPTP between the PPTP Access Concentrator (PAC) and the PPTP Network Server (PNS) is deployed over IP, which is why the routing field was not necessary (as the routing is done by IP). The other difference with GRE is the use of the key field. In PPTP, the key field is divided in two parts: the higher two bytes, which are used for the payload length, and the lower two bytes which are for the call Id. The call Id represents the owner of the packet.

One of the advantages of PPTP is that it only needs two devices to be deployed at each end of the tunnel. A PNS at one end and a PAC at the other. There is no need to know or interact with the network between both ends. For data confidentiality, PPTP uses an encryption technique called Microsoft Point-to-Point Encryption (MPPE). MPPE uses the RSA RC4 encryption algorithm and a session key to encrypt the packet. Three key lengths are supported: 40 bits, 56 bits, and 128 bits. Another advantage of PPTP is that there

is no need for an agreement with the service provider. The administrator in charge of the PPTP session has complete control over it. However the administrator must be able to install, configure, and manage a PPTP device at both ends of the tunnel.

The disadvantage of PPTP is that it uses TCP for all its signaling so it does not support multipath, and always uses the same path for a session. PPTP is end-user initiated because of its design. Only both ends of the tunnel know about the tunnel, so the service provider is not aware of PPTP. Because of that there is no Quality of Service (QoS) possible.

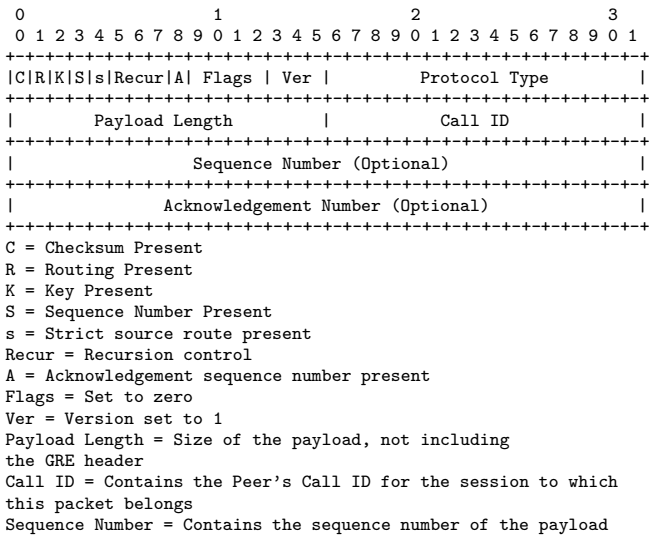


Figure 6: PPTP Header

4.1.2.2 L2TP: Layer Two Tunneling Protocol

L2TP was developed by the IETF and proposed as a standard in "RFC 2661" [43]. L2TP is designed to transport PPP frames (Layer 2) over a packet-switched network (Layer 3). With L2TP it is possible to have two PPP endpoints residing on different networks interconnected by a Layer 3 network. It allows extending the Layer 2 network to other Layer 2 networks interconnected through a Layer 3 network. To design L2TP, the IETF used the Layer-2 Forwarding (L2F) and the Point-to-Point Tunneling Protocol (PPTP) as a starting point. L2F is a Cisco proprietary tunneling protocol which provides a tunneling service for PPP frames. PPTP was developed by Microsoft and is also designed to transport PPP frames over Layer 3 networks. L2TP works with two devices, the L2TP Access Concentrator (LAC) and the L2TP Network Server (LNS). Those are the endpoints of the L2TP tunnel. The LAC is located at the ISP's Point of Presence (POP). The LAC exchanges PPP messages with users, and communicates with customers' LNS to establish tunnels. To use L2TP, the ISP needs to be informed because they must have a L2TP-capable POP. This POP must encapsulate PPP frames within L2TP ones, and forward them through the correct tunnel toward the LNS, which belongs to the customer. The LNS must accept L2TP frames, and strip the L2TP encapsulation in order to deliver the PPP frames to the appropriate interface. L2TP possesses integrated security techniques such as an authentication between the client and the LAC at the initiation of the tunnel, a tunnel authentication between the LNS and the LAC, and a client authentication and authorization between the client and the LNS. This last authentication can use the Password Authentication Protocol (PAP), the Challenge Handshake Authentication Protocol (CHAP), or a one time password. After that, the PPP session begins and the data can be exchanged.

With L2TP, the ISP is needed which results in an extra cost in order to establish the tunnel. Nevertheless with the ISP's involvement, it is possible to add Quality of Service guarantees (QoS) and to benefit from the ISP IP network reliability. The fact that the L2TP encapsulation is done by the LAC means that there is no need for client software. This is advantageous because it removes the difficulties associated with managing remote devices. However there is no multipath management because of the design of L2TP, in which a client is in one tunnel and the tunnel has only one path. But load balancing and redundancy are possible thanks to multiple home gateways.

4.1.2.3 L2TPv3: Layer Two Tunneling Protocol - Version 3

In L2TPv3, "RFC 3931" [44], the tunnel can be established between L2TP Control Connection Endpoint (LCCE) which are the LAC and the LNS. The novelty is that it is possible to have a LAC-to-LAC tunnel or a LNS-to-LNS tunnel. In addition a device can be a LAC for some sessions and a LNS for others. Another modification is the use of two headers, the control message header (Figure 7) and the data message header (Figure 8), instead of one header for all messages.

The control message header has the same length as the original one but with one less field. The Session ID field is now 4 bytes long, instead of 2 bytes, and the Tunnel ID field is removed. L2TPv3 replaces the data header with a L2TPv3 Session Header. The RFC states that, "The L2TP Session Header is specific to the encapsulating Packet-Switched Network (PSN) over which the L2TP traffic is delivered. The Session Header MUST provide (1) a method of distinguishing traffic among multiple L2TP data sessions and (2) a method of distinguishing data messages from control messages."

The LCCE from L2TPv3 does not need to be at a Point Of Presence (POP) of an ISP. Consequently, it is possible to establish a tunnel without the ISP's help, thus reducing the cost. However, without the ISP there is no service guarantee on the Layer 3 network.

4.1.2.4 PWE3: Pseudo Wire Emulation Edge-to-Edge

Pseudo Wire Emulation Edge-to-Edge (PWE3) is a technology that emulates services from Layer 2 such as Frame Relay, ATM, Ethernet over packet switched networks (PSN) using IP or MPLS. It was proposed in

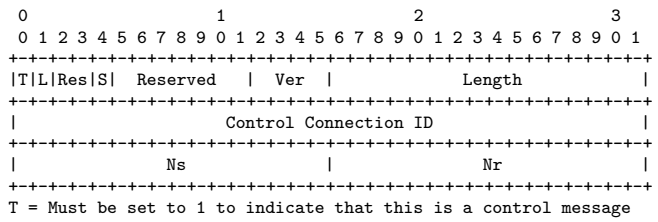


Figure 7: L2TPv3 control message header

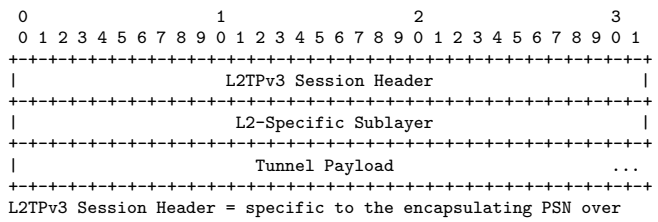


Figure 8: L2TPv3 data message header over IP

"RFC 3985" [45]. PWE3 defines an encapsulation function which encapsulates service-specific bit streams, cells, or PDUs. This service-specific data is encapsulated at the ingress node in order to be sent over the PSN. The encapsulation is done by the Provider Edge (PE), then the data is carried across a PSN tunnel. PWE3 is an encapsulation protocol which emulates a Layer 2 service over a PSN network. However, to tunnel the data, it needs a tunneling protocol such as L2TP or MPLS. This protocol uses a Host isolation method if it is used with L2TP and a Core isolation method if it is used with MPLS.

4.2 Core isolation

The Core isolation method requires that each node of the network possesses a wider knowledge of the topology than when using the Host isolation method. As a matter of fact, in the Core isolation method each node on the path (switch or router) has to check the packet in order to verify if it can be forwarded toward its destination. The benefit of such an isolation method is that the packet is dropped at the closest node from the source if the destination is not reachable for policy reasons. This method considerably reduces traffic, by preventing the transmission of useless traffic to nodes which are not concerned by such traffic. However it is necessary to have a global view of the network topology in order to transmit packets. This implies either a pre-configured network with strict rules or an auto-configurable network. The first case means that the topology is rigid which is contrary to virtualization principles such as live migrations. In the second case, it would increase the waiting time before being able to make two entities of the network communicate. This increase happens due to the time needed for exploring and sharing the network's information.

4.2.1 Core isolation for Layer 2 networks

Both Core isolation protocols introduce in this section require the underneath network to be a Layer 2 network.

4.2.1.1 VLAN: Virtual LAN

A VLAN emulates an ordinary LAN over different networks as defined in the "802.1Q IEEE standard" [46]. The nodes belonging to a VLAN are members of this VLAN. A member of a VLAN communicates with the other members of the VLAN as if they were on the same LAN despite their geographical location. VLAN members are in a logically separated LAN and share a single broadcast domain. They do not know that there are not on the same physical LAN. The other nodes, not member of the VLAN, will not see the traffic from the VLAN and will not receive any of the broadcast messages from the VLAN. All the traffic from a VLAN is isolated from the rest of the network.

There are three methods to recognize the members of a VLAN. The first method is port based. The switch knows that the node connected at the specified port is a VLAN member. The specified port is tagged and is now processing VLAN-only messages. The second method is based on the recognition of the MAC address. And the third method is based on the recognition of the IP address. Independent of the method, the packets of the VLAN are tagged with a 4-byte header (Figure 9) between switches and routers. This field contains the VLAN ID (VID) field, which is 12 bits long. The VID is used to know which VLAN the message belongs to, since switches and routers can multiplex VLANs on a link. Such link is called a VLAN trunk.

If a VLAN member has moved and the VLAN is configured to use MAC addresses, the VLAN can recognize that the member has moved. The VLAN can then automatically reconfigure itself without the need to change the member's IP address.

Among VLAN advantages we have that VLANs facilitate administration of logical groups of stations. They allow stations to communicate as if they were on the same LAN. The traffic of a VLAN is only sent to members of the VLAN which allows flow separation. A VLAN diminishes the size of a broadcast domain and so improves bandwidth. There is also a security improvement thanks to a logical isolation of the VLAN members. An ISP agreement is not needed to establish a VLAN.

A disadvantage of VLANs is that there are only 4096 VLANs because of the size of the tag. To solve this, the IEEE 802.1ad standard [5], presented in Section 5.2.1.1, has been developed and it increases the number of VLANs. Another issue in the original definition of the 802.1Q is the lack of a control plane which enable

automatically provisioning the path on each switch. This last issue is fixed by the use of the GARP VLAN Registration Protocol (GVRP) [47], which is a Generic Attribute Registration Protocol application.

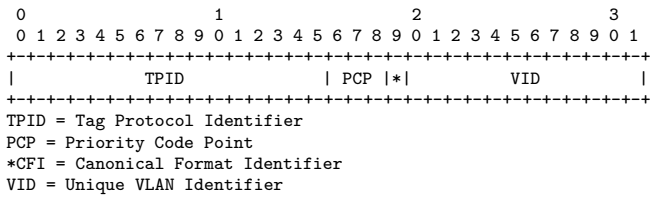


Figure 9: VLAN Header

4.2.1.2 802.1ad - Provider Bridges

The first draft [48] of the 802.1ad standard was released in 2002 and was intended to enable a service provider to offer separate LAN segments to its users over its own network. Therefore, both the user and the provider possess their own VLAN field which increase the number of available VLANs. Even if this solution was proposed before the growth of cloud data center, it has been adapted to fit to cloud data center networks, therefore we present this solution in Section 5.2.1.1.

4.2.2 Core isolation protocols for Layer 3 networks

In this section we present three Core isolation protocols which can be used only if the underneath network is a Layer 3 network.

4.2.2.1 MPLS: Multiprotocol Label Switching

Multiprotocol Label Switching (MPLS), defined in "RFC 3031" [49], is a circuit technique that uses label stacks on packets in order to forward them. MPLS uses Layer 3 (IP) routing technique with Layer 2 forwarding in order to increase the performance/price ratio of routing devices and to be open to new routing services invisible at the label forwarding level.

MPLS decreases the processing time of each packet, with only a label of 20 bits (Figure 10) to look at to forward the packet. It has the ability to work over any Layer 2 technology such as ATM, Frame Relay, Ethernet, or PPP. MPLS has traffic engineering techniques with the Resource reSerVation Protocol (RSVP) [50] or Constraint-based Routing Label Distribution Protocol (CR-LDP) [51] and enables Quality of Service (QoS) using DiffServ [52].

MPLS packets are named "labeled packets" and the routers which support MPLS are called "Label Switching Routers" (LSR). The packets are labeled, at the ingress LSR, depending on the forwarding equivalence class (FEC) they belong to. Those labels are locally used, each LSR changes it depending on the label the next LSR in the path as announced for the FEC. For each FEC exists at least one path across the MPLS network. This path is a Label Switched Path (LSP). All the packets of one FEC takes the same LSP. On this LSP, the labeled packets are forwarded based on their labels. After all the label changes, the egress LSR removes the label.

MPLS creates a tunnel for the traffic from the rules it uses. It is also possible to manually edit the labels to define a LSP through the MPLS network. The traffic is tunneled all along the path thanks to the configuration and rules established in the control plane.

In order to use MPLS, the network must be MPLS-ready and configured with a FEC for the traffic. ISP intervention is needed to have a FEC configured, meaning extra cost for the client. However customers could have QoS for their traffic.

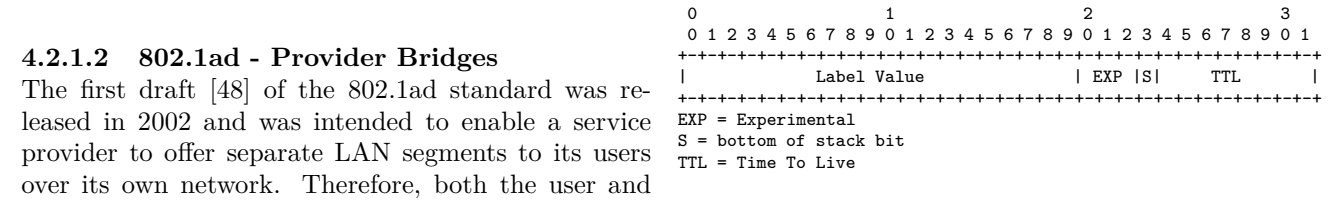


Figure 10: MPLS header

4.2.2.2 GMPLS: Generalized Multi-Protocol Label Switching

GMPLS was proposed in "RFC 3471" [53] and updated in "RFC 3945" [54] as an extension of MPLS. This extension adds support for new switching types such as Time-Division Multiplexing (TDM), lambda, and fiber port switching. To support those new switching types, GMPLS has new functionalities which modify the exchange of labels and the Label switched Path (LSP) unidirectional characteristic. MPLS forwards data based on a label, but the new switching techniques are not based on header processing, so GMPLS must define five new interfaces on the Label Switching Routers (LSR).

1. The Packet Switch Capable (PSC) interface, like the one from MPLS, uses the header of the packet for routing.

2. The Layer-2 Switch Capable (L2SC) interface uses the frame header, like the MAC header or the ATM header, to forward the frame.

3. The Time-Division Multiplex Capable (TDM) interface switches data thanks to the data's time slot in a repeating cycle.

4. The Lambda Switch Capable (LSC) interface receives data and switches it via its wavelength when it was received.

5. The Fiber-Switch Capable (FSC) interface switches the data based on its position in physical space.

For the LSC interface, the header is 32 bits long and contains only a Label field (Figure 11). The other interfaces use the same header which contains a Label field with a variable length (Figure 12). However, to establish a circuit, two interfaces of the same type are

needed at each end. In GMPLS it is possible to establish a hierarchy of LSPs on the same interface or between different interfaces. If it is on the same interface, it means that the LSR was able to multiplex the LSPs. If it occurs between interfaces then that means that the LSP start with one type of interface and another one is used along the path. If such an interface change happens on the path, then the original LSP is nested into another LSP. This new LSP must end before the original LSP in order to have the same interface type for the final one as the first one. For example, the LSP starts and ends on a PSC interface and along the way the interface changes into FSC so the PSC LSP is nested into a FSC LSP. As MPLS, GMPLS uses the control plane to establish rules, labels, to route all the data of an LSP which are tunneled through a unique path. GMPLS extends LSRs capabilities from MPLS by allowing different techniques for data forwarding. However GMPLS shares the same constraint as MPLS, in that there must be an agreement with the ISP before using it.

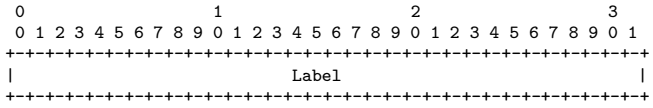


Figure 11: GMPLS header for Lambda interface

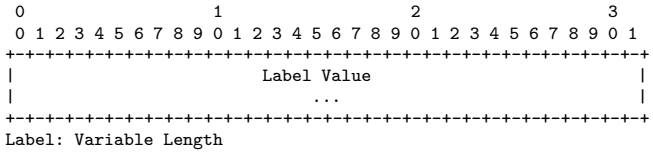


Figure 12: GMPLS header for PSC, L2SC, TDM and FSC interfaces

4.2.2.3 BGP/MPLS IP Virtual Private Networks

In Border Gateway Protocol (BGP) / MultiProtocol Label Switching (MPLS), described in "RFC 4364" [55], the idea is to use the MPLS label to forward the packet in the network and BGP to exchange the route information between the LSRs. As shown in Figure 13, clients need to install at least one Customer Edge (CE) router at each site they want to connect. The CE has to know every private IP address from the site it is in. The CEs are then connected to Provider Edges (PE) provided by the ISP. Each PE learns all the IP addresses accessible through the CE it is connected to and then uses BGP to exchange the addresses with the other PEs over the ISP's core network. The PE creates one or more Virtual Routing and Forwarding (VRF) table(s) containing the information of the path to each PE and each device in his local network. The core network router, which is working with MPLS, does not know any of the clients addresses.

A Virtual Private Networks (VPN) contains at least two PEs to connect two sites. In order to create such a network, the client and the ISP must have an agreement. However a BGP/MPLS VPN system allows the overlapping of address spaces between VPNs, so clients could use any addresses they want in their VPN. BGP/MPLS IP VPNs grants privacy if the network is well configured. But there is no encryption, no authentication, and no integrity check method. In order to add security measures IPsec must be used.

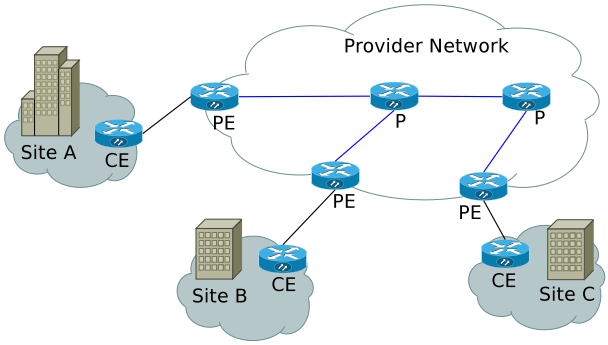


Figure 13: BGP/MPLS network

5 Network isolation in cloud data center with multi-tenant capabilities

The solutions and protocols shown in Section 4 are not appropriate for isolation in Cloud Data Centers (Cloud DC) prone to virtualization. New data centers use virtualization technologies in order to increase the number of tenants sharing the same infrastructures and the limits of those isolation techniques are not sufficient for accommodating all clients.

For example, GRE 4.1.1.1 provisioning is not scalable. We must know beforehand how many clients there will be in the data center, which is not possible in virtualized data centers. PPTP is end user-initiated because of its design, thus preventing the administrator of the data center from using it freely.

The VLAN limit of 4096 different identifiers is very small in comparison to the number of clients in a virtualized data center. Already in 2010, VMware users were running an average of 12.5 virtual machines per physical server [56]. However those users were not necessarily professional data center, and consequently may not have purchased the best server. In [57] the number of VMs per server is given as a ratio of 25 VMs per 1 server, and is expected to grow to 35 VMs per server. However, VMware is currently advertising that some of its servers can host 29 VMs [58]. We need to be careful with these statements because we do not know what kind of VMs, work-intensive or not, they take into account in their calculations. Nevertheless, sticking with 29 VMs per server and each VM belonging to a different client, the VLAN limit is reached with 141 servers, which is a small number of servers for a cloud data center. In [59] the number of servers expected in Amazon data center is 46.000 and in [60] we have

approximations of the number of servers owned by the largest data center companies.

MPLS (Paragraph 4.2.2.1) could be a solution for Layer 3 data centers as it has enough different labels to accommodate for more than a million customers. However it is not widely used in data centers because of a complexity issue concerning the distribution of the labels over the network and because of its cost [61].

In this section we group solutions based on two criteria. The type of isolation (Host isolation or Core isolation) is the first criterion. Then, the second criterion to further separate the solutions is the layer of the underneath infrastructure required by the solution.

5.1 Host isolation for Cloud DC

In this section we present 8 solutions which we consider as Host isolation solutions designed for Cloud DC. Those solutions are: Diverter, BlueShield, NetLord, LISP, NVGRE, STT, VL2, and DOVE.

5.1.1 Host isolation for Layer 2 Cloud DC

Protocols introduced in this section use the Host isolation technique and work over a Layer 2 network.

5.1.1.1 Diverter

Diverter [8] creates an overlay network with a software-only approach, thus alleviating the need for manual configuration of switches and routers. The software module, called VNET, is installed on the host of each physical server. The server's packets (VMs and host packets) and the packets from the network are intercepted by the VNET which processes them. During this process, the VNET replaces the MAC addresses of the packet in order to have no virtual addresses appearing on the core network. The destination MAC address is replaced by the MAC address of the physical server hosting the destination VM. The source MAC address is replaced by the MAC address of the server which hosts the source VM. In the Layer 2 core network, the switches perform packet forwarding using the server's MAC address. Tenant isolation is done thanks to the VNET's control of the packet. If there is no rule in both VNETs allowing the communication between the VMs then the packet is not sent by the ingress VNET. If it is mistakenly sent, then the packet is dropped by the receiver VNET. The control of the packet is done two times at both VNETs. This implies that both VNETs must have the same rules allowing this communication between the two VMs.

In Diverter the tenants cannot choose the addressing scheme they want. They must use IP addresses that follow a specific format which is:

10.tenant.subnet.vm

Where *tenant* is the tenant ID, *subnet* the number of the subnet belonging to the tenant in which the VM is present and *vm* is the number of the vm in the subnet. With this addressing scheme there is no risk of having identical addresses.

In conclusion Diverter provides Layer 3 network virtualization, over a large flat Layer 2 network, in which tenants are isolated. Tenants can also control their own IP subnet and VMs addresses as long as they respect the restrictions on IP addresses imposed by Diverter.

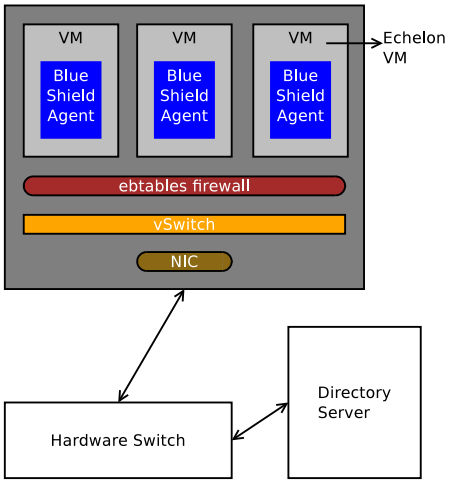


Figure 14: BlueShield architecture (adapted from figure of BlueShied paper [11])

5.1.1.2 BlueShield

BlueShield [11] is an architecture which neither adds a header nor modifies the already existing header. It also does not use a tag or VLAN like value to separate tenants' traffic. Instead, it prevents the tenants' traffic from being sent on the Layer 2 network by blocking address resolution and preventing the configuration of static hardware address entries. With these characteristics, BlueShield provides a complete isolation between tenants.

In order to allow communication between VMs of the same tenant - or not, depending of the tenants' demands - BlueShield uses a BlueShield Agent (BSA) in each VM and a vSwitch at each server (Figure 14). The BSA will see all the ARP requests made by the VM and will convert them in directory look-up (DLU) requests addressed to one or multiple Directory Servers (DSs). The ARP requests are then dropped by the vSwitch of the server before reaching the NIC and the network. The DS searches in its rules whether or not the source VM can communicate with the destination VM. If communication is not allowed, then the DS does not answer and the VMs can not communicate. Otherwise the DS answers the request. As the BSA can send a request to multiple DS, they all answer, so there is a requirement for synchronization of DSs' rules.

BlueShield defines Echelon VMs as those whose task is to increase security and isolation. These Echelon VMs are disseminated on the network and share security rules that they enforce by scanning all the packets passing through them. In order for the traffic to pass through an Echelon VM, the rules in the DS must be modified accordingly. The DS, instead of answering with the MAC address of the destination VM, will send

the Echelon VM MAC address.

In conclusion, BlueShield is a technique which allows tenants' data isolation but not the isolation of tenants' address-space. In addition, the rules in the DS, enforcing this isolation, must be the same on all the DSs and the Echelon VMs, if these last devices are used. The establishment and configuration of these rules lies with the administrator, and the techniques are those of his or her choosing.

5.1.2 Host isolation for both Layer 2 and Layer 3 Cloud DC

In this section we present one Host isolation protocol which can be used over either a Layer 2 or a Layer 3 network.

5.1.2.1 NetLord

In "NetLord: A Scalable Multi-Tenant Network Architecture for Virtualized Datacenters"[13] the authors proposed a new multi-tenant network architecture. The core network is a Layer 2 (Ethernet) network and the use of Layer 3 (IP) is done at the last hop between the edge switch and the server.

To provide tenant traffic isolation in the network, NetLord encapsulates tenant data with both Layer 3 (IP) and Layer 2 (Ethernet) headers (Figure 15). To do so, NetLord uses an agent in the hypervisor of the server to control all the VMs on the server. This agent has to encapsulate, route, decapsulate and deliver the tenant's packet to the recipient.

The source NetLord Agent (NLA) encapsulates the tenant's packet with an IP header and an Ethernet header. The Ethernet destination address of the added Ethernet header is the MAC address of the egress edge switch. The IP destination address of the added IP header is composed of two values. The first is the number of the switch port (P) to which the machine is connected, and the second is the Tenant_ID (TID). This IP address is analyzed at the egress edge switch where it is used to route the packet toward the correct server by using the port number P from the IP address. Then, when the packet is received by the server, it is handed off to the destination NLA. This NLA has to use the TID part of the IP address to send the data to the correct tenant. The use of IP routing at the last hop allows the use of a single edge switch MAC address in order to communicate with the VMs on the server beyond this edge switch. This way, physical and virtual machines, from other servers, will only have one mac address to store, the edge switch MAC address. In addition, the mac addresses of the VMs are not exposed on the core network. However the tenant's ID is exposed in the outer IP header. This exposition can be used by the provider to apply per-tenant traffic management in the core network without the need of per-flow Access Control Lists (ACLs) in the switches.

NetLord also provides address-space isolation for tenants. The tenants are able to use any Layer 2 or Layer 3 addresses because NetLord does not impose restrictions on addresses, and there is no risk of badly routed packets because of these addresses. As stated earlier, the ingress switch will use the MAC address of the egress switch on the core network to forward the data. Then the IP address, composed of the port number and the TID, will be used at the egress switch. At any given time, the addresses defined by the tenant are only visible in the tenant virtual network.

The tenant data between the egress and ingress switches are conveyed over the Layer 2 network thanks to VLANs. In order to choose which VLAN to use, NetLord applies the SPAIN [62] selection algorithm. However to support the SPAIN multipath technique and stock per-tenant configuration information, NetLord uses Configuration Repository which are databases. It also uses the same mechanisms as Diverter [8] to support virtual routing.

To establish a NetLord architecture, edge switches that support IP forwarding must be used, which is not a common feature for commodity switches. In addition, the use of SPAIN implies a scalability issue and there is no support for bandwidth guarantee. In conclusion, NetLord provides tenant isolation but has some drawbacks in other areas.

5.1.3 Host isolation for Layer 3 Cloud DC

The protocols introduced in this section use the Host isolation technique and require a Layer 3 network.

5.1.3.1 LISP: The Locator/Identifier Separation Protocol

The Locator/Identifier Separation Protocol (LISP), presented in "RFC 6830" [2], aims at splitting the routing and the addressing functionalities. Currently, the IP address, a single field, is used both for routing and for addressing a device. In LISP, the routing functionality is done by Routing Locators (RLOCs) and the addressing functionality is done by Endpoint Identifiers (EIDs). An RLOC is an address, the same size as an IP address, of an Egress Tunnel Router (ETR). This RLOC indicates the location of the device in the network. This value is the one used by the Ingress Tunnel Router (ITR) to route the packet through the network toward the ETR. The ETR is the gateway of the private network. Then, to route the packet to the correct node in the private network the EID value is used. All the EID of the private network is mapped in the ETR. This value also has the same length as an IP address (32-bit for IPv4, or 128-bit for IPv6). Such a split is done by using different numbering spaces for EIDs and RLOCs. By doing this, LISP improves the scalability of the routing system thanks to the possibility of a greater aggregation of RLOCs than IP addresses. However in order to have this better aggregation, the RLOCs must be allocated in a way that is congruent with the network's topology. On the other hand, the EIDs identify nodes in the boundaries of the private network and are assigned independently from the network topology.

The encapsulation of the packet, in an IPv4 network, is shown in Figure 16. The outer header is the IP

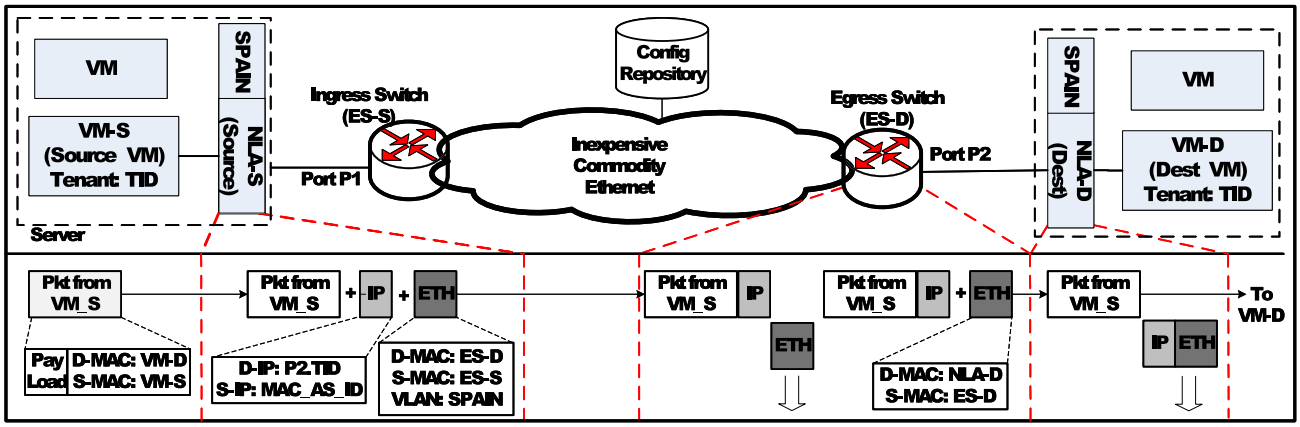


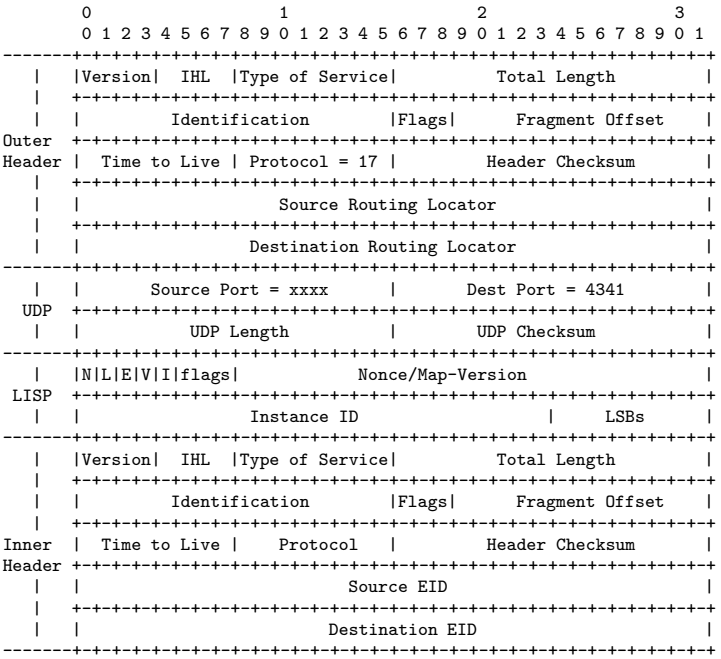
Figure 15: NetLord architecture (figure from NetLord paper [13])

header with the RLOCs addresses as the source and destination addresses. Then the UDP header is added and followed by the LISP header which is 4 bytes long. The inner header is also an IP header but the source and destination addresses are now the EIDs addresses.

It is a tunneling protocol that simplifies routing operations such as multi-homed routing and facilitates scalable any-to-any WAN connectivity. It also improves the scalability of the routing system through greater aggregation of RLOCs. However to benefit from LISP advantages it must use a LISP-enabled ISP. The ISP also benefits from using LISP because it has less information in his routing devices thanks to RLOCs aggregation.

5.1.3.2 NVGRE: Network Virtualization using Generic Routing Encapsulation

Network Virtualization using Generic Routing Encapsulation (NVGRE), detailed in "NVGRE: Network Virtualization using Generic Routing Encapsulation" [3], is based on the Generic Routing Encapsulation (GRE) [40] encapsulation method. NVGRE allows the creation of virtual Layer 2 topologies on top of a physical Layer 3 network. The goal of NVGRE is to improve the handling of multitenancy in data centers. Network Virtualization is used in order to provide both isolation and concurrency between virtual networks on the same physical network infrastructure. To improve isolation, NVGRE modifies the GRE header by replacing the Key field with two fields, virtual Subnet ID (VSID) and FlowID (Figure 17). The first 24 bits are for the VSID field and the following 8 bits for the FlowID field. The VSID is used to identify the virtual Layer-2 network. With its 24 bits it is possible to have 2^{24} virtual layer-2 networks which is more than the 4096 VLANs. The flowID is used to provide per-flow entropy in the same VSID. This NVGRE packet can then be encapsulated in both versions of IP whereas NVGRE cannot contain a 802.1Q tag. The NVGRE tunnel needs NVGRE endpoints between the virtual and physical networks. Those endpoints could be servers, network devices, or part of a hypervisor. NVGRE is using the IP address scalability to lower the size of Top of Rack




```
Inner Header = header on the datagram received from the originating host
Outer Header = header prepended by an ITR
IHL = IP-Header-Length
N = nonce-present
L = 'Locator-Status-Bits' field enabled
E = echo-nonce-request
V = Map-Version present
I = Instance ID
flags = 3-bit field reserved for future flag use
LISP Nonce = 24-bit value that is randomly generated by an ITR when
the N-bit is set to 1
LISP Locator-Status-Bits (LSBs) = set by an ITR to indicate to an ETR
the up/down status of the Locators in the source site when the L-bit
is also set
```

Figure 16: LISP IPv4-in-IPv4 Header Format

switches' MAC address table. For the moment, the fact that NVGRE is a work in progress and not a standard prevents it from being widely deployed, while awaiting possible modifications.

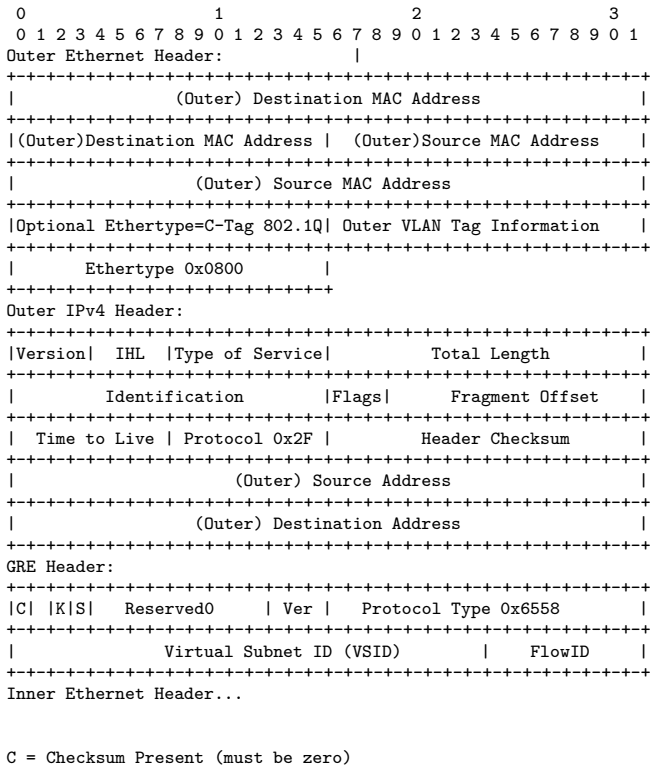


Figure 17: NVGRE Header

5.1.3.3 STT: Stateless Transport Tunneling Protocol

The Stateless Transport Tunneling Protocol (STT), introduced in "A Stateless Transport Tunneling Protocol for Network Virtualization (STT)" [4], is a new IP-based encapsulation and tunneling protocol which adds a new header (Figure 18) to the packet and also modifies the TCP header. The new header contains a 64-bit Context ID field which can be used to differentiate $2^{64} \approx 1.8 \times 10^{19}$ virtual networks. The modifications done to the TCP header are about the meaning and use of both the Sequence Number (SEQ) and the Acknowledgment Number (ACK). The SEQ field is now divided into two parts. The upper 16 bits of the SEQ field are used to indicate the length of the STT frame in bytes. The second part of the SEQ field, the lower 16 bits, is used for the offset, expressed in bytes, of the fragment within the STT frame. Reusing the TCP header allows the STT to be easily encapsulated in IP datagrams. The Protocol Number field for IPv4 or the Next Header field for IPv6 have the same value as for regular TCP. An additional difference between TCP and STT is that STT, as the name indicates, does not use state.

The goal of STT is to tunnel packets efficiently so it supports Standard Equal Cost Multipath (ECMP). Nevertheless, STT imposes that all the packets belonging to the same flow follow the same path. The multipath is done on a flow basis and not a packet basis. However, the most important drawback of STT is the fact that there must not be any middle boxes on the path. If those middle boxes are present, then they have to be configured to let STT frames pass through. This implies that access to the middle boxes is required. So, for the moment, it is not feasible to have an STT tunnel between two sites linked by an unmanageable network. In addition, STT is not a standard, so not all devices will be able to work with it.

Figure 18: STT header

5.1.3.4 VL2

VL2, presented in [15], is an architecture designed to allow agility, and notably the capacity to assign

any server to any service. To do so, VL2 uses two different IP address families, the Location-specific IP addresses (LAs) and application-specific IP addresses (AAs). This addressing scheme separates server names, the AAs addresses, and their locations, the LAs addresses. However, this implies that a mapping between AAs addresses and LAs addresses is needed. This mapping is created when application servers are provisioned to a service and assigned AAs addresses. This mapping is then stored in a directory system which must be reliable. To improve this reliability the directory system can be replicated and can use several directory servers but this implies that those directory servers are synchronized. The directory system is used to achieve addresses resolution but every server must implement a module called a VL2 agent to contact the directory system. This VL2 agent contacts the directory system to retrieve the LA address corresponding to an AA address. Each AA address is associated with an LA address. This LA address is the identifier of the Top of the Rack switch to which the server, identified by the AA address, is connected. The AA address remains the same even if the LA address is changed due to a virtual machine migration or re-provisioning. The AAs addresses are assigned to servers and the LAs addresses to the switches and interfaces. To do this LA address assignment, switches run an IP-based link state routing protocol.

VL2 works over a Clos topology [63] and a Layer 3 network. This network routes traffic by LAs addresses, so in order to route the traffic between servers with AA addresses, encapsulation is needed. This encapsulation is done by the VL2 agent which encapsulates the IP packet in an IP packet (IP-in-IP) and uses the associated LA address, in the directory system or its local cache, with the AA address destination address.

In VL2, the isolation of the server is achieved through the use of rules in the directory system. Additionally, those rules are enforced by each VL2 agent. For example if a server is not allowed to send a packet to a different server, the directory service will not provide an LA address to the VL2 agent for the packet which will be dropped by the VL2 agent.

5.1.3.5 DOVE: Distributed Overlay Virtual nEtwork

Distributed Overlay Virtual nEtwork [16, 17] is a technique designed with a centralized control plane over a Layer 3 network. DOVE does not provide an encapsulation protocol and uses others protocols, such as VXLAN, NVGRE, or STT, as long as those protocols allow the use of two parameters.

The first parameter is the virtual network ID and the second, a policy specifier defined by a domain ID which is optional. By not providing an encapsulation protocol, DOVE is not limited to Ethernet emulation and could be used over a Layer 2 network.

Dove provides tenant isolation thanks to the use of an encapsulation protocol whose header is added by dSwitches, and the use of the DOVE Policy Service (DPS). The dSwitches are the edge switches of the DOVE overlay network. They are used in each physical server to act as the tunnel endpoint for the VMs of these servers. The DPS is a unique component in the DOVE network, whose function is to process dSwitches policy requests. It maintains all the information regarding existing virtual networks as well as their correlation with the physical infrastructure, policy actions, and rules. It is thanks to these policy requests and responses that a dSwitch knows if a VM can communicate with a different VM, and learns the address of the dSwitch which manages the other VM.

As a solution using a centralized control plane, DOVE needs a "highly available, resilient, and scalable" device to host the DPS as stated by the authors. As we have seen in the PortLand architecture, the Fabric manager needed at least 15 CPU cores working nonstop to process the ARP requests for 27.648 end hosts which each make 25 ARP requests per second. Here in DOVE, the issue is worse with the DPS. In PortLand, it was just a database query to retrieve a PMAC address. In DOVE, the lookup searches for the address of the dSwitch, and corresponding policy rules and actions in order to determine the next action.

5.2 Core isolation for Cloud DC

In this section we introduce protocols using the Core isolation technique and with multi-tenant capabilities. We divide them into three categories depending on the Layer of the underneath network.

5.2.1 Core isolation for Layer 2 Cloud DC

The five protocols presented in this section required that the underneath network be a Layer 2 network in order to be used.

5.2.1.1 802.1ad (QinQ)

The IEEE 802.1ad standard [5] also known as "QinQ" is in fact an amendment to the IEEE standard 802.1Q. This amendment enables service providers to offer isolation to their customers' traffic. In addition to the VLAN information of the client, defined by the 802.1Q standard, this new 802.1ad standard defines another VLAN for the provider. The customer VLAN header is called the C-TAG (customer TAG) and is the inner header. The outer header is the S-TAG (Service TAG) for the provider. In Figure 19 we represent the two VLAN headers. The TPID0 field has a default value of 0x88A8 which is different than the default value (0x8100) of the 802.1Q standard. TPID1 is configured with the default value 0x8100. This differentiation indicates to the switch that there are two TAGs.

Thanks to the S-TAG header, the provider can manage only one VLAN for all the VLANs of one client. He is able to provide $2^{12} = 4096$ VLANS to each of his 4096 clients which results in $2^{12} * 2^{12} = 16777216$ different VLANs. If it is the solution chosen by the provider then the 802.1ad VLAN management is identical to the 802.1Q VLAN management because the provider only cares for the S-TAG header. It is the

client responsibility to manage his/her 4096 VLANs in his/her network.

However, most of the time, one client does not need 4096 VLANS and the provider has more than 4096 clients. So instead of using both TAGs separately, the provider adds both TAGs in order to have 16777216 VLANs. This way the management of such a solution is more complex than the 802.1Q solution, but yields greater scalability. For switching the frames, the switches have to recover the VID of both TAGS and verify in a database with up to 16777216 values instead of 4096. This implies more work to obtain the VLAN ID and consequently more time to verify the VLAN ID in the database, it also uses more memory space because of its increased size. It means more CPU, more memory, and more latency at each switch. Additionally, irrespective of the way both TAGs are managed, the overhead is increased by four bytes.

The advantage of the 802.1ad standard is that it raises the limit of VLANs possible from 4096, with 802.1Q, to 16777216, which should be sufficient for network growth during the next few years. If this new limit is still too small then there is the possibility to use the 802.1ad VLAN TAG stacking solution and add more VLAN TAGs to the header. However it is a non-standard solution and might results in overhead issues because each time we add a VLAN TAG, the header increases by four bytes for only 12 bits of VID.

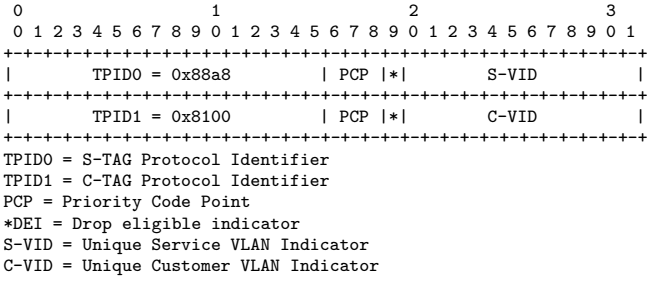


Figure 19: 802.1ad header

5.2.1.2 802.1ah (mac-in-mac)

The 802.1ah IEEE standard [6] was developed after the 802.1ad standard [5] in order to provide a method for interconnecting Provider Bridged Networks. This protocol is intended for network providers in order to attend to their needs for more service VLANs. This standard is also known as Provider Backbone Bridges (PBB) or "mac-in-mac". As the last name indicates, the idea is to add another MAC header on top of the existing MAC header. This new MAC header is added by a Provider Edge (PE) switch. This allows for the core network switches to only save the MAC of the PE switches, thus no MAC information of the client are used for switching inside the core network. All the mapping work is done by the PE switches and they are the ones responsible for encapsulating and decapsulating the messages.

The encapsulation is done by adding a new MAC header to the message. This new MAC header (Figure 21) is composed of two different MAC headers. The first one is the new header from the 802.1ah standard. This MAC header can be divided in two parts. The first part is the one with the Backbone components. The fields of this part are:

- MAC Backbone Destination Address (B-DA), 6 bytes long
- MAC Backbone Source Address (B-SA), 6 bytes long
- EtherType with a size of 2 bytes and a value of 0x88a8
- Priority Code Point (3 bits) and the Drop Eligible Indicator (1 bit)
- Backbone VLAN indicator (B-VID) with a size of 12 bits

After this Backbone part, the second part, called the Service encapsulation, is three bytes long and contains the following fields:

- EtherType with a value of 0x887e on two bytes
- Priority Code Point (3 bits) and the Drop Eligible Indicator (1 bit)
- Used Customer Address (1 bit) indicates if the customer address is valid or not
- Interface Service Instance Indicator (I-SID) with a size of 20 bits

With this new 802.1ah header we now have another MAC header for the provider to use. There are now 4096 VLANs possible with the B-VID. In each VLAN there are $2^{20} = 1048576$ supported services with the I-SID field. This could amount to a total of 4294967296 VLANs with only the new header. Additionally only the PE switches have to learn the customers' MAC addresses (C-DA and C-SA) and have to add and suppress the new 802.1ah header.

The 802.1ah standard is an evolution of the 802.1ad standard which is an evolution of the 802.1Q standard. Each new standard has added information in the header of the message. The Figure 20 shows the evolution of the 802.1 header. We can see that in order to increase the number of VLAN identifiers, the size of the header keeps increasing. This increase implies, as stated in the 802.1ad standard 5.2.1.1, that switches, at least the PE switches, use more CPU time to process the header and use more memory to save all the information of the VLANs.

5.2.1.3 Private VLANs

Private VLANs is a solution developed by Cisco and presented in "RFC 5517" [65]. This solution is based on the aggregated VLAN model proposed in "RFC 3069" [66]. The idea is to have a principal VLAN subdivided with secondary VLANs. The principal VLAN broadcast domain is therefore divided in smaller subdomains. A subdomain is defined by the designation

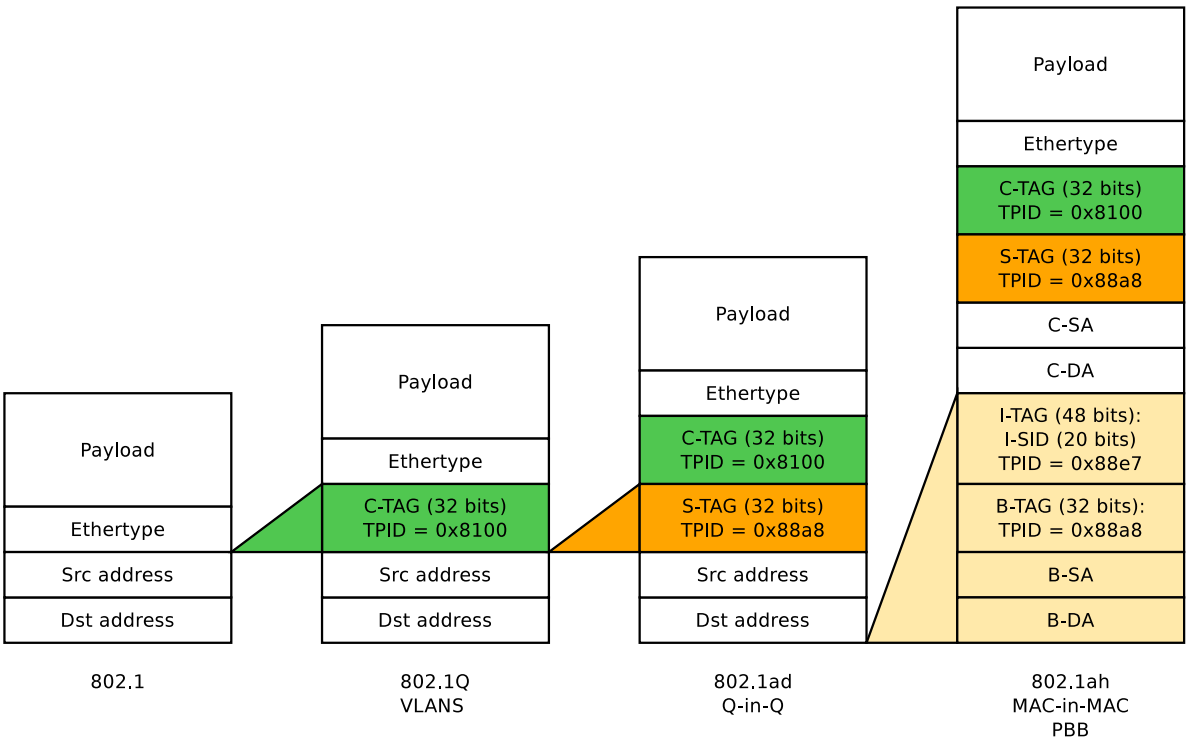


Figure 20: 802.1, 802.1Q, 802.1ad and 802.1ah frame formats (figure from "IEEE 802.1ah Basics"[64])

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             B-DA                              ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 B-DA (cont.)                 |    B-SA                           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 B-SA (cont.)                                                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       TPID0 = 0x88a8          | PCP |*|         B-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       TPID1 = 0x88e7          | PCP |*|!| RES |      I-SID      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 I-SI (cont.)                  |     C-DA                          ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 C-DA (cont.)                                                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             C-SA                              ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 C-SA (cont.)                  |        TPID2 = 0x88a8         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PCP |*|        S-VID          |        TPID3 = 0x8100         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PCP |*|        C-VID          |  Ethertype of Payload         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Payload ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
B-DA = Backbone Destination Address
B-SA = Backbone Source Address
TPID0 = B-TAG Protocol Identifier
PCP = Priority Code Point
*DEI = Drop Eligible Indicator
B-VID = unique Backbone VLAN Indicator
TPID1 = I-TAG Protocol Identifier
!UCA = Used Customer Address
(0=Valid, 1=Not Valid)
RES = Reserved
I-SID = Interface Service Instance Indicator
C-DA = Customer Destination Address
C-SA = Customer Source Address
TPID2 = S-TAG Protocol Identifier
S-VID = unique Service VLAN Indicator
TPID3 = C-TAG Protocol Identifier
C-VID = unique Customer VLAN Indicator
```

Figure 21: 802.1ah header

of the switch's ports group. In [65] there are three port designations. These port designations are as follows:

1. Isolated port: An isolated port can not talk with an isolated port or a community port.
2. Community port: A community port belongs to a group of ports. Those ports can communicate between themselves and with any promiscuous port.
3. Promiscuous port: A promiscuous port can talk with all the other ports.

In order to create the subdomains within a VLAN domain, the VLAN ID is not enough. An additional VLAN ID is used. To refer to a specific Private VLAN, at least one pair of VLAN IDs is necessary. A pair of VLAN IDs is composed of one primary VLAN ID and of one secondary VLAN ID. The primary VLAN ID is the VLAN identifier of the whole Private VLAN domain. This scheme of VLAN pairing only requires the traffic from the primary and secondary VLANs to be tagged following the IEEE 802.1Q standard. It only uses a single tag at most, thanks to the 1:1 correspondence, between a secondary VLAN and its primary VLAN. The Private VLAN technique allows for a greater number of VLANs thanks to the recycling of VLAN IDs in secondary VLANs. It also allows for better addresses assignment in a domain because these addresses are shared between all the members of the private VLAN domain.

5.2.1.4 PortLand

The Portland architecture [9] is one that uses a centralized control plane, with the core network working

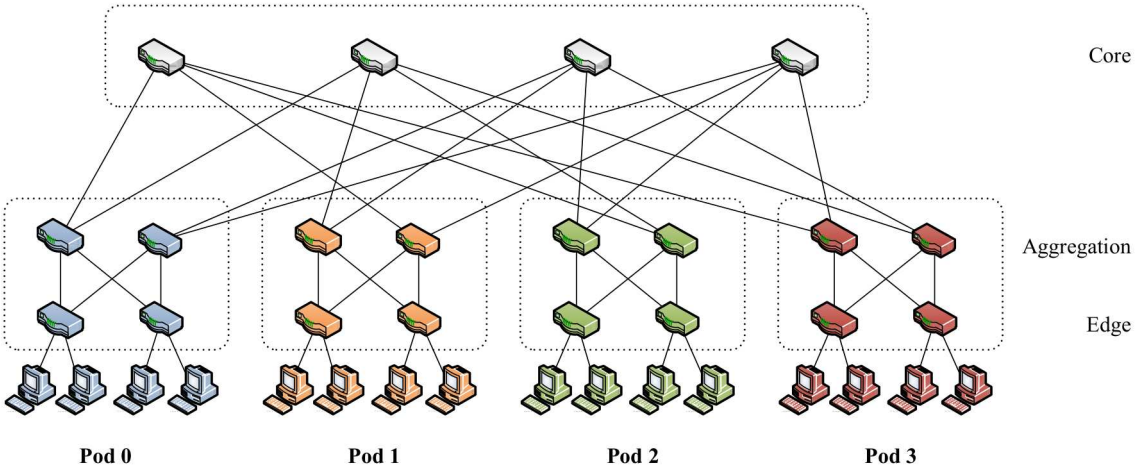


Figure 22: A fat tree topology (figure from PortLand paper [9])

at Layer 2. The topology of the network must be a multi-rooted fat-tree [67] as in Figure 22. To manage forwarding and addressing, a fabric manager is used. A fabric manager is a user process running on a dedicated machine which manages soft states in order to limit or even eliminate the need for administrator configuration. In addition to the fabric manager, Portland introduces new MAC addresses for each end host. This new MAC address, called Pseudo MAC (PMAC) address, encodes the position of the end host in the topology. The PMAC is 48 bits long and is composed of four parts.

1. pod (16 bits) is the pod number of the edge switch
2. position (8 bits) the position of the end host in the pod
3. port (8 bits) is the switch port number the host is connected to
4. vmid (16 bits) is used to differentiate the virtual machine on the physical machine

The PMAC is not known by the end host which keeps using its actual MAC (AMAC) for its packets. When an edge switch sees a new AMAC, coming from a connected machine, it has to create a new PMAC and map the IP address with the AMAC and the PMAC. It then has to announce the new mapping between the IP address and the PMAC address to the fabric manager. This way when an edge switch wants to forward a message with only the IP address, it will do an Address Resolution Protocol (ARP) request which will be processed by the fabric manager and receive the PMAC address in the answer. Edge switches are responsible for mapping the PMAC to the AMAC. They also have to replace the AMAC with the PMAC for outgoing packets and replace the PMAC with the AMAC for arriving packets. This way Portland can uses a hierarchical PMAC addressing with only the pod part, the first eight bits, used for forwarding the data through the core switches. Then at the aggregation switches, the position part is used and, at the next level, the edge switches uses the port part. Finally the last part, the vmid, is used by the server to know to which VM to deliver the packet.

Portland design implies that the fabric manager learns all the correspondences between PMAC addresses and IP addresses and uses this table to answer ARP requests from the edge switches. The edge switches then use the PMAC addresses they received to change the destination addresses. Since PMAC addresses are hierarchical, this enables switches to have smaller forwarding tables.

The fact that the fabric manager, a single machine, has to manage all the ARP traffic makes this architecture not able to scale. For example in a data center with 27,648 end hosts (not tenants) and each host makes 25 ARP requests per second, the fabric manager will need approximatively 15 CPU cores working non-stop to only manage the ARP requests.

5.2.1.5 SEC2 : Secure Elastic Cloud Computing

Secure Elastic Cloud Computing(SEC2) [10] is an architecture which uses a centralized control plane over a Layer 2 core network. In this architecture, the network is divided in one core domain and several edge domains (Figure 23). An edge domain possesses an identifier, the edge id (eid), which is unique among the edge domains. Each edge domain is connected to the core domain via Forwarding Elements (FEs) which manage address resolution and enforce policy rules. In an edge domain, tenants are isolated thanks to the use of VLANs. A tenant's subnet is identified by a unique Customer network id (cnet id). This implies that there is a 1:1 correspondence between a VLAN ID and a cnet id. In SEC2 there are only 4096 tenants in an

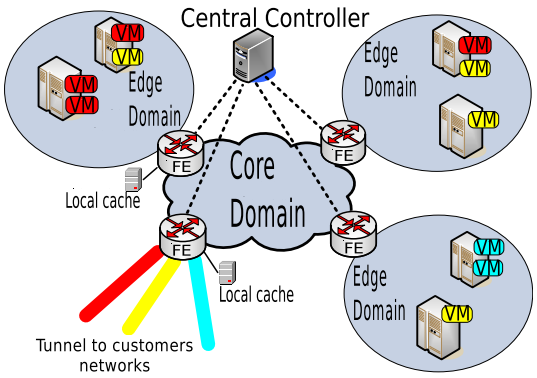


Figure 23: SEC2 Architecture (adapted from figure of SEC2 paper [10])

edge domain. However, the VLAN ID can be reused in a different edge domain so the maximum number of VLANs allowed does not limit the number of tenants. As SEC2 does not limit the number of edge domains, to increase the number of tenants, the solution is to create a new edge domain. A tenant's VMs are identified by the combination of the cnet id and their IP addresses. These IP addresses are freely chosen by the tenant and there is no restriction on them. The VM MAC address is then mapped to this combination (cnet id, IP). The MAC address is not needed by the FE to forward the packets because the pair (cnet id, IP) is unique.

For resolving addresses and enforcing rules, FEs must obtain the information from the Central Controller (CC) on an on-demand basis. When a VM wants to send a packet to a different VM, the VM only knows the IP address and therefore will do an ARP request. This ARP request is intercepted by the FE which looks in its local cache to see if the answer is present. If not, it sends a request to the CC which answers with information such as the MAC address of the receiver, the eid of the domain in which the receiver is located, and the VLAN ID of the receiver. The FE then answers the ARP request with the MAC address. The other information is saved in the FE's local database. When the packet reaches the ingress FE, the FE will encapsulate the packet with a MAC header. The destination address will be the eid previously received, and the VLAN number will be replaced by the one received from the CC.

For the CC to be able to answer the FE requests, the core, edge, and network information must be stored. The following mappings are maintained by the CC:

- VM MAC ↔ (cnet id, IP). To resolve the IP address of a VM and obtain its MAC address.
- VM MAC ↔ edge domain id (eid). To know in which edge domain the VM is located.
- eid ↔ FE MAC address list. To determine between which FE to establish the data tunnel if there are multiple FEs for a single edge domain.
- (cnet id, eid) ↔ VLAN id. To identify which VLAN to use in the receiver edge domain.
- cnet id ↔ rules and actions. To know if both tenants agree to communicate.

To allow inter sub-network communication, each tenant must have at least one public IP address stored in the CC.

Even if the design uses a centralized controllers, the CC can be distributed and the information can be divided per tenant. The author provides an example:

> For example, different customers can be assigned to different CCs by using Distributed Hash Table (DHT). Since the management and policy control of different customer networks are relatively independent, such partition does not affect the functionality of CC.

SEC2 provides tenant isolation and also address-space isolation thanks to the use of VLAN in edge domains. FEs enforce the rules and policies that are stored in the CC, which also prevents inter-tenant communication if it has not been previously agreed.

5.2.1.6 VNT: Virtual Network over TRILL

In [14] the authors propose a new technique of overlay network done over a Layer 2 network using the Transparent Interconnection of Lots of Links (TRILL) protocol [68, 69]. This overlay network is called Virtual Network over TRILL (VNT). In order to provide tenant isolation, VNT adds a Virtual Network Identifier (VNI) field in the TRILL header (Figure 24). The VNT header is composed as follows. The first 64 bits correspond to the basic TRILL header. They are followed by a block of 32 bits describing the criticality of the options. Then there are a reserved field of 18 bits and a flow ID field of 14 bits. The VNT extension is added as an option in the header, with the Type, Length, Value (TLV) format, and need a 64-bit block. The VNI field is 24 bits long and can differentiate approximatively 16 million tenants. A VNI is unique in the core network and is associated with one tenant. To apply this VNT extension to the packet, a new network component is introduced and is called a Virtual Switch (VS). A VS has to manage all the interfaces with the same VNI Tag (all the interfaces of one tenant). The provider administrator must link every new VMs of a tenant to the unique VS managing the VNI Tag associated with the tenant.

Tenants are free to use any Layer 2 or Layer 3 address they want in their virtual network. These addresses are not visible in the core network and cannot affect packet routing. The routing of the tenant data is done via two different routing techniques at different layers. The first routing, the virtual routing, is done at Layer 2, in the core network, through the VNI tag and some rules in the RBridges. A RBridge can only send a packet toward another RBridge or an end host if they share the same VNI tag. The second routing is done at Layer 3, and is dependent on both the tenant's network and the tenant's Layer 3 endpoint configuration.

This overlay technique allows an isolation of tenants' data thanks to a VNI Tag in the TRILL header. This VNI Tag is a 24-bit value which allows for approximately 16 million different tenants, which is better than the limit of 4096 VLANs. The VNI Tag is also used to establish unique tree topology for each virtual network associated to this VNI Tag using the intermediate system to intermediate system (IS-IS) protocol [70, 71]. This way the data tagged with a VNI will only be propagated along this tree and will not be send to other tenants' host, which ensure tenant isolation. Moreover the packets are routed based on the VNI tag in the physical network which isolates the space address of each tenant, so that a tenant can use Layer 3 and Layer 2 addresses.

However, VNT being based on TRILL, it is impossible to interconnect multiple data center without merging their control plane into one, resulting in losing each data center independence and increasing the broadcast domain. To prevent the merging of TRILL network when being interconnected and to keep each data center control plane independent, the Multi-Level TRILL Protocol with VNT (MLTP/VNT) solution has been developed. This solution, describe in [72], mostly improve TRILL scalability, thus allowing for a better use of VNT.

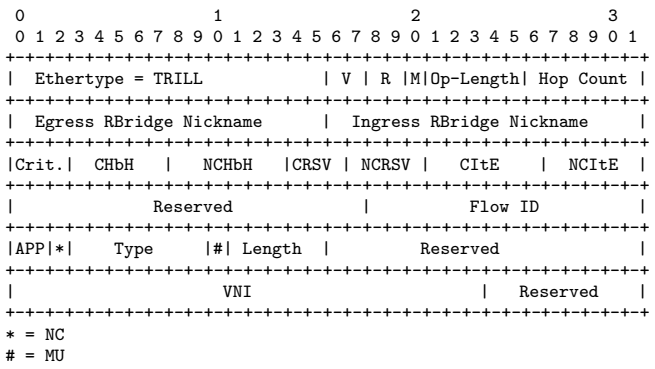


Figure 24: VNT Header

5.2.2 Core isolation for both Layer 2 and Layer 3 Cloud DC

The protocol introduced in this section uses the Core isolation technique and works over a Layer 2 and/or Layer 3 network.

5.2.2.1 VSITE

In [12], the VSITE architecture is proposed in order to allow enterprises (companies) to have seamless Layer 2 extensions in the cloud. In the paper, the tenants are considered to be exclusively enterprises that need to expand their networks. VSITE defines this extension as the collection of resources of an enterprise within a data center, and called it a virtual stub network, or vstub. The enterprise customer edge (CE) switch communicates with the cloud edge switch to exchange MAC information via an OTV-like protocol. For communication over the public network, VSITE uses Ethernet over IP (Ethernet over GRE (Section 4.1.1.1) or EtherIP [73] protocol) to transport data between the cloud edge switch and the CE. This OTV-like protocol uses a control plane protocol to exchange MAC addresses among sites which eliminates the cross-site MAC flooding. To suppress this flooding without modifying the VMs behavior, the hypervisor is tasked to intercept all the VMs' DHCP and ARP messages.

In VSITE there is no global VLAN ID assigned to enterprises but rather, local VLAN IDs at the data center edge location (between the switch and the VMs connected to it). The VLAN IDs are not statically assigned to enterprises. Therefore, the cloud edge switch has to map the VLAN ID of the enterprise to a locally-significant unique VLAN ID for the traffic from the enterprise's network to the VMs. The reverse operation has to be done at the hypervisor for the traffic from the VMs to the enterprise's network.

To ensure isolation of tenants' data, VSITE encodes the tenants' ID in the MAC addresses. The hypervisor will then ensure the traffic isolation by verifying that the VM receiving the packet belongs to the enterprise through the tenant ID in the MAC address. The hypervisor, by checking the tenant id, must either accept or drop the packet. The MAC address (48 bits long) is divided in two:

1. X bits for a tenant's id
2. 48 - x bits for the VM's id

Where X value is the administrator's choice.

The core network of the data center can be a Layer 2 or a Layer 3 network. In a Layer 2 network situation, the MAC-in-MAC encapsulation technique allows for a location MAC address (locMAC). With a Layer 3 network, the packet is encapsulated in an IP packet with a location IP address (locIP). The locIP or locMAC are location addresses assigned to a VM. Each VM possesses a location address which allows the separation of its name and location. The name of the VM is the IP address assigned by the enterprise, the enterprise IP (entIP). The location of the VM is indicated by the IP address, or the MAC address, of the switch to which the VM is logically connected. However this logical connection must be done via Ethernet. This location address is used to route the packet in the core network. All locIP (or locMAC) addresses are stored in a directory server. Because of this, a VM or data center edge has to send a lookup request to the directory server to retrieve the locIP of the destination if the information is not already in its local cache. The directory server maintains the mapping between entIP and pertinent information including locIP, MAC, and potentially, a VLAN ID.

The VSITE architecture has a centralized control plane and relies on hypervisor security to provide protection against MAC address spoofing, a VM impersonating a different VM, or a DDOS attack from a VM. It also uses Core isolation protocols in both its edge domains (VLANS) and its core network (MAC-in-MAC or IP-in-IP). However, data transported over

the public network is not protected.

5.2.3 Core isolation for Layer 3 Cloud DC

The protocols introduced in this section use the Core isolation technique and require a Layer 3 network.

5.2.3.1 VRF: Virtual Routing and Forwarding

Virtual Routing and Forwarding (VRF), described in [74], is a technology included in routers. This technology allows a router to have multiple instances of a routing table to exist and work at the same time. With these multiple routing tables it is possible to segment the network and separate users in different routing table instances. The Customer Edge (CE) router does the local routing and then exchanges the information with the Provider Edge (PE) router. The PE router creates a VRF instance with the information from this CE. This new VRF instance is used by the PE for every packet to and from this CE. For each CE corresponds a VRF instance in the PE. This allows for the creation of a Virtual Private Network (VPN). The PE router creates at least one routing table instance for each VPN. Then the PE routes the packet from each VPN by searching in the corresponding VRF instance. The separation of the VRF instances provides a secure access to all the devices in a specific VRF instance. It also allows the use of the same or overlapping IP addresses without conflict. The VRF method is intended to help the ISPs provide private networks to their clients while using the same physical infrastructure. These private networks, over shared infrastructures, are called Virtual Private Networks (VPNs). It is as if the whole network of each client is tunneled. This solution is based on MPLS for the routing of the packets in the core network. It is not used in data centers.

5.2.3.2 VXLAN: Virtual eXtensible Local Area Network

Virtual eXtensible Local Area Network (VXLAN), detailed in "draft-mahalingam-dutt-dcops-vxlan-09" [7], is being developed in order to expand the VLAN method and remove the VLAN limit of 4096. VXLAN allows overlaying a Layer 2 network on a Layer 3 physical network. Such tunnels begin and end at VXLAN Tunnel EndPoints (VTEPs). The ingress VTEP encapsulates the packet and adds a VXLAN header of 8 bytes to the packet. The header (Figure 25) is composed of 8 bits of Flags, 1 bit for each flag, with the I flag, the 5th one, set to 1 for a valid VXLAN Network ID (VNI). Then comes a 24-bit long reserved field. It is followed by the VXLAN Network Identifier (VNI) field, 24 bits long, used to identify which individual VXLAN overlay network the packet belongs to. Finally the header ends with another reserved field with a length of 8 bits. Once the VXLAN header is added, the packet is then encapsulated within a UDP header. This UDP header must use the value 4789 as destination port. This value has been assigned by the IANA as the VXLAN UDP port. The source port is provided by the VTEP. Both headers are removed at the egress VTEP.

One advantage of VXLAN is that it expands the VLAN technology with a larger number of VXLAN possible. On the other hand, VTEPs must not fragment encapsulated VXLAN packets and if such a packet has been fragmented along the path it must be silently discard by the egress VTEP. As UDP is used to encapsulate the data and that a packet must be discarded silently, the source does not know that its packet has been discarded. There is no security measure so it is recommended to use IPSec to add security mechanisms.

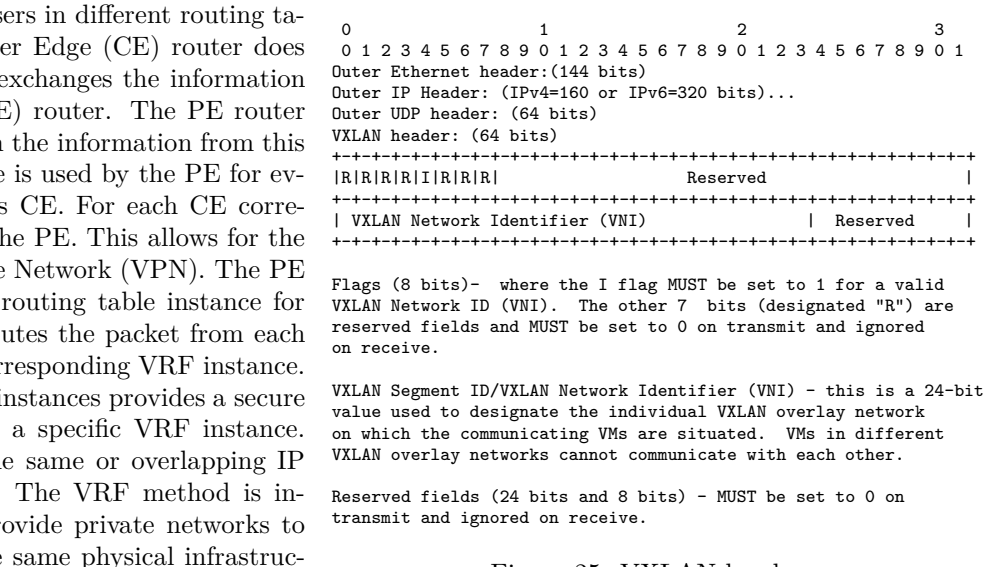


Figure 25: VXLAN header

6 Network isolations provided by several cloud tools

This section regroups a succinct list of several tools used in cloud deployment which provide network isolation while relying on already established tunneling protocols.

6.1 Cisco guide

In [19] tenant isolation is done thanks to path isolation and device virtualization. Path isolation is as if the packet from this path went through a tunnel over the network (Figure 3). The device virtualization is achieved by creating virtual devices such as VMs or virtual switches.

Path isolation is done thanks to several techniques which provide an independent logical path over a shared infrastructure. To create these paths, two technologies, over different layer, are mainly used:

1. Layer 2 separation with VLAN
2. Layer 3 separation with VRF

Separation on the Layer 2 is done thanks to Virtual Local Area Network (VLAN) presented in Section 4.2.1.1.

At the Layer 3, the separation is done with Virtual Routing and Forwarding (VRF) presented in Section 5.2.3.1.

This solution works for small multi-tenancy clouds, with less than 4096 tenants, which corresponds to the VLAN limit. However in today's cloud data center with virtualization, there is a need to host more than 4096 tenants so this solution is not scalable enough.

In [20] tenant isolation is done with additional logical mechanisms such as virtual Network Interface Controllers (vNICs), IPsec or SSL Virtual Private Networks, packet filtering, and firewall policies.

The Security Architecture for the Internet Protocol (IPsec) is a security architecture focused on IP-Layer security, as mentioned in "RFC 4301" [75]. The Secure Sockets Layer (SSL) Protocol is designed to provide privacy and reliability between two communicating applications as described in "RFC 6101" [76].

6.2 Openstack software

Openstack manages tenant isolation thanks to the use of VLANs or Layer 2 tunneling with GRE, as stated in the OpenStack Security Guide [77]. For the Layer 2 tunneling, OpenStack Networking currently supports GRE (Section 4.1.1.1) and VXLAN (Section 5.2.3.2). However the support of VXLAN is added via the Neutron Modular Layer 2 (ml2) plugin.

The security guide explains that "the choice of technology to provide L2 isolation is dependent upon the scope and size of tenant networks that will be created in your deployment". Indeed if the environment has a large number of Layer 2 networks (ie. more than 4096 tenants with each their own sub-network), the VLAN limit could be reached and no more tenants could be added. Therefore it is better to use the tunneling method with GRE or VXLAN.

6.3 OpenFlow controller

OpenFlow [78] has a centralized control plane and uses an OpenFlow controller. This OpenFlow controller has a global view of the network and decides what is best for the network. It then sends its decisions to the compatible OpenFlow network switches. These switches can be hardware or software. Belonging to this last category, Open vSwitch is a virtual switch for hypervisors. It provides network connectivity to virtual machines. Open vSwitch works via a flow table which defines rules and actions for each flow.

In order to isolate the flows from each tenant, OpenFlow uses VLANs (Section 4.2.1.1) and GRE (Section 4.1.1.1) as a tunneling protocol. However Open vSwitch can also manage the Stateless Transport Tunneling (STT) protocol (Section 5.1.3.3) inherited from Nicira development.

6.4 Amazon's Virtual Private Cloud (VPC)

Amazon's VPC [79] is a proprietary solution so we only have scarce information. It provides a Layer 3 abstraction with a full IP address space virtualization. It is possible to create up to 200 sub-networks per VPC and to have up to 5 VPCs per customer. In order to communicate with Amazon's VPN you must have:

- The ability to establish IKE Security Association using Pre-Shared Keys (RFC 2409) [80].
- The ability to establish IPSec Security Associations in Tunnel mode (RFC 4301) [75].
- The ability to establish Border Gateway Protocol (BGP) peering (RFC 4271)[81].
- The ability to utilize IPSec Dead Peer Detection (RFC 3706) [82].
- The ability to adjust the Maximum Segment Size of TCP packets entering the VPN tunnel (RFC 4459) [83].
- The ability to reset the "Don't Fragment" flag on packets (RFC 791) [84].
- The ability to fragment IP packets prior to encryption (RFC 4459) [83].

However we do not know what happens inside the VPC. We could not find any documentation of how it is implemented either, and hence cannot comment on its isolation techniques.

7 Comparison

In this section, a comparison between fifteen of the solutions (protocols and architectures) previously introduced in Section 5 will be presented. The solutions being compared are: 1. LISP, 2. NVGRE, 3. STT, 4. 802.1ad, 5. 802.1ah, 6. VXLAN 7. Diverter, 8. PortLand, 9. SEC2, 10. BlueShield, 11. VSITE, 12. NetLord, 13. VNT, 14. VL2, 15. DOVE. This comparison is made by using six criteria. The first criterion is the complexity of use of the solution. The second is the overhead induced by each solution. Then we compare the solutions' capability to migrate VMs, followed by a comparison of their resilience. The fifth criterion is scalability, and finally, we study if it is possible and easily manageable to have multiple data centers.

7.1 Complexity comparison

To determine the complexity of a technique we take into account six criteria:

1. Centralized/Distributed control plane
2. Network restrictions
3. Tunnel configuration and establishment

4. Tunnel management and maintenance
5. Multi-protocol
6. Security mechanism

Table 2 summarize this comparison.

7.1.1 Centralized/Distributed control plane

The first criterion is if the technique has a centralized control-plane or not. Among the presented solutions, eight of them have a centralized control-plane. These solutions are LISP, PortLand, SEC2, BlueShield, VSITE, NetLord, VL2 and DOVE. They all possess a key component which is used to resolve addresses. PortLand, SEC2, BlueShield, VSITE, VL2, and DOVE architectures use this centralized controller to also maintain the rules allowing tenant traffic isolation. However these architectures mitigate the consequences of a failure of this key component through replication. This replication increases complexity because those replicas must all possess the same information, which implies a synchronization of these components.

The other architectures do not possess such a key component and possess a distributed control-plane. The failure of a single device will not compromise the entire architecture. However they need to store redundant, and sometimes unused, information in every switch or VM to do the address resolution. Whereas with a centralized control plane approach the switches only get the information they need from the key component on an on-demand basis.

The centralized control plane design also has other issues. For example, with the PortLand architecture, in a data center with 27.648 end hosts (not tenants), and each host makes 25 ARP requests per second, the fabric manager will need approximatively 15 CPU cores working full-time just to manage the ARP requests. In addition, in PortLand, the fabric manager only manages ARP requests, whereas the other architectures, which also use a centralized controller, additionally manage rules and policies, which increases the workload of this component. To mitigate this increased workload, redundancy of the centralized controller and local caches in switches are used. There is a need for synchronization between all these components. To prevent data routed with outdated information from the local caches to attain a tenant network, SEC2, Blueshield, VSITE and VL2 use local modules (FEs, Echelon VMs, Hypervisors, and VL2 agents respectively), to enforce rules, and to drop packets that do not conform to these rules.

7.1.2 Network restrictions

Only three solutions, 802.1ad, 802.1ah, and VSITE, do not impose restrictions on the underlying network. Then there are architectures that impose a Layer level on the network such as SEC2 and BlueShield which need a Layer 2 network, or DOVE and LISP which need a Layer 3 network. Three architectures need a specific topology such as a Layer 2 multi-rooted fat tree topology for PortLand, a Layer 3 Clos topology for VL2 and a flat Layer 2 network for Diverter. Two protocols (NVGRE and VXLAN) require that the Layer 3 underlying network does not fragment packets and for VXLAN that there is an IGMP querier function enabled. For NetLord, the network must be a Layer 2 network but with edge switches supporting IP routing, which is why NetLord is put as a Layer 2-3 architecture. VNT is using the TRILL header so it needs the edge switch to support the TRILL protocol and a TRILL or Layer 2 core network. Finally STT is the one that is the most tricky, as it only needs a Layer 3 network but it uses a modified TCP header. STT needs to be allowed to transit in all the middle boxes of the network.

In this category we started with solutions that do not impose restrictions on the underlying network, continued with architectures needing a specific Layer level, and finished with architectures needing a very specific topology or protocol. In order to use these latter architectures, the underlying network might have to be heavily modified, which could be difficult or even impossible to do on already established infrastructures.

7.1.3 Tunnel configuration, establishment, management and maintenance

BlueShield, Diverter, and PortLand are exceptions because they do not use encapsulation protocols. Instead, to provide isolation, BlueShield uses a Directory server and ARP requests to verify whether a VM can communicate with a different VM. If communication is permitted in the Directory server rules, then the address resolution is possible and the directory server responds to the ARP request. If not then the directory server does not reply and the address resolution fails, therefore data is not sent between the VMs. For Diverter the VNET replaces the VM's MAC address with the server's MAC address, and verifies whether the communication is allowed.

Eight solutions (LISP, NVGRE, STT, VXLAN, SEC2, NetLord, VNT and VL2) among the other twelve use an implicit tunnel configuration and establishment for the core network part. They do not exchange messages or make reservation to establish the tunnel.

The others three use an explicit tunnel configuration. Those three solutions are the same three which does not impose restriction on the network. 802.1ah and 802.1ad both use the GARP VLAN Registration Protocol (GVRP), which is a GARP application, in order to distribute the VLAN registration information in the network. The last of the three, VSITE, uses the MPLS VPN protocol on the public network, which uses an explicit tunnel configuration.

Even if tunnel establishment could potentially be implicit, for twelve of the solutions, tunnel maintenance and management must still be explicit. LISP uses addresses mapping in its ITR (Ingress Tunnel Router) and ETR (Egress Tunnel Router). 802.1ad and 802.1ah both use join and leave messages sent by end stations

and bridges. VXLAN also uses join and leave messages. However they are sent by VTEPs with the goal of keeping the distribution tree of each VNI updated to reach all the clients of this VNI. Diverter, SEC2, BLueShield, VSITE, and VL2 each maintain rules for isolation which need to be managed and updated according to tenant requirement, or in case of VM migration. To enforce these rules, they all defined agent modules detailed in Section 7.2.3. PortLand uses soft states to maintain the tunnel and VNT uses temporary forwarding database in its RBridges. NetLord uses a SPAIN agent to manage the tunnel.

The two which do not have tunnel management are NVGRE and STT. For DOVE, tunnel management is dependent upon the encapsulation protocol.

7.1.4 Multi-protocol and security mechanism

Only PortLand and VNT are multi-protocol. NVGRE and BlueShield accept protocols from the second Layer because they use the MAC address to forward the packet at the last hop to the correct VM. STT, 802.1ad, 802.1ah, VXLAN, VSITE, and NetLord solutions also use the MAC address for delivering the packet but the protocol must be Ethernet. LISP, Diverter, SEC2 and VL2 use the IP address instead of the MAC to deliver packets, therefore the protocol must be IP.

About security mechanisms, these architectures do not define any encryption, authentication or integrity verification techniques. However five of them (Diverter, SEC2, BlueShield, VSITE, VL2) have security mechanisms for tenant isolation thanks to their agents and directory services which enforce pre-established isolation policies. In BlueShield, the Echelon VM (the agent) can also be associated with a firewall to improve security by processing the traffic only after it traverses the firewall.

7.2 Overhead comparison

To determine overhead we list the encapsulation headers, messages, and components used by each architecture. All these elements are summarized in Table 3.

7.2.1 Encapsulation

Among the fifteen solutions presented, one (BlueShield) does not use any encapsulation or address rewriting, two (Diverter and PortLand) do not use encapsulation either, instead rewriting the address of the packet. For Diverter, the MAC address of the virtual machine is replaced by the MAC address of the physical remote node. In PortLand, the Actual MAC (AMAC) is replaced by the Pseudo MAC (PMAC). The others use encapsulation.

LISP defines its own header of 64 bits and also uses both UDP and IP headers as outer headers. So when the packet enters a LISP tunnel, a header of 288 bits, for the IPv4 version, or a header of 448 bits, for the IPv6 version, is added to the packet.

NVGRE has its own header (64 bits) and then encapsulates the packet within an outer IPv4 header (160 bits) and an outer MAC Ethernet header (144 bits), making a total of 368 bits in IPv4 or 528 bits in IPv6.

STT encapsulates Ethernet messages with a STT header(144 bits) then with a TCP-Like header(192 bits), an outer IP header(IPV4: 160 bits, IPV6: 320 bits) and finally an outer Ethernet header (144 bits) for a total cost of 640(IPv4) or 800(IPv6) bits.

VXLAN also defines its own 64-bit header for encapsulation, and does not rewrite addresses.

802.1ad modifies the MAC header by adding an S-TAG of 32 bits, after both MAC addresses, followed by a C-TAG of 32 bits. After these modifications, the header size increased by 64 bits. The 802.1ah header is increased by a complete MAC header. As such we have a MAC destination (64 bits), a MAC source (64 bits), a B-TAG (32 bits), and an I-TAG (48 bits) for a total of 176 bits. However it is possible to use the 802.1 standard with 802.1Q frames or with 802.1ad frames. In the first case we have to add the 802.1Q header which is 32 bits long and so the new header is 208 bits long. In the second case, the new header is increased by the S-TAG and the C-TAG from the 802.1ad standard and is now 240 bits long.

DOVE does not specify an encapsulation protocol but proposes to use one of the three previously presented protocols (NVGRE, VXLAN, or STT). SEC2 and VSITE use MAC encapsulation with a header of $18 * 8 = 144$ bits. In addition VSITE changes the destination address with a locMAC address. However VSITE can be deployed over a Layer 3 network so instead of MAC encapsulation it can use IP encapsulation (160 bits in IPV4, 320 bits in IPV6) and replace the destination address with a locIP address. VL2 also uses an IP encapsulation and rewrites the destination address with a LA (Location Address). As VNT is deployed over a TRILL or a MLTP network, it uses a modified version of the TRILL header. This modified version is 192 bits long. NetLord uses both MAC and IP headers to encapsulate the data, which creates an overhead of 304 bits in IPv4 and 464 bits in IPv6. NetLord is the solution which has the largest header overhead.

7.2.2 Messages

Both NVGRE and STT do not exchange messages. This is explained by the fact that these two solutions are tunneling protocols and they only define an encapsulation technique. Then, BlueShield only uses lookup requests, from the BlueShield agent to the Directory Server, to resolve addresses and suppress ARP broadcasts. Diverter does not provide a specific type of message, and resolves addresses using the ARP protocol.

802.1ad and 802.1ah both use the Generic Attribute Registration Protocol. VXLAN requires its VTEP to manage the distribution tree of each VNI by sending join and leave messages for each VNI.

VNT is based over a TRILL or a MLTP network, which implies the use of TRILL messages. As the Control plane in TRILL and in MLTP is based on the IS-IS protocol, in order to route frames it uses SPF (Short Path First) tree topology generated by Link State PDU (LSP) messages.

DOVE, like all the other solutions with a centralized control plane (PortLand, SEC2, BlueShield, VSITE, NetLord, and VL2), must store the mapping between VMs' addresses and dSwitches' addresses in the DOVE Policy Service (DPS). All the servers of the DPS have to exchange information to be synchronized. DOVE dSwitches must also make unicast requests to retrieve the information from the DPS.

SEC2 needs communication between Forwarding Elements (FE) and the Central Controller (CC) to first save the mapping between the VM addresses and the FE addresses. Then the FEs have to intercept the ARP requests of the VMs and convert these to unicast lookup requests sent to the CC. However, it is possible that there are more than one CC which must possess the same information. This implies the need for synchronization messages between CCs. The other possibility is that every FE sends information to all CCs. Additionally, SEC2 uses VLAN, so it needs to use the Generic Attribute Registration Protocol.

NetLord is based on the Diverter model for resolving addresses, but it also needs unicast messages between SPAIN agents and the repository to obtain the table that maps destinations and sets of VLANs. In case of a topology change, new messages must be sent to update this table.

VL2 uses registration messages to store the association between AAs and LAs in the directory system. To resolve addresses the VL2 agents send lookup requests to the directory service. However there might be multiple directory services so they must be synchronized. In addition, for LA address assignment, VL2 uses an IP-based link state routing protocol.

VSITE uses an OTV-like protocol to exchange MAC addresses between CEc (Customer Edge cloud), CEt (Customer Edge tenant) and the directory server. This protocol allows the elimination of the cross-site MAC learning flooding. As an architecture with a centralized control plane, VSITE has to perform Directory lookups for address resolution. For these queries, unicast messages are sent from the CEc or VSITE agent to the Directory server. For address resolution between CEt and CEc they exchange MAC reachability information using OTV control plane protocol.

PortLand has four functionalities that need messages. The first is the registration of new source MAC address, as seen at the ingress switch, at the fabric manager to save the mapping between the PMAC, the MAC, and the IP addresses. To resolve addresses, PortLand intercepts the ARP requests of the VMs and converts them in unicast lookup requests to the fabric manager. However, if the fabric manager cannot answer one of the lookup requests, it must broadcast an ARP request to all end hosts. In addition, PortLand switches periodically send a Location Discovery Message (LDM) out of all their ports, both to identify their position, and to perform health checks. Finally, it is possible to have fabric manager synchronization messages in the case of redundant fabric managers.

LISP uses five different messages. The LISP Map-Request is used to request a mapping for a specific EID, to check the reachability of an RLOC, or to update a mapping before the expiration of the TTL. However in [2], it is RECOMMENDED that a Map-Request for the same EID-Prefix be sent no more than once per second. The second message is a LISP Map-Reply which is the answer to the LISP Map-Request. This message returns an EID-prefix whose length is at most equal to the EID-Prefix requested. The LISP Encapsulated Control Message(ECM) contains the control packets of the xTRs and also the mapping database system from [85]. Also defined in [85], the messages LISP Map-Register and LISP Map-Notify are used to manage the xTR/EID-Prefixes associations.

7.2.3 Components

Every solution presented in this survey carries at least one new networking dependency. Only five of them define exactly one component. LISP needs an xTR component. This component is the device at each end of the tunnel. It must function both as an Egress Tunnel Router (ETR) and as an Ingress Tunnel Router (ITR). Additionally the xTR needs to work as a Proxy ETR (PETR) and as a Proxy ITR (PITR) in order to connect LISP to non-LISP sites. Diverter introduces VNET, a software module which resides within the host OS on each physical node. Then NVGRE, VXLAN, and STT need NVGRE Endpoints, VXLAN Tunnel Endpoints (VTEP), and STT Endpoints. All three are modules in switches, servers, or hypervisors, which encapsulate and decapsulate the packets.

With two new components each, PortLand, VNT, VL2, and DOVE belong to the same group. Portland, VL2, and DOVE use a centralized controller. PortLand defines a Fabric Manager, VL2 a Directory System, and DOVE a DOVE Policy Service. To apply the rules stored in these centralized controllers, PortLand uses edge switches which must be able to perform MAC to PMAC header rewriting. VL2 defines an VL2 agent added to the hypervisor and DOVE defines dSwitches which are the edge switches of the DOVE overlay network. On the other hand, VNT does not use such a centralized controller. Instead VNT uses RBridges, which provide the advantages of Layer 2 (Bridges), Layer 3 (Routers), and Virtual Switches(VS). A VS is dedicated to host all interfaces tagged with a particular VNI corresponding to a tenant ID.

SEC2 and VSITE define three components each. SEC 2 uses a Central Controller (CC) but this device could possibly be on several servers for redundancy or load balancing. To enforce the rules of the

CC, Forwarding Elements (FEs) are introduced. A FE is a switch that intercepts ARP requests from VMs and encapsulates data if necessary. The third component is a web portal, where each customer can set up security policies for their network, which then translates them into policy settings saved in the CC. VSITE uses a Directory server to save the addresses associations. It presents a component called CEc (Customer Edge cloud) which is in the cloud data center. This CEc encapsulates the Ethernet frames received from the tenants' private networks with an IP header. The IP destination address is the address of the Top of the Rack switch which hosts the Ethernet frame's destination device. This device is in the tenant vstub. This CEc prevents the overlapping of VLAN IDs from multiple companies by translating this VLAN ID into a locally unique one. For an Ethernet frame from cloud VMs, the translation is done at the VSITE agent in the hypervisor. The Ethernet frame is then encapsulated with an outer IP header.

NetLord uses NetLord Agent (NLA) implemented at each physical server to encapsulate the data with an IP header and then with an Ethernet header. To do load balancing when sending packets, the NLA uses a SPAIN agent that is implemented in the NLA. The third component, an edge switch, is not really a new one. However, this edge switch must be able to read the IP header of the packet. And the last new component is a configuration repository which maintains all the configurations of the tenant virtual networks.

As the other architectures with a centralized control plane, BlueShield uses a Directory Server and an agent, called BlueShield Agent, to enforce the rules of the Directory server. For security measures, an Echelon VM is introduced and is in fact a VM that scans the traffic to apply added actions such as sending the traffic flow through a firewall. To suppress ARP flooding, a virtual switch is installed in the server and converts the ARP requests to unicast directory lookups. An additional component, 'ebtables' firewall, has been used to block all broadcast and multicast traffic.

Both 802.1ad and 802.1ah require that all the devices of the network adhere to their respective standard. These two solutions being IEEE standards, the devices are not modified by the network administrator but by the manufacturers of these devices in order to comply with the standard.

7.3 Migration of VM comparison

The migration of a VM is an important task in a virtualized data center. When a server needs to be shut down for maintenance, the VMs on this server must not be stopped so they have to be moved to another server. If a client's location changes, it might be interesting to move their VM accordingly. VM migration can be done in two different ways, an offline migration or a live migration. The offline migration will stop the service by terminating the session and establishing a new session once it has finished migrating. The one we are interested in here is the live migration which allows for a continuity of service and session even while the VM is being moved. We summarize this comparison in Tables 4 and 5.

LISP RFC [2] defines five types of mobility, however only three of them concern endpoint migration:

1. Slow endpoint Mobility: An endpoint migration without session continuity uses "RFC 4192" [86].
2. Fast Endpoint Mobility: An endpoint migration with session continuity.
3. LISP Mobile Node Mobility: An xTR migration.

Among these three types of mobility, only the last two are of interest to us for this comparison. For the Fast Endpoint Mobility, the solution is to use the technique of home and foreign agents. The home agent, the endpoint original agent, redirects traffic to the foreign agent of the network to which the endpoint moved. This technique is defined in "RFC 5944" [87] for IPv4 and in "RFC 6275" [88] and "RFC 4866" [89] for IPv6. However the last migration, the LISP mobile node mobility, allows the migration of device without the need of agents. As the device is itself an xTR, it can use topologically independent EID IP addresses. Thus it only has to register itself at the MAP-servers and Map-Resolvers of the network. This last solution is explained in [90].

NVGRE is an encapsulation and tunneling protocol. Its original goals were to increase the number of VLAN subnets; the VLAN technology being limited to 4096 subnets, and to achieve a multi-tenant environment. [91] states that NVGRE achieved its goals. However, NVGRE left the management of VM migration to IP because of its use of a UDP header. In order to improve the management of VM migration, the draft [92] defines new extensions for the control plane of NVGRE. Among these extensions, one is interesting for host migration. The REDIRECT message is in fact the original message, or at least the maximum data of the original message a REDIRECT message could contain, sent back to the sender. This REDIRECT message is sent by the old NVE, where the endpoint was hosted before migrating. The data of the returning packet starts with the address of the new NVE managing the endpoint. This address is 32 bits long and is the first information in the payload of the packet. Then follows a copy of as much data as possible of the original message. This way the sender now knows the address of the new NVE. However it is not specified how long the old NVE must maintain the information of the VM migration.

As NVGRE, STT is an encapsulation and tunneling protocol. However as opposed to NVGRE, there are no STT mechanisms for VM migration. STT is working with IP and it uses IP mechanisms to manage VM migration, but it must also use the IP mobility mechanism of STT.

802.1ad and 802.1ah manage VM migration the same way thanks to the GARP VLAN Registration Protocol (GVRP). When a VM moves, it must send a GVRP

message to the closest switch in order to indicate that the VLAN announced in the message is of interest for this machine. This way the VLAN tree will reach the VM. As the VM does not change its IP address, the connection is not lost. However, in order to keep the connection, the VLAN must be deployed in the destination device ahead of time in order to already have the distribution tree of this VLAN reach this device. Even with this advance deployment the migrating VM must stay in the same Layer 2 network.

VXLAN is a tunneling technique that allows a VM to migrate even to another network across a Layer 3 network. To do so, VXLAN uses join and leave messages destined to VTEP in order to indicate which distribution tree the VTEP must associate with. As for 802.1ad or 802.1ah, in order to have session continuity, the destination VTEP must be informed ahead of time that it must join the distribution tree requested by the migrating VM.

Diverter was designed to increase isolation between tenants' networks without degrading the overall performance of the network. A VM uses a virtual IP address which is created based on the Farm and Sub-network it belongs to. This IP address is formatted as follows:*10.Farm.Subnet.Host*. So in Diverter, all the VMs of one client belongs to the same sub-network and this sub-network is in one Farm. If a VM migrates to another server it means that this new server will now have to extend the Farm and the Sub-network. However, to discover the mapping between IP and MAC addresses, the VNET ARP engine uses multicast ARP, so the migration of the VM is not detected at the beginning of the migration, but only when the VNET ARP engine sends an ARP query, and the response has been received from the new server. If an existing connection was established between the VM pending migration and another VM, this connection will be interrupted at the beginning of the migration. The VNET of the non-migrating VM will continue to associate the MAC address of the old server, where the migrating VM was hosted, to the traffic of this session. This traffic will be lost until the VNET ARP cache entry times out and the VNET does an ARP query to retrieve the new MAC address. Nevertheless, since the IP address stays the same there could be no interruption of session even if, during the migration and until the VTEP learns the new MAC address, the traffic is lost. The session continuity depends on two parameters: The ARP cache entry timeout, and the TCP timeout. If the first one is longer than the second, then the session is lost and there is no live migration. On the other hand, if the TCP timeout is longer than the ARP cache entry timeout, the new MAC address will be retrieved before the end of the TCP session thus having session continuity and live migration.

PortLand defines Layer 2 messages to be sent when a VM migrates. Additionally, VMs' IP addresses remain unchanged during the migration. Thus, PortLand manages live migration as well as session continuity. These messages are sent only after the VM migration. The first message is a gratuitous ARP, sent by the migrated VM, which contains the new IP to MAC address mapping. It is forwarded to the fabric manager which then forwards an invalidation message intended to the old switch of the migrated VM. Upon reception of this message, the old switch sets up a flow table entry to trap the packets destined for the VM which has migrated. Additionally, when such packet is received at the old switch, it sends back a unicast gratuitous ARP to give the new PMAC address of the migrated VM. Optionally, to limit the loss of packets, the old switch can transmit the trapped packet to the VM.

As SEC2 architecture is composed of multiple edge domains, where the VM are hosted, and one core domain, which interconnect these edge domains, there are two ways a VM can migrate. First the VM stays in the edge domain. The migration consists of transferring a dynamic VM state from a source to destination hosts. Once the transfer is complete, a gratuitous ARP message is sent by the destination host to announce the VM's new location. This ARP message is sent only within the VLAN inside the edge domain, and can only reach hosts in this VLAN. However, if the VM migrates to a different edge domain, then the Central Controller (CC) has to update the VM's location in its table, including both eid and VLAN id. Since the IP address is not modified and the MAC address change is induced by the gratuitous ARP, then the migration is done without losing the session continuity and so SEC2 can perform live migration.

In [11], it is said that BlueShield allows live migration of protected VM. It is possible because BlueShield uses a Layer 2 core network and addressing scheme. As the IP address of the VM is untouched, the continuity of the session is preserved. However the process of migrating a VM is not clearly defined in the solution. We can guess that as each VM need to have a BlueShield agent which manages ARP queries, this same agent must warn the directory server of the migration of the VM and provide the new MAC address. This way the directory server informs the other VMs of the new address of the migrated VM. The echelon VM could manage the current traffic address replacement.

VSITE manages VM live migration thanks to the MAC learning mechanism and by using a location IP address, which is the IP of any Layer 3 switch that interconnects data center edge and core. As such, a VM migration can be considered "live" if it takes place inside one data center edge. This migration does not modify the IP address of the VM because the VM is still connected to the same Layer 3 switch and no routing updates are required. However, if the VM does migrate to another Layer 3 switch, then the location IP address is changed and the migration is no longer "live". Thus the directory service, both edge routers, and the server's hypervisor configuration must be updated.

When a VM starts or migrates in NetLord, the NetLord agent (NLA) in the hypervisor of the correspond-

ing server will have to broadcast a NLA-HERE message to report the location of the VM to the other NLAs. The NL-ARP table entries are permanent so only one message is sufficient to update the ARP-table. However if the broadcast is lost, the ARP-table does not have the correct information. Additionally, if packets for the migrated VM are already sent, then upon arrival of those packets, the server, which does not host the VM destination any more, has to reply with an unicast NLA-NOTHERE message. When receiving a NLA-NOTHERE message, the NLA will broadcast a NLA-WHERE message in order to retrieve the correct MAC address for the migrated VM. The IP address of the VM remains unchanged throughout the migration so the session remains uninterrupted. In this way, NetLord can do live VM migration.

VNT is based on the TRILL protocol which routes the messages thanks to Layer 2 nicknames. A VM is associated to a RBridge nickname in the core network. A message for a VM is modified by the ingress RBridge which routes the frame to the egress RBridge associated with the destination VM. When a VM migrates in TRILL the only modification is the association between an RBridge nickname and the VM, except if the VM remains in the domain managed by the RBridge. As such, the IP address and MAC address of a VM is not used for routing or forwarding purposes, so they remain unchanged during a VM migration thereby preserving the session continuity and realizing a live migration. Huawei, in [93] even qualify the migration of VM with TRILL as "Smooth VM migration".

VL2, like TRILL, uses an addressing scheme which separates the server address and addresses used for routing purposes. The server addresses are called application-specific addresses (AAs) and are not modified when a VM migrates. The modified address is the location-specific address (LA) which is the one used for routing the packets. Each AA address is associated with a LA address and it is the VL2 directory system which manages those associations. When a VM migrates and changes its AA/LA association, it is the directory system which must update the mapping and thus must inform the other VMs which want to communicate with the migrated VM. As during the migration process, if neither the IP nor the MAC address of the VM changes, then the session was uninterrupted and the migration was performed live.

DOVE works with tunneling protocols such as STT, VXLAN, and NVGRE. These protocols decouple the logic domain from the physical infrastructure while respecting machine mobility, address space isolation, and multitenancy. However, to handle VM migration and address resolution for these migrated VMs, a dSwitch must, upon detection of a newly-hosted VM, send a location update to the DPS. This allows the DPS to update its address resolution information.

7.4 Resilience comparison

In this section we look at techniques such as redundancy, multipath and backup that the solutions provide in order to manage failures.

LISP resilience is done through redundancy of its components. More than one CE (Customer Edge) router with LISP capabilities can be used which translates to more than one xTR with the the same IP address. Thus the RLOC becomes an anycast address and if one of the xTR fails then the traffic is automatically routed to the other with the same address. To manage these redundant xTRs, we have two arguments. First is priority; the higher the priority, the less favorable. Second is weight; if two xTR share the same priority then the traffic is divided according to their weight. For example, if xTR1 has a weight of 10 and xTR2 a weight of 5, then the traffic ratio will be 2:1 with xTR1 receiving the double of traffic than xTR2. Additionally, Mapping Server (MS) and Mapping Resolver (MR) are key components. To assure resilience, backup devices may be needed. When using multi-homed sites, with multiple xTR, it is no longer possible for the site to control its point of entry when the anycast route is advertised. The scope of advertisement is also reduced to /32 (/128 in IPv6) prefixes.

NVGRE is an encapsulating and tunneling protocol. There is no resilience in NVGRE because the sole function of NVGRE, encapsulation, is done by an important element, the hypervisor. In the event of failure of this element the packet could not reach the destination because it could not pass the hypervisor. It is possible to add resilience on the path by using multipath techniques such as ECMP (Equal-Cost Multipath) or [94] but this is not included in NVGRE.

STT's use of a TCP-like packet but lacking all the TCP functionalities results in the loss of IP datagrams in the event of congestion, or when a router on the path is not STT-enabled. In this case the router will drop the packet. In order to prevent such an undetected packet loss, the solution is to use a real TCP header in the outer IP header. As with NVGRE, it is possible to use ECMP in addition to STT for better path resilience. However there is a necessity that all packets of the same flow follow the same path and that all paths are used efficiently. STT endpoints are designed to be virtual switches running in software. These endpoints are mostly inside the servers and so each server is an endpoint. There is no need for endpoint redundancy since if the endpoint is down then the server is down, and so are the VMs.

The resilience in 802.1ad and 802.1ah can be obtained by aggregating links. If one link is down connectivity is maintained by the remaining backup links. In a similar way redundant switches bring resilience to the network. Therefor in 802.1ad and 802.1ah resilience is accomplished by network hardware redundancy.

Like NVGRE, VXLAN is a tunneling technique. VXLAN endpoints, the VTEP, are located within the

hypervisors of each server hosting VMs. VXLAN does not define resilience techniques. Hypervisor failure is managed through the use of redundant servers and the migration of VMs to those backup servers. However the session continuity might be interrupted. Session continuity could be maintained if a faulty hypervisor is detected prior to total failure thereby allowing a backup server to join this VNI permitting live VM migration. This solution is not defined in [7] and is only a possible solution to enhance VXLAN resilience.

Since Diverter uses a Layer 2 core network it is possible to use ECMP in order to increase resiliency of the path. Additionally, a Farm's virtual gateway is distributed in all the VMs of this Farm so there is no risk that the failure of the virtual gateway will block all communications with this Farm's VMs. However if a server is down then the Farm must be replicated on another server. As there is no live migration in Diverter, all the connections must be re-established.

PortLand's core network is based on a multi-rooted fat-tree topology, which increases link capacity at the tree summit, and uses the ECMP protocol. Additionally there is redundancy at the aggregation and core level switches. However the most important element in the PortLand solution is the fabric manager. If the fabric manager fails then address resolution is no longer possible. For this reason the fabric manager should be replicated. These backups however don't need to be exact replicas, since the fabric manager does not maintain a hard state.

Sec2 uses multiple FEs per site in order to increase reliability. Also the Central Controller (CC) can have a backup if needed. Additionally the CC can become a Distributed Controller (DC). As client networks are managed independently from each other, it is possible to have several controllers, each managing different client networks. However this solution increases the administration complexity and may result in having a backup for each small controller.

BlueShield has a centralized controller, the directory server, whose role is to resolve addresses and to enforce isolations rules. If this device fails then there will be no communication between VMs as they will be unable to obtain each other's MAC address. To prevent such a situation, the directory server is replicated over several devices. Also, to improve reliability, a BlueShield agent will send its queries to several directory server devices in order to have at least one answer. If a BlueShield agent, being located in each VM, fails then logically the VM itself will have failed. The same is true for the vSwitches and ebtables, as they are located in each server. BlueShield imposes a Layer 2 network but nothing else so it is possible to use ECMP in this Layer 2 network.

There is no specific technique for resilience defined in VSITE. However a server can be multi-homed to multiple top-of-rack switches. In this case there must be a master switch to handle the locIP. This master or slave configuration of the switches is done with the virtual router redundancy protocol [95]. As VSITE is a solution with a centralized controller (the Directory server) it might be necessary to replicate this controller.

In order to benefit from a high-bandwidth resilient multipath fabric using Ethernet switches, NetLord relies on SPAIN [62]. Like the other solutions using a centralized controller, it might be necessary to have redundant configuration repositories, not only for availability but also for improvement in performance. The NetLord's agents are all located inside the hypervisors of each physical server, so for NLA redundancy we must have server redundancy.

VNT, being a distributed solution, has no need for redundant centralized controller. Additionally, VNT uses ECMP for multiple paths, so even if an RBridge fails, the traffic is sent to another RBridge. Each VNI (a.k.a tenant) can have its own multicast distribution tree and it is possible to configure a backup tree if needed.

The Clos topology, used by VL2, provides a simple and resilient topology. Routing in such a topology is done by taking a random path up to a random intermediate switch and then taking another random path down to a destination ToR switch. However VL2 has a centralized controller (the directory server) which must be replicated. Otherwise, in case of failure, address resolution would be impossible.

The use of tunneling protocols in DOVE provides multipath capabilities and routing resiliency. The key component of DOVE, the DOVE Policy Service, must be resilient to ensure high availability. It maintains the information of the network in order to resolve dSwitch policy requests. DPS should additionally be replicated and have multiple backups.

7.5 Scalability comparison

In virtualized data center and virtualized environment in general, the goal is to share the infrastructure among the maximum number of clients, thus scalability is an important criteria.

As mentioned in [96], the separation of the Endpoint Identifiers (EIDs) and Routing Locators (RLOCs) in LISP, allows for a better scalability through a greater aggregation of RLOCs. However new limits are imposed, notably one RLOC address having $2^{32} = 4294967296$ possible EIDs in IPv4 and $2^{128} \approx 3,4*10^{38}$ in IPv6. These limits are also applicable for the number of possible RLOCs, so we can see that the address space is scalable. However the MS and MR are hardware components with memory and CPU limitations. MS and MR must store mapping information between RLOCs and EIDs, but seeing as there are, for one RLOC with an IPv4 addressing, approximately 4 billion EID addresses, we can conclude that the scalability issue lies within the MS and MR components. For example in [97] the maximum number of NAT translations stored is 2147483647 which is only half of the number of EIDs in IPv4. This limitation is based on

a theoretical maximum within the Cisco IOS XE operating system, and not upon any physical hardware limitation.

The use of NVGRE endpoints allows the representation of multiple Customer Address (CA) by only one Provider Address (PA). This way the core network routers have fewer addresses to store and manage. Also the sizes of MAC address tables at the Top of Rack switches are reduced. It is also possible to increase scalability by implementing proxy ARP at each NVGRE endpoints to prevent most broadcast traffic and convert the rest to multicast traffic which reduces the load on the control plane. To prevent most broadcast traffic, NVGRE endpoints must be placed within the hypervisor. The VSID field, in the NVGRE header, is 24 bits long so there are $2^{24} = 16777216$ virtual Layer 2 networks.

In STT the core network only knows the IP addresses of each virtual switch, one virtual switch per server at most. In the worst case scenario, there is only one VM per server so there is the same number of virtual switches as VMs and since it uses an IP based address scheme, it has a similar scalability as IP including the ability to aggregate. However, usually a server hosts more than one VM so we have greater scalability. Additionally, with a Context ID of 64 bits it is possible to have $2^{64} \approx 1.8{*}10^{19}$ IDs. Consequently, scalability limitations reside in the virtual switches, which are unable to manage so many IDs. Another limit is the number of VMs per server. This last limit can be mitigated if we change the STT endpoint location. By using a dedicated device in front of several servers, the number of VMs managed by this device will be higher than if the STT endpoint was in the server itself. However, this implies additional network hardware.

Both 802.1ad and 802.1ah standards are evolutions of the 802.1Q standard with the VLAN solution. As such, they both increase the limit of VLAN from 4096 to $2^{12} * 2^{12} = 16777216$ for 802.1ad and to $2^{12} * 2^{20} = 4294967296$ for 802.1ah, and with the optional fields to $2^{12} * 2^{20} * 2^{12} * 2^{12} \approx 7 * 10^{16}$. The issue here lies with the switches, which have to manage this number of VLANs. A switch is incapable of managing so many VLANs, so in order to reduce this number, join and leave messages ensure that the switch manages only the VLANs needed by the endpoints.

VXLAN uses a VXLAN Network Identifier (VNI) which is 24 bits long so there are $2^{24} = 16777216$ VNIs. However VXLAN works at the software level, which impacts overall performance because hardware offload is not possible. Whether or not this is important, draft [98] attempted to address this question. They observed increased CPU and unstable throughput 5.6 Gb across a 10Gb network. However those results must be taken with a grain of salt,because, as stated in the paper, the tests were realized using only one server when the design and purpose of VXLAN is for a multi-server environment.

With its virtual IP addressing scheme, Diverter manages up to 16 million VMs system-wide. However the solution for IP addressing is more restrictive with regards to the number of tenants. With an IP of the type *10.F.S.H* we have 255 distinct farms, with 255 subnets in each farm. If we consider that each client uses one Sub-network then we have a maximum of $255 \times 255 = 65025$ clients in total. This limit can be modified, as stated in [8] since this address scheme is by default and may be modified prior to network deployment. With this in mind, we are faced with another scalability issue; to determine in advance how many farms, subnets, and hosts would be required. On the other hand, this addressing technique allows the core switches to only see one MAC address per server, which reduces the size of the MAC forwarding table.

As discussed in Section 5.2.1.4, PortLand is a solution with a centralized control plane. The fabric manager, a single machine, must manage all ARP traffic for the whole network, rendering this architecture unscalable in a large data center. For example in a data center with 27.648 end hosts (not tenants), each making 25 ARP requests per second, the fabric manager would need approximatively 15 CPU cores working full time just to handle the ARP requests. Additionally there is no notion of tenant isolation in the solution. In order to provide isolation, rules could be enforced by the fabric manager. For example, the fabric manager will only respond to ARP queries when allowed by policies. However, doing so increases overhead on the fabric manager, thereby further decreasing the number of end hosts that can be managed. A possible solution would be to have additional fabric manager for fewer end hosts thereby reducing the load on the fabric manager, but in [9] this solution is preceded by "it should be possible", so additional fabric manager configuration may be necessary.

Like PortLand, SEC2 is a solution using a centralized control plane. The Centralized Controller is the key element for addresses resolution, rules and isolation enforcement. The results of the CC can be extrapolated from PortLand fabric manager results. In fact those results might be "worse" seeing that the CC has more actions to do when processing an ARP request than the fabric manager. In order to reduce the load, the SEC2 CC can become a Distributed Controller with each device managing some client networks. This way we can increase the number of edge domains in the data center. Another limitation is the number of client networks in each edge domain. As the tenant isolation in edge domains is done thanks to VLANs, the number of 4096 tenants is the limit. However, the number of tenants within the DC is limited by the number of domain edges multiplied by the number of VLANs per edge domain. The limit of edge domains is the maximum number of MAC addresses. As long as there are free MAC addresses we can add edge domains. One Edge domain is associated with all its FEs' MAC address.

BlueShield improves its scalability by suppressing all

VMs' ARP broadcast and converting them to unicast directory server lookups. However the PortLand experience can be used as reference for this solution. We saw that for $\approx$ 27000 end hosts each sending 25 ARP requests per second, the centralized controller will need approximatively 15 CPU cores working non-stop to manage these requests. In order to overcome this limitation, BlueShield uses redundant directory servers to share the load. Nevertheless, contrary to SEC2, in BlueShield each directory server must have the same information. Even if we increase the number of directory servers in order to alleviate the CPU load, we will have another limitation imposed by the physical memory of the device. Additionally, the directory server must not only save the ARP information but also the rules indicating which VMs can exchange data. As a consequence of this quantity of information the DS must look through, the latency is increased.

Using locIP based on the Layer 3 switch virtual IP, VSITE can aggregate multiple VMs under one IP address. The VMs MAC addresses are only known inside the data center edges. This allows for smaller table size in core network routers as they only learn the locIP addresses. However, like the other solutions using a centralized control plane, one scalability limit is given by the capacities of the directory server which must store both IP addresses, the locIP and the real IP, MAC addresses, VLANs, and must resolve address queries. Additionally, VSITE uses VLANs for client isolation and therefore imposes a limit of 4096 virtual networks.

Concerning scalability, NetLord uses an IP encoding which gives 24 bits for the Tenant_ID value. With 24 bits it is possible to have $2^{24} = 16777216$ simultaneous tenants. The encapsulation scheme prevents Layer 2 switches to see and save all Layer 2 addresses. These Layer 2 switches see the local Layer 2 addresses and the addresses of all the edge switches. The authors of [13] estimate that NetLord can support: $N = V \times R \times \sqrt{(\frac{F}{2})}$ virtual machines. Where V is the number of VMs per physical server, R is the switch radix, and F is the MAC forwarding information base (FIB) size (in entries). In Table 1, they presented results for V = 50. Additionally NetLord use multipath technology based on SPAIN and so achieves a throughput similar to that of machine-to-machine communication.

Table 1: NetLord worst-case limits on unique MAC addresses (From [13])

Switch Radix	FIB Sizes			
	16K	32K	64K	128K
24	108,600	153,600	217,200	307,200
48	217,200	307,200	434,400	614,400
72	325,800	460,800	651,2600	921,600
94	425,350	601,600	850,700	1,203,200
120	543,000	768,000	1,086,000	1,536,000
144	651,600	921,600	1,303,200	1,843,200

The VNT solution based on TRILL has the same scalability advantages as TRILL. The core RBridges only learn the nicknames of the other RBridges. An edge RBridge aggregate multiple VMs MAC addresses under its nickname. So RBridge forwarding database sizes are reduced compared to classical Ethernet forwarding databases. [69] states :

> "... unicast forwarding tables of transit RBridges to be sized with the number of RBridges rather than the total number of end nodes ..."

It is also true if VNT is used with MLTP. Additionally, VNT introduces a VNI TAG to separate the virtual networks. This TAG is 24 bits long so it can accommodate $2^{24} = 16777216$ virtual networks which should be sufficient for the next few years.

The scalability of the VL2 solution is limited by the capacity of the directory server. In order to increase scalability, VL2 uses additional directory servers. These additional directory servers improve the maximum lookup rate. Some experimental results are given in [15]. In those experiments, the goal was to process the most lookup requests possible, while ensuring sub-10ms latency for 99% of the requests. They found that a directory server can manage 17000 lookups/sec and that the lookup rates increase linearly with the increase of servers. In the worst case scenario, chosen in [15], 100000 servers simultaneously performing 10 lookup requests requires 60 servers in the directory system. We can conclude that the scalability limitation of VL2 comes from its directory system.

DOVE's scalability is achieved thanks to the tunneling protocol it uses. As such the choice of the tunneling protocol is bound by several attributes. Among them are the interoperability and scalability attributes. Those attributes define that the protocol must used genuine headers for delivery and it must adapt to different underlays. However the scalability issue is located in the DPS. As DOVE is a solution with a centralized control plane, we have the same issue of having the centralized controller being overloaded by the amount of policy requests. So the DPS must be scalable but the means to achieve this are not specified in the article.

7.6 Multi data center comparison

Multi data center interconnection is interesting since there often may be multiple physical facilities for a given virtual data center. For this reason we will identify if the proposed solutions have inherent multi data center capabilities.

LISP is adapted for multiple data centers as long as each data center is a LISP site with at least one xTR and a RLOC address. In this case then all devices in the data center have EID addresses that are associated to the RLOC address of the xTR of the data center. In fact, [99] examines the best possible deployment of LISP in a data center and section 5 discusses data center interconnection over a wan network.

However, if some data centers are not LISP-enabled then we need to refer to RFC 6832 [100], which describes how an interconnection between a LISP site and a non LISP site is possible and implemented. This standard introduced three such mechanisms. One uses a new network element, a LISP Proxy Ingress Tunnel Router (Proxy-ITR), installed at non LISP site. Another mechanism adds another layer of Network Address Translation (NAT) at xTR. And the last also uses a new network element, a Proxy Egress Tunnel Router (Proxy-ETR).

NVGRE can be used like a site-to-site VPN. To do so each site needs a VPN gateway which supports NVGRE. These gateways will then establish a tunnel between them and encapsulate and respectively decapsulate the sent and received packets.

As mentioned in [4], "STT deployments are almost entirely limited at present to intra-data center environments". This is explained by the fact that STT uses a TCP-like header that has the same fields as a TCP header but not the same functionalities. As such, the middle boxes which do not have STT knowledge will drop the packets. That is why, for now, STT is only used in environments where the same administrative entity can manage all the middle boxes to process STT packets. So for now, even if theoretically STT can be used like a site-to-site VPN, it is not practically feasible.

802.1ad and 802.1ah can interconnect data centers if the network between them is a Layer 2 network. Most of the time however, it is a Layer 3 network thus the frames need to be encapsulated in IP. All the switches in the data center and in the Layer 2 network must respect the 802.1ad or 802.1ah standards. As both are standards, all the recent switches from manufacturers support them.

VXLAN is designed mostly for intra data center communication, however it is possible to use it like a site-to-site VPN with VXLAN gateways at each site. Additionally, [101] proposes the use of Ethernet VPN (E-VPN) technology to interconnect VXLAN sites. It could also be used to interconnect NVGRE sites. However this solution imposes the use of IP/MPLS networks between the sites.

Diverter does not specify any inter data center communication techniques. Nevertheless, each farm hosts multiple subnets, each with multiple hosts, and we can extrapolate in saying that a farm could represent a data center. This way we see that to have multiple data center interconnected with Diverter the only requirement would be to have a Layer 2 connection between those data centers. However, doing so would result in poor scalability. Additionally all the control traffic would have to reach all the VNETs from all data center which might be a costly use of the interconnection links.

PortLand is based on a fat tree topology which is a data center topology. So in order to use PortLand for inter data center communication it is mandatory to have a Layer 2 fat tree topology between the data centers. If the interconnection of each data center core switches via a Layer 2 fat-tree topology is achievable then we could achieve a large-scale PortLand network spanning multiple data centers. This being said, multi data center connectivity is not discussed in the solution brief.

By design, SEC2 is already multi-domain. We have a core domain which interconnects several edge domains. We could see the edge domains as data centers and the core domain as a Layer 2 interconnection between the data centers. Additionally we need a centralized controller reachable by all forwarding elements (FE) in all data centers. If we use a distributed controller then each member device must also be reachable by the FEs. The only issue is scalability. As tenant isolation in edge domains is done via VLANs, it means that in each data center we will have at most 4096 VLANs which is insufficient for virtualized data centers.

The BlueShield solution is based upon preventing address resolution by blocking ARP queries and converting them to unicast directory server lookups via the vSwitch. There is no notion of multi data center in the paper, however we can imagine a simple solution with a directory server replicated in each data center which manages all the rules for inter data center communication throughout the network.

By design, VSITE interconnects multiple client sites to a data center. So in the same manner we can also interconnect data centers. In order to manage this, it is necessary to increase the directory server capabilities to match the increase in information it stores. This directory server will have to store the information concerning all VMs in the network. Also each data center will need at least one cloud data center in order to implement OTV-like protocol, which exchanges MAC reachability information with other cloud data centers and the directory server.

NetLord does not address the multi data center issue. However it is possible to interconnect multiple data centers and as a result implement a larger network. All these data centers will share the same control plane, meaning that all control messages will travel across the public interconnection to reach all the data centers. This implies that the configuration repository will have to store the information for all data centers, which presents a potential scalability problem. Additionally, this interconnection will have to be done with a Layer 2 message transporting tunnel.

The TRILL protocol is multi-data-center-ready, and thereby also is VNT. However, to manage this multi data center network, the solution used by TRILL is to have one big network with one control plane shared among the data centers. This is not scalable seeing that there are only 16 bits for a nickname, 65536 nicknames in total, and that they all must be unique. This also means that the interconnection of those TRILL data centers must be done using site-to-site tunnels. However, when using the MLTP/VNT solution, the

merging issue does not exist anymore as each data center control plane remains independent. Additionally, MLTP introduce a new nickname management which increase the number of available nicknames to more than one billion. Nevertheless, even when using MLTP, the interconnection of MLTP data centers must be done using site-to-site tunnels.

The multi data center issue is not discussed in VL2. It might however be possible to interconnect multiple data centers. The directory system could be externalized in order to manage the whole network, which could span multiple data centers.

Like VL2, DOVE does not address the multi data centers issue. Nevertheless it might be possible to have the DOVE solution span over multiple data centers. However, this means that the DPS will have to manage the request of even more dSwitches from all the data centers. This way the control plane is shared among all the data centers and the DPS redundancy is the new scalability limit.

8 Discussion

Tunneling solutions based on a centralized controller need to tackle the scalability issue. Current solutions with a centralized control plane need a centralized controller with substantial processing power or such devices are costly. A solution could be to use multiple devices aggregated to form a centralized controller in order to share the load. However, those devices have to be synchronized. Another solution could be to have smaller interconnected data center. However the interconnection between data centers is not really addressed. Even if some architectures are multi-site by design, those have scalability issues inside each site which prevents the site from being a data center. For those architectures, the solution might be found in using another tunneling protocol enabling a better scalability within a site.

The next possible extension of cloud computing is the hybrid cloud. Gartner [102] expects that hybrid cloud will be adopted by almost half of the largest companies by the end of 2017. In this type of cloud, tenants' traffic will be indistinctly crossing the public network between the data center network and a tenant's private network. And as in public cloud, tenants would want their traffic to be isolated from other tenants and other entities in general. The presented solutions improve tenant data security thanks to traffic isolation achieved by respecting rules and forwarding tables. However, isolation is only a part of security. Security is an area to improve as isolation is not sufficient to guaranty integrity and prevent the theft of the data. For example, if a corrupt or faulty component does not respect these rules then tenants' traffic isolation is compromised. Another case could be that a malicious user or even a data center employee might illegally access a central component. It could allow them to arbitrarily implement any rules they desire and thereby override isolation rules, even if network components correctly adhere to them. An attacker could also realize a man-in-the-middle attack or intercept the traffic, or as data centers are now more and more virtualized, with VMs migrating across these data centers to improve performance, or in case of necessity, there are more and more tenants and their data is passing through more and more devices increasing the risk for the data. Additionally, now that hybrid cloud is growing, work must be done in order to isolate traffic from end-to-end between the tenant's private network and the cloud's infrastructure.

Any solution must also take into account that the security requirements of one tenant may be considerably different to that of another. This demonstrates a need to manage several security mechanisms and policies, which increases the complexity of the network. In addition, there are potential conflicts between intrusion detection systems policies, belonging to service or infrastructure providers, and firewalls, which need to be resolved [103].

Another topic to tackle is the fact that Layer 2 solutions mostly use Spanning Tree (STP), Rapid Spanning Tree (RSTP), or Multiple Spanning Tree (MSTP) to prevent loops, thus rendering unusable a number of links and reducing the overall performance of the data center. To prevent that, a solution could be to use level 2 multipath technology. TRILL possesses such functionality and other Layer 2 solutions could use Shortest Path Bridging (SPB) specified in the IEEE 802.1aq standard.

Another area of improvement for Layer 2 solution is CPU offloading. Some Layer 2 solutions add a new header which is not yet recognized by Network Interface Cards (NICs) thus the processing of the packet is done by the CPU which consumes additional resources and decreases overall performance. A practical solution could be to program the offloading of these new headers in NICs or to distribute traffic across multiple CPUs.

9 Conclusion

Data centers are being more frequently used. Especially cloud data centers where workloads, representing 39% of total data center workloads, will continue to grow, up to 63% of the total data center workloads by 2017. The cloud data center advantage is that it hosts multiple tenants to increase infrastructure efficiency and reduce costs. However some issues arose like tenants' traffic isolation.

In this paper, we surveyed fifteen solutions that provide tenant traffic isolation in a cloud network. We first presented them and then compared their complexity, the overhead they induce, their abilities to manage VMs migration, their resilience, their scalability, and their multi data center capabilities. Each solution provides tenant traffic isolation by using varying approaches, however these solutions are not all multi-data-center-ready, and those that are have potential

issues with scalability. Nevertheless, VNT solution based on TRILL derives multi data center capability from the work already done on trill, implementing control plane isolation in each data center and an interconnection network control plane, thereby increasing scalability [104, 105, 106, 72].

Finally we identified some research areas which are not yet thoroughly discussed in these papers, and are areas for possible future research. Tenant traffic is not safe enough by just isolating it. It may be necessary to implement other security mechanisms in order to provide better security.

Data centers are increasingly being virtualized, with VMs migrating across these data centers to improve performance or in case of necessity. However the interconnection between data centers is not really addressed. When a multi data center technique is presented, there is a trade off in the scalability of the solution.

Additionally, now that hybrid clouds are growing, work must be done in order to isolate traffic from end to end between a tenant's private network and the cloud.

References

[1] Cisco global cloud index: Forecast and methodology, 2013-2018. Technical report, Cisco Systems, Inc, 2014. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html.

[2] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. The locator/id separation protocol (lisp). RFC 6830, January 2013.

[3] Murari Sridharan, Yu-Shun Wang, Albert Greenberg, Pankaj Garg, Narasimhan Venkataramiah, Kenneth Duda, Ilango Ganga, Geng Lin, Mark Pearson, Patricia Thaler, and Chait Tumuluri. Nvgre: Network virtualization using generic routing encapsulation. Work in progress, draft-sridharan-virtualization-nvgre-04, February 2014.

[4] Bruce Davie and Jesse Gross. A stateless transport tunneling protocol for network virtualization (stt). Work in progress, draft-davie-stt-06, April 2014.

[5] Institute of Electrical and Electronics Engineers. Ieee 802.1ad-2005. 802.1ad - Virtual Bridged Local Area Networks, 2005.

[6] Institute of Electrical and Electronics Engineers. Ieee 802.1ah-2008. 802.1ah - Provider Backbone Bridges, 2008.

[7] Mallik Mahalingam, Dinesh G. Dutt, Kenneth Duda, Puneet Agarwal, Lawrence Kreeger, T. Sridhar, Mike Bursell, and Chris Wright. Vxlan: A framework for overlaying virtualized layer 2 networks over layer 3 networks. Work in progress, draft-mahalingam-dutt-dcops-vxlan-09, April 2014.

[8] Aled Edwards, Anna Fischer, and Antonio Lain. Diverter: A new approach to networking within virtualized infrastructures. In *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, WREN '09, pages 103–110, New York, NY, USA, 2009. ACM. http://doi.acm.org/10.1145/1592681.1592698.

[9] Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. Portland: A scalable fault-tolerant layer 2 data center network fabric. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pages 39–50, New York, NY, USA, 2009. ACM. http://doi.acm.org/10.1145/1592568.1592575.

[10] Fang Hao, T. V. Lakshman, Sarit Mukherjee, and Haoyu Song. Secure cloud computing with a virtualized network infrastructure. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 16–16, Berkeley, CA, USA, 2010. USENIX Association. http://dl.acm.org/citation.cfm?id=1863103.1863119.

[11] Saurabh Barjatiya and Prasad Saripalli. Blueshield: A layer 2 appliance for enhanced isolation and security hardening among multi-tenant cloud workloads. In *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*, UCC '12, pages 195–198, Washington, DC, USA, 2012. IEEE Computer Society. http://dx.doi.org/10.1109/UCC.2012.21.

[12] Li Li and Thomas Woo. Vsite: A scalable and secure architecture for seamless l2 enterprise extension in the cloud. In *Secure Network Protocols (NPSec), 2010 6th IEEE Workshop on*, pages 31–36. IEEE, 2010.

[13] Jayaram Mudigonda, Praveen Yalagandula, Jeff Mogul, Bryan Stiekes, and Yanick Pouffary. Netlord: A scalable multi-tenant network architecture for virtualized datacenters. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pages 62–73, New York, NY, USA, 2011. ACM. http://doi.acm.org/10.1145/2018436.2018444.

[14] Ahmed Amamou, Kamel Haddadou, and Guy Pujolle. A trill-based multi-tenant data center network. *Computer Networks*, 68(0):35 – 53, 2014. Communications and Networking in the Cloud http://www.sciencedirect.com/science/article/pii/S1389128614000851.

[15] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: A scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pages 51–62, New York, NY, USA, 2009. ACM. http://doi.acm.org/10.1145/1592568.1592576.

[16] Liane Lewin-Eytan, Katherine Barabash, Rami Cohen, Vinit Jain, and Anna Levin. Designing modular overlay solutions for network virtualization. Technical report, IBM, 2011.

[17] R. Cohen, K. Barabash, V. Jain, R. Recio, and B. Rochwerger. Dove: Distributed overlay virtual network architecture, 2012.

[18] Rouven Krebs, Christof Momm, and Samuel Kounev. Architectural concerns in multi-tenant saas applications. In *CLOSER*, pages 426–431, 2012.

[19] Cisco virtualized multi-tenant data center, version 2.0 compact pod design guide. Technical report, Cisco Systems, Inc, 2010. http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/VMDC/2-0/design_guide/vmdcDesignGuideCompactPoD20.pdf.

[20] Cisco virtualized multi-tenant data center, version 2.2 design guide. Technical report, Cisco Systems, Inc, 2012. http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/VMDC/2-2/design_guide/vmdcDesign22.pdf.

[21] Securing multi-tenancy and cloud computing. Technical report, Juniper Networks, Inc, 2012. https://www.juniper.net/us/en/local/pdf/whitepapers/2000381-en.pdf.

[22] Steve Bobrowski. The force.com multitenant architecture. Technical report, salesforce.com, inc, 2013. http://s3.amazonaws.com/dfc-wiki/en/images/8/8b/Forcedotcom-multitenant-architecture-wp-2012-12.pdf.

[23] Virtual network overview. https://msdn.microsoft.com/en-us/library/azure/jj156007.aspx.

[24] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862 – 876, 2010.

[25] A. Fischer, J.F. Botero, M. Till Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *Communications Surveys Tutorials, IEEE*, 15(4):1888–1906, Fourth 2013.

[26] Tunneling - cisco. http://www.cisco.com/c/en/us/products/ios-nx-os-software/tunneling/index.html.

[27] What is a tunneling protocol? http://usa.kaspersky.com/internet-security-center/definitions/tunneling-protocol.

[28] Vpn tunneling protocols. https://technet.microsoft.com/en-us/library/dd469817%28v=ws.10%29.aspx.

[29] Li heng, Yang dan, and Zhang xiaohong. Survey on multi-tenant data architecture for saas. *IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 6, No 3, November 2012*, 2012.

[30] Stefan Aulbach, Torsten Grust, Dean Jacobs, Alfons Kemper, and Jan Rittinger. Multi-tenant databases for software as a service: Schema-mapping techniques. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1195–1206, New York, NY, USA, 2008. ACM.

[31] Vivek Narasayya, Sudipto Das, Manoj Syamala, Badrish Chandramouli, and Surajit Chaudhuri. Sqlvm: Performance isolation in multi-tenant relational database-as-a-service. In *6th Biennial Conference on Innovative Data Systems Research (CIDR '13)*, 2013.

[32] Ying Hua Zhou, Qi Rong Wang, Zhi Hu Wang, and Ning Wang. Db2mmt: A massive multi-tenant database platform for cloud computing. In *e-Business Engineering (ICEBE), 2011 IEEE 8th International Conference on*, pages 335–340, Oct 2011.

[33] Xuequan Zhou, Dechen Zhan, Lanshun Nie, Fanchao Meng, and Xiaofei Xu. Suitable database development framework for business component migration in saas multi-tenant model. In *Service Sciences (ICSS), 2013 International Conference on*, pages 90–95, April 2013.

[34] Wang Xue, Li Qingzhong, and Kong Lanju. Multiple sparse tables based on pivot table for multi-tenant data storage in saas. In *Information and Automation (ICIA), 2011 IEEE International Conference on*, pages 634–637, June 2011.

[35] Ahmed Amamou. *Network isolation in a virtualized datacenter*. PhD thesis, University Pierre and Marie Curie - Paris 6 - EDITE of Paris, 2013. French thesis, Isolation reseau dans un datacenter virtualise.

[36] Jinho Hwang, Sai Zeng, F.Y. Wu, and T. Wood. A component-based performance comparison of four hypervisors. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 269–276, May 2013.

[37] Hasan Fayyad-Kazan, Luc Perneel, and Martin Timmerman. Benchmarking the performance of microsoft hyper-v server, vmware esxi and xen hypervisors. *Journal of Emerging Trends in Computing and Information Sciences*, 4(12), 2013.

[38] Todd Deshane, Zachary Shepherd, J Matthews, Muli Ben-Yehuda, Amit Shah, and Balaji Rao. Quantitative comparison of xen and kvm. *Xen Summit, Boston, MA, USA*, pages 1–2, 2008.

[39] Wei Jing, Nan Guan, and Wang Yi. Performance isolation for real-time systems with xen hypervisor on multi-cores. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2014 IEEE 20th International Conference on*, pages 1–7, Aug 2014.

[40] Stan Hanks, Tony Li, Dino Farinacci, and Paul Traina. Generic routing encapsulation (gre). RFC 1701, October 1994.

[41] Dino Farinacci, Tony Li, Stan Hanks, David Meyer, and Paul Traina. Generic routing encapsulation (gre). RFC 2784, March 2000.

[42] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn. Point-to-point tunneling protocol (pptp). RFC 2637, July 1999.

[43] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter. Layer two tunneling protocol "l2tp". RFC 2661, August 1999.

[44] J. Lau, M. Townsley, and I. Goyret. Layer two tunneling protocol - version 3 (l2tpv3). RFC 3931, March 2005.

[45] S. Bryant and P. Pate. Pseudo wire emulation edge-to-edge (pwe3) architecture. RFC 3985, March 2005.

[46] Institute of Electrical and Electronics Engineers. Ieee 802.1q-2005. 802.1q - Virtual Bridged Local Area Networks, 2005.

[47] Institute of Electrical and Electronics Engineers. Ieee 802.1d-1990. 1990.

[48] Ieee.org, 802.1ad - provider bridges. `http://www.ieee802.org/1/pages/802.1ad.html`.

[49] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. RFC 3031, January 2001.

[50] Bob Braden, Lixia Zhang, Steve Berson, Shai Herzog, and Sugih Jamin. Resource reservation protocol (rsvp) – version 1 functional specification. RFC 2205, September 1997.

[51] Loa Andersson, Ross Callon, Ram Dantu, Paul Doolan, Nancy Feldman, Andre Fredette, Eric Gray, Juha Heinanen, Bilel Jamoussi, Timothy E. Kilty, and Andrew G. Malis. Constraint-based lsp setup using ldp. RFC 3212, January 2002.

[52] Kathleen Nichols, Steven Blake, Fred Baker, and David L. Black. Definition of the differentiated services field (ds field) in the ipv4 and ipv6 headers. RFC 2474, December 1998.

[53] L. Berger. Generalized multi-protocol label switching (gmpls) signaling functional description. RFC 3471, January 2003.

[54] E. Mannie. Generalized multi-protocol label switching (gmpls) architecture. RFC 3945, October 2004.

[55] E. Rosen and Y. Rekhter. Bgp/mpls ip virtual private networks (vpns). RFC 4364, February 2006.

[56] Jon Brodkin. Vmware users pack a dozen vms on each server, despite memory constraints. `http://www.networkworld.com/article/2197837/virtualization/vmware-users-pack-a-dozen-vms-on-each-server--despite-memory-constraints.html`.

[57] Lori MacVittie. Virtual machine density as the new measure of it efficiency. `https://devcentral.f5.com/articles/virtual-machine-density-as-the-new-measure-of-it-efficiency`.

[58] Determine true total cost of ownership. `http://www.vmware.com/why-choose-vmware/total-cost/virtual-machine-density.html`.

[59] Rich Miller. A look inside amazon's data centers. `http://www.datacenterknowledge.com/archives/2011/06/09/a-look-inside-amazons-data-centers/`.

[60] Rich Miller. Who has the most web servers? `http://www.datacenterknowledge.com/archives/2009/05/14/whos-got-the-most-web-servers/`.

[61] Kireeti Kompella. New take on sdn: Does mpls make sense in cloud data centers? `http://www.sdncentral.com/use-cases/does-mpls-make-sense-in-cloud-data-centers/2012/12/`.

[62] Jayaram Mudigonda, Praveen Yalagandula, Mohammad Al-Fares, and Jeffrey C. Mogul. Spain: Cots data-center ethernet for multipathing over arbitrary topologies. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 18–18, Berkeley, CA, USA, 2010. USENIX Association. `http://dl.acm.org/citation.cfm?id=1855711.1855729`.

[63] Charles Clos. A study of non-blocking switching networks. *Bell System Technical Journal, The*, 32(2):406–424, March 1953.

[64] Ronald van der Pol. Ieee 802.1ah basics (provider backbone bridges), March 2011.

[65] Marco Foschiano and Sanjib HomChaudhuri. Cisco systems' private vlans: Scalable security in a multi-client environment. RFC 5517, February 2010.

[66] Danny McPherson and Barry Dykes. Vlan aggregation for efficient ip address allocation. RFC 3069, February 2001.

[67] Charles E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Trans. Comput.*, 34(10):892–901, October 1985. `http://dl.acm.org/citation.cfm?id=4492.4495`.

[68] Radia Perlman. Rbridges: Transparent routing. In *Proceedings of the IEEE INFOCOMM 2004*, INFOCOMM '04, 2004.

[69] Radia Perlman, Donald E. Eastlake 3rd, Dinesh G. Dutt, Silvano Gai, and Anoop Ghanwani. Routing bridges (rbridges): Base protocol specification. RFC 6325, July 2011.

[70] David R. Oran. Osi is-is intra-domain routing protocol. RFC 1142, February 1990.

[71] *Information technology – Telecommunications and information exchange between systems – Intermediate system to Intermediate system intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service.* (ISO 8475), ISO/IEC 10589, 1992.

[72] Valentin Del Piccolo, Ahmed Amamou, William Dauchy, and Kamel Haddadou. Multi-tenant isolation in a trill based multi-campus network. In *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*, pages 51–57, Oct 2015.

[73] Russell Housley and Scott Hollenbeck. Etherip: Tunneling ethernet frames in ip datagrams. *Network Working Group, Request for Comments*, 3378, 2002.

[74] Cisco. *Cisco Active Network Abstraction Reference Guide, 3.7*, June 2010. Part 2 - Technology Support and Information Model Objects : Virtual Routing and Forwarding.

[75] S Kent and K Seo. Security architecture for the internet protocol. RFC 4301, December 2005.

[76] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The secure sockets layer (ssl) protocol version 3.0. RFC 6101, August 2011.

[77] Openstack security guide. Technical report, OpenStack Foundation, 2014. `http://docs.openstack.org/security-guide/security-guide.pdf`.

[78] Openflow. `https://www.opennetworking.org/sdn-resources/onf-specifications/openflow`.

[79] Amazon virtual private cloud. `http://aws.amazon.com/vpc/`.

[80] Dan Harkins and Dave Carrel. The internet key exchange (ike). RFC 2409, November 1998.

[81] Yakov Rekhter, Tony Li, and Susan Hares. A border gateway protocol 4 (bgp-4). RFC 4271, January 2006.

[82] Geoffrey Huang, Stephane Beaulieu, and Dany Rochefort. A traffic-based method of detecting dead internet key exchange (ike) peers. RFC 3706, February 2004.

[83] Pekka Savola. Mtu and fragmentation issues with in-the-network tunneling. RFC 4459, April 2006.

[84] Defense Advanced Research Projects Agency Information Processing Techniques Office. Internet protocol - darpa internet program - protocol specification. RFC 4459, September 1981.

[85] V. Fuller and D. Farinacci. Locator/id separation protocol (lisp) map-server interface. RFC 6833, January 2013.

[86] Fred Baker, Eliot Lear, and Ralph Droms. Procedures for renumbering an ipv6 network without a flag day. RFC 6325, September 2005.

[87] Charles E. Perkins. Ip mobility support for ipv4, revised. RFC 5944, November 2010.

[88] Charles E. Perkins, David B. Johnson, and Jari Arkko. Mobility support in ipv6. RFC 6275, July 2011.

[89] Jari Arkko, Christian Vogt, and Wassim Haddad. Enhanced route optimization for mobile ipv6. RFC 4866, May 2007.

[90] Dino Farinacci, Darrel Lewis, David Meyer, and Chris White. Lisp mobile node. work in progress, draft-meyer-lisp-mn-10, January 2014.

[91] Bhumip Khasnabish, Bin Liu, Baohua Lei, and Feng Wang. Mobility and interconnection of virtual machines and virtual network elements. work in progress, draft-khasnabish-vmmi-problems-03, December 2012.

[92] Murari Sridharan, Yu-Shun Wang, Pankaj Garg, and Praveen Balasubramanian. Nvgre-ext: Network virtualization using generic routing encapsulation extensions. work in progress, draft-sridharan-virtualization-nvgre-ext-02, June 2014.

[93] Technology white paper - trill. Technical report, HUAWEI TECHNOLOGIES CO., LTD., 2013. `http://www.huawei.com/ilink/enenterprise/download/HW_259594`.

[94] Alan Ford, Costin Raiciu, Mark Handley, and Olivier Bonaventure. Tcp extensions for multipath operation with multiple addresses. RFC 6824, January 2013.

[95] Stephen Nadas. Virtual router redundancy protocol (vrrp) version 3 for ipv4 and ipv6. RFC 5798, March 2010.

[96] Lisp overview... `http://lisp.cisco.com/lisp_over.html`.

[97] Ip addressing: Nat configuration guide, cisco ios xe release 3s. `http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipaddr_nat/configuration/xe-3s/nat-xe-3s-book.pdf`.

[98] Vic Liu, Bob Mandeville, Brooks Hickman, Weiguo Hao, and Zu Qiang. Problem statement for vxlan performance test. work in progress, draft-liu-nvo3-ps-vxlan-perfomance-00, July 2014.

[99] Victor Moreno, Fabio Maino, Darrel Lewis, Michael Smith, and Satyam Sinha. Lisp deployment considerations in data center networks. work in progress, draft-moreno-lisp-datacenter-deployment-00, February 2014.

[100] Darrel Lewis, David Meyer, Dino Farinacci, and Vince Fuller. Interworking between locator/id separation protocol (lisp) and non-lisp sites. RFC 6832, January 2013.

[101] Sami Boutros, Ali Sajassi, Samer Salam, Dennis Cai, Samir Thoria, Tapraj Singh, John Drake, and Jeff Tantsura. Vxlan dci using evpn. work in progress, draft-boutros-l2vpn-vxlan-evpn-04, July 2014.

[102] Gartner says nearly half of large enterprises will have hybrid cloud deployments by the end of 2017. `https://www.gartner.com/newsroom/id/2599315`.

[103] E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan. Conflict classification and analysis of distributed firewall policies. *IEEE Journal on Selected Areas in Communications*, 23(10):2069–2084, 2005.

[104] Radia Perlman, Donald Eastlake, Anoop Ghanwani, and Hongjun Zhai. Flexible multilevel trill (transparent interconnection of lots of links). Work in progress, draft-perlman-trill-rbridge-multilevel-07, January 2014.

[105] Sam Aldrin, Donald Eastlake, Tissa Senevirathne, Ayan Banerjee, and Santiago Alvarez. Trill data center interconnect. Work in progress, draft-aldrin-trill-data-center-interconnect-00, March 2012.

[106] Tissa Senevirathne, Les Ginsberg, Sam Aldrin, and Ayan Banerjee. Default nickname based approach for multilevel trill. Work in progress, draft-tissa-trill-multilevel-02, Mars 2013.

Valentin Del Piccolo is a Phd student at the University Pierre et Marie Curie (UPMC) and at GANDI SAS where he works on virtualization and multi-tenant isolation in data centers networks. He received his M.S degree in network and computer science from the University Pierre et Marie Curie in 2013, Paris, France.

Ahmed Amamou is a research engineer at GANDI SAS. He received the engineer degree in computer science from the National School of Computer science (Tunisia) in 2009 and the M.S degree in network and computer science from the same school in 2011; and the Ph.D degree in network and computer science from the University Pierre et Marie Curie in 2013, Paris, France. His research interests are Cloud computing and virtualization technologies. He is a member of the IEEE.

Kamel Haddadou received the engineering degree in computer science from INI in 2000, the M.S degree in data processing methods for industrial systems from the University of Versailles, and the PhD degree in computer networks from University Pierre et Marie Curie (UPMC), in 2002 and 2007, respectively. In 2001, he was a research assistant at the Advanced

Technology Development Centre (CDTA), Algiers, Algeria. He is currently a research fellow at the Gandi SAS, France. Since 2003, he has been involved in several projects funded by the European Commission and the French government (RAVIR, ADANETS, Adminroxy, GITAN, OGRE, ADANETS, MMQoS, SAFARI, and ARCADE). His research interests are focused primarily on Cloud computing and on resource management in wired and wireless networks. He is equally interested in designing new protocols and systems with theoretical concepts, and in providing practical implementations that are deployable in real environments. He has served as the TPC member for many international conferences, including IEEE ICC, GLOBECOM, and reviewer on a regular basis for major international journals and conferences in networking. He is a member of the IEEE.

Guy Pujolle received the PhD and "These d'Etat" degrees in computer science from the University of Paris IX and Paris XI in 1975 and 1978, respectively. He is currently a professor at University Pierre et Marie Curie (UPMC - Paris 6), a distinguished invited professor at POSTECH, Korea, and a member of the Institut Universitaire de France. During 1994-2000, he was a professor and the head of the Computer Science Department of Versailles University. He was also the professor and the head of the MASI Laboratory at Pierre et Marie Curie University (1981-1993), professor at ENST (1979-1981), and a member of the scientific staff of INRIA (1974-1979). He is the French representative at the Technical Committee on Networking at IFIP. He is an editor for ACM International Journal of Network Management, Telecommunication Systems, and an editor-in-chief of Annals of Telecommunications. He is a pioneer in high-speed networking having led the development of the first Gbit/s network to be tested in 1980. He has participated in several important patents like DPI or virtual networks. He is the cofounder of QoSMOS (www.qosmos.fr), Ucopia Communications (www.ucopia.com), Ginkgo-Networks (www.ginkgo-networks.com), EtherTrust (www.ethertrust.com), Virtuor (www.VirtuOR.fr), and Green Communications (www.greencommunications.fr). He is a senior member of the IEEE.

Table 2: Comparison of the protocols' complexities

Host isolation	**Diverter**	**BlueShield**	**NetLord**	**VL2**	**DOVE**	**LISP**	**NVGRE**	**STT**
Control plane	Distributed	Centralized, Directory Server (DS) Possible redundancy of DS	Centralized, Configuration repository	Centralized, directory system (ds)	Centralized, DOVE Policy service	Centralized, Directory Name Server (DNS)	Distributed	Distributed
Network restriction(s)	Flat Layer 2	Layer 2	Layer 2 and edge switches supporting IP forwarding	Layer 3 and Clos topologyg	Layer 3	Layer 3	Layer 3 network and No fragmentation of NVGRE packets	Layer 3 network and middle boxes (firewalls) must permit STT packets
Tunnel configuration and establishment *	Implicit	Implicit	Implicit	Implicit	Encapsulation protocol dependent	Implicit	Implicit	Implicit
Tunnel management and maintenance	Yes with forwarding table and rules in VNET	Yes, rules in the DS	Yes, with a SPAIN agent	Yes, mapping in the ds and VBL protocol	Encapsulation protocol dependent	Yes with mapping in the ITR and ETR	None	None
Multi-protocol	No, IP	Yes, Layer 2	No, Ethernet	No, IP	Encapsulation protocol dependent	No, IP	Yes, Layer-2	No, Ethernet
Security mechanism	VNET scans the traffic to enforce rules	Echelon VMs scan the traffic to enforce rules	None	VL2 agents enforce the rules	None	None	None	None

Core isolation	**PortLand**	**SEC2**	**802.1ad**	**802.1ah**	**VSITE**	**VNT**	**VXLAN**
Control plane	Centralized, Fabric manager for forwarding and addressing	Centralized, Central Controller (CC)	No	No	Centralized, Directory Server	No	No
Network restriction(s)	Layer 2 multi-rooted fat-tree	Layer 2	None	None	None	Layer 2 with TRILL enabled edges switch	Layer 3 network, no fragmentation of VXLAN packets and IGMP querier function
Tunnel configuration and establishment *	Implicit	Implicit	Explicit, GVRP	Explicit, GVRP	Explicit, MPLS VPN on public network and implicit in *vstub*	Implicit	Implicit
Tunnel management and maintenance	Yes, soft states	Yes, rules in CC	GVRP, join and leave messages by both end stations and Bridges	GVRP, join and leave messages by both end stations and Bridges	Yes, mapping in directory server and hypervisor	Yes, temporary forwarding database entry in RBridges	Join and leave messages by VTEPs
Multi-protocol	Yes	No, IP	No, Ethernet	No, Ethernet	No, Ethernet	Yes	No, Ethernet
Security mechanism	None	FEs enforce CC rules	None	None	Hypervisors enforce rules of directory server	None	None

* **Implicit**:based on connectionless IP service model. **Explicit**:tunnel establishing procedure such as control messages exchange or registration procedures.

Table 3: Comparison of the protocols' overhead

Host isolation	**Diverter**	**BlueShield**	**NetLord**	**VL2**	**DOVE**
Encapsulation header	None but IP addresse restriction	None	MAC and IP encapsulation with address rewriting (MAC+IPv4= 304 bits, MAC+IPv6= 464 bits)	IP header with a LA address (160 or 320 bits)	NVGRE, STT, or VXLAN headers
Messages	Multicast ARP messages	Directory look-up request	Adddress resolution based on Diverter model. SPAIN agent request to the repository	Messages for registration and mapping. Look-up requests. Messages for directory. IP-based link state routing protocol for LA address assignation	Policy requests Messages for registration of rules and topology.
Component(s)	VNET in each physical host with VNET ARP engine	Directory server, vSwitch, ebtables firewall, BlueShield agent, Echelon VM	NetLord Agent (NLA), SPAIN agent, Edge switches with IP routing capacities, Configuration repository	VL2 agent, Directory system	dSwitches, DOVE Policy Service (DPS)

Host isolation	**LISP**	**NVGRE**	**STT**
Encapsulation header	Outer IP header(IPV4: 160 bits, IPV6: 320 bits) + UDP header(64 bits)+ LISP header (64 bits) = 288(IPv4) or 448(IPv6) bits	Outer Ethernet header (144 bits) + Outer IP header(IPV4: 160 bits, IPV6: 320 bits) + NVGRE header (64 bits) = 368(IPv4) or 528(IPv6) bits	Outer Ethernet header (144 bits) + Outer IP header(IPV4: 160 bits, IPV6: 320 bits) + TCP-Like header(192 bits) + STT header(144 bits) = 640(IPv4) or 800(IPv6) bits
Messages	Map-Request, Map-Reply Map-Register, Map-Notify Encapsulated Control Message	None	None
Component(s)	xTR (ITR,ETR,PETR,PITR)	NVGRE Endpoints	STT Endpoints

Core isolation	**PortLand**	**SEC2**	**VSITE**	**VNT**
Encapsulation header	None	MAC header (144 bits)	MAC-in-MAC in Layer 2 network (144 bits) IP encapsulation in Layer 3 network (IPV4: 160 bits, IPV6: 320 bits)	Encapsulation with a VNT header (192 bits)
Messages	Unicast ARP in best case Worst case ARP broadcast to all end hosts Location Discovery Protocol messages Registration messages	Unicast ARP Customer messages for CC rules Uses GARP protocol	OTV-like protocol messages Directory lookup request	TRILL messages IS-IS protocol messages SPF tree generated based on Link State PDU (LSP) messages
Component(s)	Edge switches must perform MAC to PMAC header rewriting	Central Controller, Forwarding Elements, web portal	Directory server Cloud data center CEc VSITE agent	RBridges Virtual Switch

Core isolation	**802.1ad**	**802.1ah**	**VXLAN**
Encapsulation header	S-TAG (32 bits) + C-TAG (32 bits) = 64 bits	B-DA(48 bits) + B-SA(48 bits) + B-TAG(32 bits) + I-TAG(48 bits) = 176 bits +(optional) S-TAG (32 bits) + C-TAG (32 bits) = 240 bits	Outer Ethernet header(144 bits) + Outer IP Header: (IPv4=160 or IPv6=320 bits) + Outer UDP header: (64 bits) + VXLAN header: (64 bits) = 432(IPv4) or 592(IPv6)
Messages	Generic Attribute Registration Protocol	Generic Attribute Registration Protocol	Join and leave messages
Component(s)	Devices must abide by the 802.1ad standard	Devices must abide by the 802.1ah standard	VXLAN Tunnel EndPoints (VTEP)

Table 4: Comparison of the Host isolation protocols

Host isolation	**Diverter**	**BlueShield**	**NetLord**	**VL2**
Migration	Migration live or offline depending on time out values.	Live migration.	Live migration. Uses NetLord Agent messages (NLA): NLA-HERE, NLA-NOTHERE and NLA-WHERE to signal the VM migration. VM's IP or MAC address unchanged.	Live migration. Separation of location Addresses (LA) and application-specific addresses (AA).
Resilience	ECMP for multipath. Virtual gateway distributed among all the VM of the Sub-network.	Multiple replicas of the directory server. Possibility to use ECMP.	Relies on SPAIN for multipath. Configuration Repository might be replicated.	Resilience provided by a Clos topology. Redundancy of the Directory server
Scalability	16 millions VMs system wide. However number of client depend on the division of the IP address. The division must be done before starting the network, no modification after.	Centralized controller. CPU load lessen by replicating directory server but memory is limited.	16777216 Tenant_IDs (24 bits) $V \times R \times \sqrt{(\frac{F}{2})}$ virtual machines V = number of VMs per physical server R = switch radix, F = FIB size in entries.	One directory server can manage up to 17000 lookups/sec. The lookup rates increase linearly with the increase of servers.
Multi data center	Not specified. Possible with a Layer 2 interconnection. Control traffic travel between DC. Creates one big network over multiple DC	Not specified. Possibly a directory server replicated in each data center for inter-data centers communication rules.	Not specified. But possible with Layer 2 tunnels between data centers and one control plane spanning over all the data centers.	Not specified but possible. It will require an important directory system to manage the whole network.

Host isolation	**DOVE**	**LISP**	**NVGRE**	**STT**
Migration	Tunneling protocol dependent. Additionally dSwitch must inform the DPS when a new VM is detected by it.	IPv4 Mobility (RFC5944), IPv6 Mobility (RFC 6275, RFC 4866). Endpoint is an xTR itself.	REDIRECT messages	No STT mechanisms
Resilience	Multipath and routing resilience thanks to tunnel protocol. Redundancy of the Dove Policy Server.	Redundancy of xTR, MR and MS.	Multipath possible but not included in NVGRE	Multipath (ECMP) possible but not included in STT
Scalability	Number of tenants is tunnel protocol dependent: VXLAN has a 24 bits long VNI ≈ 16000000, NVGRE also has a 24 bits long VSID and STT has a 64 bits long Context ID $\approx 1.8 \times 10^{19}$. DPS is the scalability limiting component.	Big number of EIDs and RLOCs possible. One RLOC address associated with multiple EIDs addresses. Issue with the MS and MR maximum information saved.	One PA associated with multiple CA. Suppress most of the control plane broadcasts messages and convert some of them in multicast messages.	Context ID fields is 64 bits long. Issue with the virtual switch which can not manage this much IDs.
Multi data center	Not specified but possible. It will require an important DPS to manage the whole network.	Yes even with non LISP data center	Yes as a site-to-site VPN. Each site must have a NVGRE gateway	Theoretically, yes as a site-to-site VPN. Practically, no because of the middle boxes issue

Table 5: Comparison of the Core isolation protocols

Core isolation	**PortLand**	**SEC2**	**VSITE**	**VNT**
Migration	Live migration thanks to gratuitous ARP. Possibility of lessening the number of lost packets with redirection.	Live migration thanks to gratuitous ARP.	Live migration if the VM stays in the same location otherwise offline migration.	Live migration. Based on TRILL or MLTP which uses RBridge nicknames for forwarding the messages so VM's IP or MAC address unchanged.
Resilience	Fat-tree topology induced resilience. Fabric manager back up even with slightly non identical information.	Multiple FEs. Backups of Central Controller (CC). Can uses a Distributed controller instead of CC.	Master/slave switches configuration with the virtual router redundancy protocol.	ECMP for multipath. Redundant multicast distribution tree. No centralized controller.
Scalability	Centralized controller. Huge stress on Fabric manager. Not scalable by default: $\approx$ 27000 hosts with 25 ARP request/second = Fabric Manager with 15 CPUs.	Centralized controller. Huge stress on Centralized Controller. Possibility to transform the CC in a Distributed Controller. Only 4096 VLANs by edge domain. Number of edge domain is not limited but depend on MAC address usage.	VLAN for isolation. Aggregation of multiple VMs under a locIP.	Multiple VMs MAC addresses aggregated under one RBridge nickname. VNI TAG (24 bits) allows for 16777216 virtual networks.
Multi data center	Not specified. Layer 2 Fat-tree topology to interconnect the core switches of each data centers and get one network spanning over multiple DC. Not really feasible in reality seeing the cost induced by the interconnection topology.	Multi domains by design but scalability issue, only 4096 VLANs per domain.	Multi sites by design but scalability issue, only 4096 VLANs per sites.	Ready for multi data center. When using TRILL it creates one big network with one control plane spanning over all the data centers. Whereas MLTP keeps each data center independent. Needs Layer 2 tunnels between data centers.

Core isolation	**802.1ad**	**802.1ah**	**VXLAN**
Migration	Need to allocate resources for the VLAN in the destination network ahead of time to have session continuity. Migration restricted to the same Layer 2 network.	Need to allocate resources for the VLAN in the destination network ahead of time to have session continuity. Migration restricted to the same Layer 2 network.	Need to allocate resources for the VXLAN in the destination network ahead of time to have session continuity. Migration across Layer 3 network possible.
Resilience	Link aggregation and switches redundancy.	Link aggregation and switches redundancy	VTEP in hypervisor so redundancy of server in order to migrate the VMs to a new server if the hypervisor is down.
Scalability	VLAN limit up to 16777216	VLAN limit up to $\approx 7 \times 10^{16}$	16777216 VNI possible.
Multi data center	Yes with the same VLANs on all data center.	Yes with the same VLANs on all data center.	Possible to use VXLAN as a site-to-site VPN with VTEP gateways.