Rapport final du projet TouSIX : SFC en environnement SDN pour facilement ajouter des fonctionnalités et améliorer la robustesse de la fabrique de TouIX

Rémy Lapeyrade, Marc Bruyère, Philippe Owezarski

Rapport final du projet
(février 2017 – août 2018)

SFC en environnement SDN pour facilement ajouter des fonctionnalités et améliorer la robustesse de la fabrique de TouIX

Rémy Lapeyrade, Marc Bruyère, Philippe Owezarski

LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France

Mai 2019

Table des matières

1. Introduction

Les nœuds d'échange Internet (ou IXP pour Internet eXchange Point) sont des éléments critiques de l'architecture de l'Internet actuel pour l'interconnexion des réseaux. Leur importance et attrait croissants requièrent des niveaux de flexibilité importants que le concept de réseaux programmables (ou SDN : Software Defined Network) semble en mesure d'offrir. Ensemble, IXP et SDN présentent une promesse pour la mise en place et la gestion de fabriques de communication qui dépasse le traditionnel cadre intra-domaine [1], [2], [3]. En effet, en transportant d'énormes volumes de trafic et en interconnectant de multitudes de réseaux de types différents, les IXPs sont devenus des éléments centraux de l'Internet mondial [4], [5]. Ainsi, les IXPs sont devenus un type de réseaux universel affectant largement le partage des informations dans l'Internet.
Toutefois, transformer les IXPs de leur forme classique en une version complètement SDN représente un challenge scientifique et technologique. La scalabilité et la fiabilité sont deux aspects essentiels des IXPs qui ne peuvent pas être négligés au moment de la migration vers une solution SDN dont l'objectif premier est une plus grande souplesse et flexibilité d'utilisation, couplée à des besoins moindre en matière de ressources humaines. Par exemple, l'interruption des canaux de contrôle vers le plan de données peut provoquer de graves perturbations dans l'IXP, avec de potentielles répercussions sur des centaines de réseaux et des volumes de trafic conséquents [6], [7].
Au cours de la première année du projet TouSIX, nous avons conçu *Umbrella*, une nouvelle approche pour la gestion de la fabrique qui réduit le risque d'une dépendance excessive de la fabrique par rapport au plan de contrôle, et permettant en outre au contrôleur de ne gérer qu'une fabrique aux principes de conception simplifiés. Umbrella s'appuie sur la programmabilité des SDN pour aborder la gestion d'une partie du trafic, le trafic ARP (Address Resolution Protocol), directement dans le plan de données. Dans Umbrella, le seul rôle du contrôleur consiste à superviser le réseau, en s'appuyant sur sa connaissance globale du dit réseau.
Umbrella complète les architectures SDN précédentes pour les IXPs de deux façons : d'abord, Umbrella permet de gérer des IXPs de plus en plus complexes en mettant en œuvre des fabriques plus fiables et plus scalables. Umbrella permet ensuite de mettre en place des architectures SDN-IXP qui ne se limitent pas à des communications mono-saut, ce qui permet d'appliquer cette approche à la plupart des topologies d'IXPs. Nous envisageons les IXP-SDN avec des architectures dans lesquelles le contrôleur agit comme un superviseur intelligent plutôt que comme un élément de décision critique et dangereux. Umbrella est un premier pas dans cette direction.
Globalement, Umbrella fonctionne de la façon suivante : premièrement, le contrôleur Umbrella obtient directement la configuration des membres de l'IXP (association d'adresses MAC et IP, de numéros de ports, …), calcule ses chemins internes à l'IXP et les installe dans tous les commutateurs de l'IXP. Ensuite, pour chaque paquet entrant dans la fabrique, le commutateur OpenFlow (OF) d'entrée encode le numéro de chemin à suivre vers le commutateur de sortie dans le champ de l'adresse MAC de destination. Cela a la particularité par effet de bord de transformer tous les massages broadcast (comme le trafic ARP par exemple) en messages unicast. Les commutateurs de cœur utilisent alors le chemin ainsi encodé pour forwarder les paquets. A la fin, le commutateur de sortie rétablit l'adresse MAC de destination dans le champ approprié du paquet.

Les principales contributions des travaux effectués lors de la première année du projet TouSIX sont :

- Nous avons proposé l'architecture Umbrella et montré comment elle influence la programmabilité SDN dans le plan de données. Nous avons aussi montré comment déployer de façon incrémentale Umbrella, et démontré ses aspects pratiques au travers de son déploiement réel sur le réseau TouIX.
- Nous avons montré comment Umbrella étend les solutions actuelles pour les IXP-SDN. Umbrella est compatible avec les solutions actuelles [3] et permet leur implémentation sur des IXPs multi-sauts, tout en réduisant les risques de disfonctionnement du plan de données.
- Finalement, nous diffusons notre implémentation d'Umbrella, et notamment de son application TouSIX-manager qui génère les règles Umbrella et la configuration BIRD pour le server de route (ou RS pour Route server).

2. UMBRELLA : Une nouvelle fabrique SDN pour les IXPs

Les IXPs appliquent des règles strictes [19], [20] pour réduire l'effet de bord dû à l'utilisation d'un domaine broadcast partagé de niveau 2 ; par exemple, l'adresse MAC du routeur sur lequel les membres se connectent à la fabrique doit être connu à l'avance. C'est seulement à ce moment que l'IXP alloue un port Ethernet sur le commutateur de bordure et une adresse IP publique de son espace [21] et configure une liste de contrôle d'accès (ou ACL pour Access control List) avec cette adresse MAC. C'est ainsi que la localisation de tous les routeurs des membres est connue de l'IXP. C'est pour cela qu'Umbrella élimine le besoin d'un mécanisme de découverte des localisations des équipements et qui reposerait sur des messages en mode broadcast (i.e., requêtes ARP, IPv6 neighbor discovery) et par conséquent rend inutile un proxy-ARP comme cela est proposé dans les précédentes solutions pour les IXP-SDN [15], [16], [2], [3]. Umbrella convertit à la volée les paquets broadcast en paquets unicast en utilisant la possibilité offerte par OF de ré-écrire le champ d'adresse MAC et qui correspond à une règle donnée [22]. La Figure 1 montre une vision de haut niveau de l'idée proposée en utilisant la topologie d'IXP représentée sur la Figure 2 comme scénario de référence.

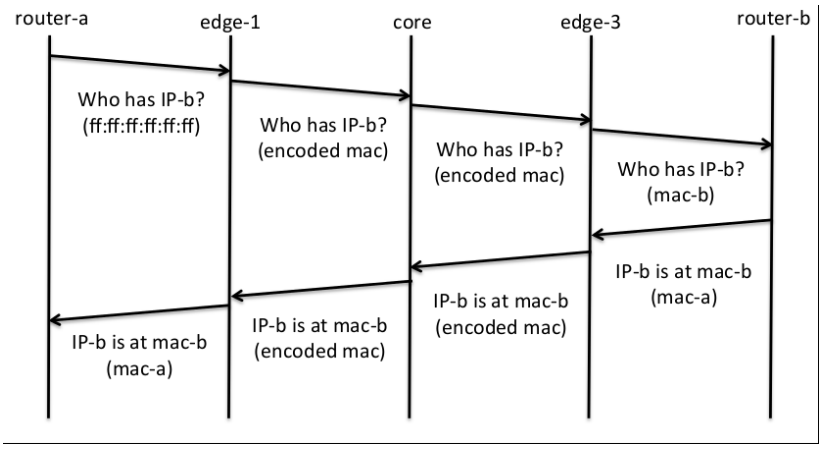

Figure 1. Handshake pour les requêtes ARP

Nous proposons d'utiliser un mécanisme de forwarding basé sur des labels pour réduire le nombre de règles dans le cœur de la fabrique de l'IXP. Les commutateurs de bordure Umbrella écrivent explicitement les ports destination pour chaque saut dans le champ destination de la trame MAC du paquet. Le premier octet de l'adresse MAC indique donc pour chacun des commutateurs de cœur traversés le numéro de port de sortie à utiliser.

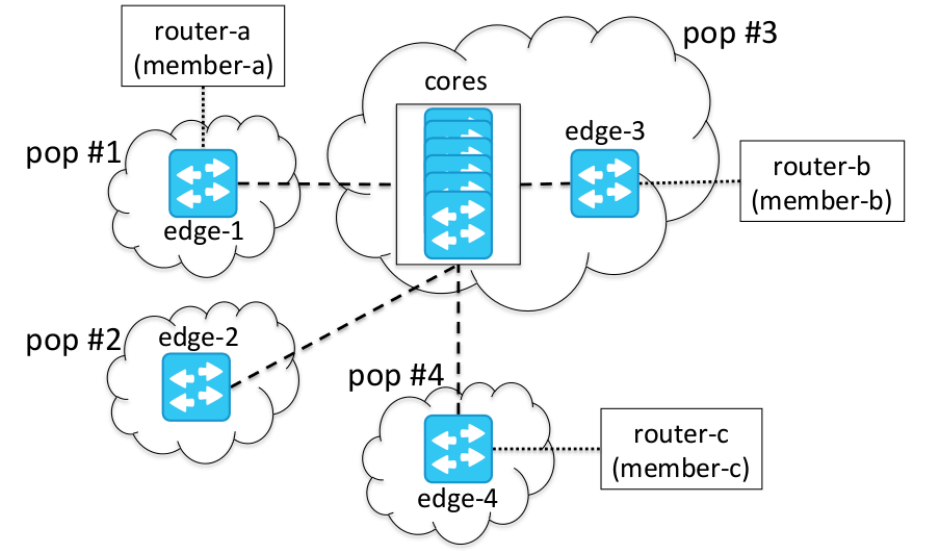

Figure 2. Topologie classique d'un IXP de taille significative

Le tableau 1 montre un exemple d'une table de flux d'un commutateur de cœur d'un IXP qui utilise l'approche Umbrella. Avec Umbrella, le nombre d'entrées dans la table des flux par commutateur de cœur reste limité par rapport au nombre de ports physiques actifs dans le commutateur. Cet aspect est important pour garantir la scalabilité de la fabrique. Le mécanisme d'encodage d'Umbrella est aujourd'hui limité à 256 ports de sortie par saut. Toutefois, il est tout à fait possible d'utiliser plus de bits si ce nombre n'est pas suffisant.

Adresse MAC de destination (Masque)	Port de sortie
01 :00 :00 :00 :00 :00 (ff :00 :00 :00 :00 :00	1
02 :00 :00 :00 :00 :00 (ff :00 :00 :00 :00 :00	2
03 :00 :00 :00 :00 :00 (ff :00 :00 :00 :00 :00	3
04 :00 :00 :00 :00 :00 (ff :00 :00 :00 :00 :00	4

Tableau 1. Exemple d'une table de flux Umbrella dans un commutateur de cœur d'un IXP

Nous expliquons maintenant comment Umbrella fonctionne sur la topologie décrite sur la Figure 2. Pour deux membres a et b qui veulent établir une association (peering), Umbrella détermine le chemin au sein de la fabrique de l'IXP de la façon suivante. Le commutateur de bordure edge-3 est connecté à un commutateur de cœur par le port 2 et à router-b par le port 3. Le routeur router-a du membre member-a envoie une requête ARP (i.e., un message broadcast) au routeur router-b. Le commutateur edge-1 reçoit la trame, ré-écrit la trame avec le champ d'adresse MAC de destination avec les ports correspondants, 2 et 3, 02:03:00:00:00:00, et le forwarde au commutateur de cœur indiqué. Lorsque la trame atteint le cœur, elle est redirigée vers le port de sortie 2, et ensuite vers le commutateur edge-3 (i.e.,

le forwarding dans le cœur repose sur l'octet de poids fort). Enfin, edge-3, avant de forwarder la trame par le port de sortie indiqué dans le second octet de l'adresse MAC, ré-écrit ce champs avec l'adresse réelle du routeur router-b.

Lorsque la source et la destination sont directement connectées au même commutateur de bordure, aucun encodage n'est requis, et l'adresse broadcast de destination est directement remplacée par l'adresse MAC de destination par le commutateur de bordure concerné. Dans un scénario IPv6, l'indication OF dans le commutateur de bordure doit être placée dans le champs *IPv6 ND target* du paquet de sollicitation *ICMPv6 Neighbor* [23]. La table de correspondance sur le commutateur de bordure doit conserver les associations entre les adresses IPv6 et leur localisation, comme dans le cas IP4.

Une approche de commutation basée sur des labels

Le mécanisme de propagation (forwarding) d'umbrella permet d'utiliser des commutateurs traditionnels (sans implémentation d'OF) dans le cœur, limitant ainsi les investissements (et coûts) de mise à jours des matériels. Un commutateur de cœur doit seulement forwarder les paquets sur la base de règles de filtrage d'accès, alors que les commutateurs de bordure doivent intégrer OF pour ré-écrire le champ MAC de destination.

Cette approche est directement applicable pour des IXPs permettant un seul saut dans le cœur (comme AMS-IX et DE-CIX), mais n'est pas applicable pour des fabriques autorisant plusieurs sauts (comme LINX et MSK-IX). Avec un seul saut, le port de sortie est encodé dans l'octet de poids fort de l'adresse MAC de destination. Dans le cas multi-sauts, comme un paquet peut traverser plusieurs commutateurs de cœur, un nouveau mécanisme d'encodage est requis pour indiquer les différents ports de sortie au niveau des différents commutateurs de cœur.

La figure 3 présente un exemple de fabrique d'IXP à sauts multiples dans le cœur. Le commutateur de bordure edge-a doit traverser deux commutateurs de cœur sur le chemin b pour atteindre edge-d. C'est un cas très fréquent dans les topologies hypercube, comme celles adoptées par LINX ou MSK-IX.

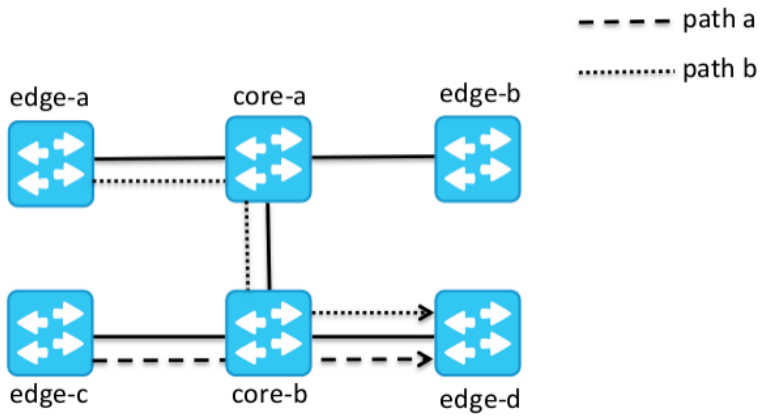

Figure 3. Exemple de cœur multi-sauts

Adapter Umbrella aux topologies multi-sauts d'IXP n'est pas trivial. Il faudrait pour cela définir un mécanisme d'encodage des adresses MAC de destination dans lequel l'octet de poids fort indique le port de sortie du premier commutateur de cœur (i.e., core-a), le second octet indique le port de sortie du second commutateur de cœur (i.e., core-b), et ainsi de suite. C'est tout bonnement impossible. En effet, selon la route, un commutateur de cœur peut être soit le premier soit le second sur le chemin à suivre. Une autre solution pourrait consister à considérer les ports d'entrée de la trame dans les règles de forwarding installées au niveau des commutateurs de cœur. Avec les ports d'entrée, on peut savoir où se situe le commutateur sur le chemin entre la source et la destination, et ainsi considérer le bon octet dans le champ de l'adresse MAC de destination. Hélas, cette approche ne peut pas fonctionner sur des topologies arbitraires. De plus, un tel mécanisme conduirait à une explosion du nombre de règles dans le cœur, le nombre de règles de forwarding grandissant de façon quadratique avec le nombre de ports d'entrée possibles.

En mettant en place une méthode à base de *source routing*, Umbrella s'absout enfin de ce problème. Dans l'approche Umbrella pour les fabriques multi-sauts d'IXP, le premier commutateur de bordure doit sélectionner le chemin à emprunter pour chaque paquet. Une liste ordonnée des ports de sortie est alors encodée dans le champ d'adresse MAC de destination, comme une pile de labels. Finalement, chaque nœud de cœur traite les trames en considérant la valeur au sommet de la pile, et la dépile avant de les forwarder (Figure 4).

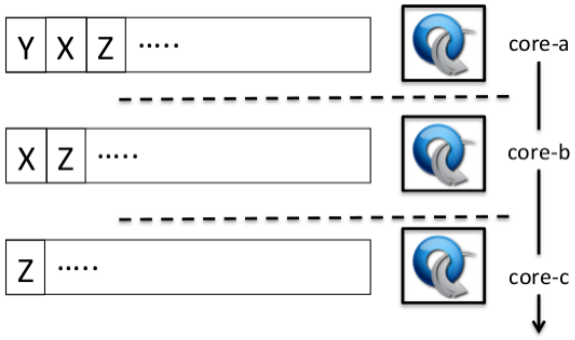

Figure 4. Exemple pour le mécanisme de forwarding d'Umbrella

Avec cette configuration, chaque commutateur n'a besoin de ne regarder que l'octet de poids fort de l'adresse, peut importe la place qu'il occupe sur le chemin vers la destination. Retirer de la pile le dernier label utilisé nécessite de pouvoir ré-écrire les entêtes, rendant cette solution possible uniquement sur des commutateurs de cœur compatibles avec OF. En particulier, chaque commutateur de cœur doit posséder deux tables d'actions : forwarding et *coppy-field*[1]. Cette solution présente donc deux limites pratiques (qui seront détaillées et traiter plus longuement dans la partie 5.2) :

- Le nombre maximum de ports de sorties qui peuvent être adressés lors de chaque saut est limité à 256, puisque le numéro de port de sortie pour chaque commutateur est encodé dans l'octet de poids fort du champ d'adresse MAC de destination.

- [1] Les specifications d'OF 1.5 intègrent ces capacités de copie et écriture des champs des entêtes.

- Le nombre maximal de sauts au sein de l'IXP est limité à 6, car nous ne pouvons utiliser que les 6 octets de l'adresse MAC pour encoder tout le cheminement des trames.

Cependant, nous devons surtout considérer Umbrella comme une approche permettant une séparation plus forte entre les plans de contrôle et de données. Les limitations pratiques peuvent se résoudre avec l'apparition de nouvelles implémentations des différents protocoles.

3. Déploiement et évaluation d'Umbrella sur TouIX : de TouIX à TouSIX

Le réseau TouIX était une infrastructure interconnectée autour de 4 POPs (Point of Presence) et interconnectée avec deux autres IXP : FranceIX et LyonIX. TouIX a été renommé TouSIX après sa migration vers une approche OF. Depuis mai 2015, TouSIX est le premier IXP européen (et à ce jour le seul) à exploiter la technologie OF pour ses opération quotidiennes. L'architecture TouSIX utilise l'approche Umbrella. Cette partie commence par présenter les travaux de migration de TouIX vers une solution OF, puis montre ensuite comment TouSIX et son approche Umbrella fonctionnent et résolvent les problèmes présents avec l'ancien réseau TouIX.

TouIX

La partie supérieure de la Figure 5 montre la topologie de TouIX. Le site primaire a été Cogent où le BIRD RS a été installé. Les équipement CISCO[2] utilisés dans la fabrique ont été configurés avec trois VLAN : un VLAN admin, un VLAN TouIX et un VLAN IX pour les 2 IXP externes connectés. Comme les équipements CISCO installés étaient vieillissants et plus réellement appropriés pour la fabrique, il a été décidé de refondre complètement le réseau et de créer un tout nouvel IX pour faciliter sa gestion

En particulier, les mauvaises configurations de membres (i.e., avec des boucles non souhaitées) affectaient l'ensemble des opérations de la fabrique, et rendaient délicate la gestion de l'IXP. Plusieurs « tempêtes ARP » (ARP storm) ont été subies. L'absence d'un système de monitoring moderne et efficace couplée à des apparitions irrégulières de ce phénomène l'ont rendu difficile à détecter. Les commutateurs CISCO ne disposaient pas des règles de filtrage requises pour empêcher ce trafic non voulu de perturber la fabrique. Protéger la fabrique du trafic indésirable est une nécessité, non seulement pour protéger les routeurs pairs, mais aussi pour améliorer la stabilité et la facilité de gestion de toute l'infrastructure. Umbrella a été notre réponse pour solutionner ces problèmes.

[2] TouIX used Cisco WS-C3550-12G (TLS00), WS-C3560G (Cogent), WS- C2960G-24TC-L (Zayo) and WS-C2960G-24TC-L (Hotel Telcom).

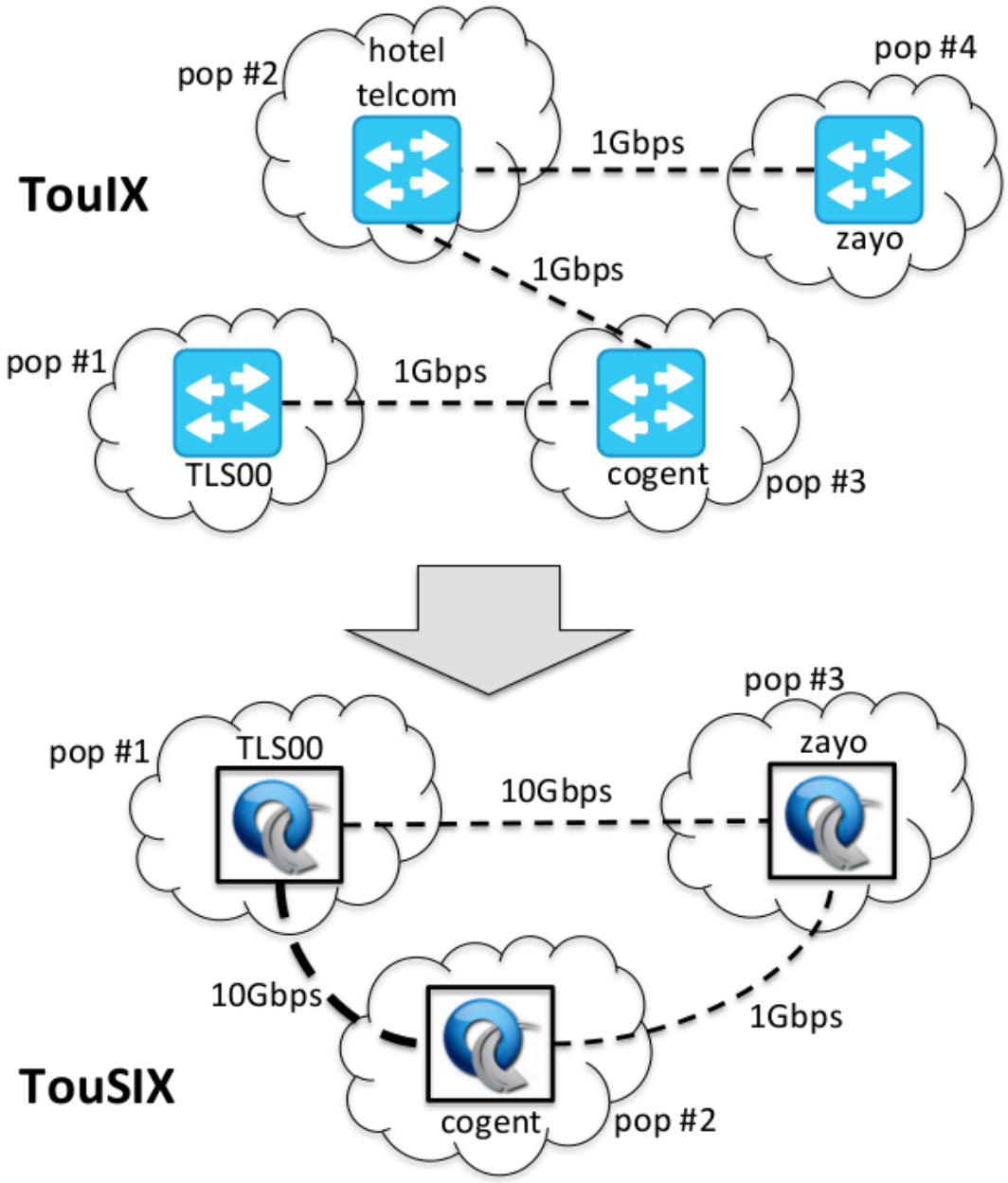

Figure 5. D'un IXP classique à un IXP complètement opéré avec OF

TouSIX : un IXP totalement opéré par SDN

La nouvelle topologie de TouSIX a trois PoPS, avec un commutateur OF sur chacun d'eux qui agit en tant que commutateurs de bordure. Il n'y a pas de routeur de cœur (cf. partie basse de la Figure 5). Les trois commutateurs ont été programmés selon les principes d'umbrella et remplacent les quatre anciens commutateurs de TouIX. Le site de l'*Hotel des Télécoms* a été supprimé durant cette migration et son unique membre a été connecté directement au commutateur de *Cogent*.

Une longue période de test était requise pour valider cette nouvelle configuration système et sa stabilité avant de l'utiliser en production.

Au début de la campagne de test, seuls les commutateurs Pica8 intégraient de façon matérielle un agent Open vSwitch, faisant du commutateur Pica8 P-3290 le choix naturel pour TouSIX. Il a été décidé de faire reposer TouSIX sur le commutateur logiciel le plus largement utilisé dans le monde, de plus avec un niveau de performance similaire à celui de solutions hardware. Toutefois, la dernière version du système PicOS au début de la campagne de test ne permettait pas d'installer des règles mentionnant le champ ipv6 nd. Cette option est requise pour mettre en œuvre les mécanismes d'Umbrella sur le trafic IPv6 (cf. section 5.1). De plus, le système logiciel PicOS vidait entièrement la table de flux lors de reboots matériels, ou lorsque la connexion avec le contrôleur était perdue. En collaboration avec Pica8, ces problèmes ont pu être résolus, et ont conduit à une nouvelle version de PicOS, i.e., 2.6.

La période de tests a duré quelques mois, et la version de la nouvelle fabrique a pu être installée. Les commutateurs Pica8 fonctionnant en mode Open vSwitch ont été placés au dessus des commutateurs Cisco. Les commutateurs Pica8 servent pour le plan de données, et les Cisco ont été conservés pour transporter le trafic du plan de contrôle. Grâce aux nouveaux commutateurs, la bande passante a été augmentée d'un facteur 2 ou 3 entre les commutateurs de bordure. De plus, la connexion entre *TLS00* et *Cogent* a été améliorée avec le mécanisme d'agrégation de liens (Link AGgregation, LAG). Les câbles des membres ont été migrés des commutateurs Cisco vers les Pica8, avec seulement quelques secondes d'interruption de service pour chaque membre. Les commutateurs Pica8 ont été configurés pour ne pas vider la table des flux lorsque l'agent OF perd sa connexion avec le contrôleur OF. Le contrôleur n'est cependant plus le point de faiblesse du réseau grâce à la forte séparation entre les plans de contrôle et de données. Le déploiement en cours repose sur Ryu [18], le contrôleur open-source des NTT Labs, alors qu'on attend de pouvoir évaluer le contrôleur ONOS de ON.Labs [30] et son parallélisme. D'autre part, les développeurs utilisent l'API REST Pica8 pour communiquer avec une interface graphique (GUI : Graphical User Interface), qui simplifie beaucoup les tâches opérationnelles classiques.

1) Expérience avec un déploiement réel : La fabrique Umbrella de TouSIX a fonctionné sans problème et avec un support réduit des administrateurs. Le Tableau 2 trace certains résultats avec Umbrella en fonctionnement réel : conversion du trafic ARP broadcast en unicast, blocage du trafic non désirable, annonce de 399 préfixes IP. Umbrella réduit ainsi le volume total moyen du trafic ARP qui transite sur la fabrique de 97%. De plus, une application de gestion – le TouSIX-manager – a été développée pour permettre aux membres de configurer leur peering directement, en générant automatiquement les règles OF pour le commutateur SDN, ainsi que pour configurer le RS.

Le Tableau 3 montre le trafic moyen et crête sur TouSIX. TouSIX concentre le trafic de 14 nœuds, comprenant les routeurs et les serveurs (même s'ils ne sont pas tous actifs à un instant donné, toutes les règles correspondantes restent installées). Les commutateurs doivent intégrer des règles pour le forwarding MAC, le routage ARP et le routage ICMPv6 NS. Comme seul IPv4 était en production à cette époque, le nombre total de règles par commutateurs est de 28. Umbrella permet aussi de limiter la consommation de ressources qui présente des niveaux d'utilisation CPU sans pics, et stable aux alentours de 8%. A l'inverse, les commutateurs du TouIX subissaient des pics de charge atteignant les 100% de leur capacité. Ces pics de charge CPU des anciens commutateurs TouIX étaient le résultat de problèmes de convergence de différents protocoles du plan de contrôle.

En terme de temps de récupération sur erreur, lorsqu'un commutateur avec une table vide se

connecte au contrôleur, installer les règles de flux (i.e., l'appel à la base de données et d'émission des règles vers le commutateur) prend environ 5.6 secondes, et un reboot hard prend environ 64 secondes. Avec Umbrella, la connexion d'un nouveau routeur à TouSIX marche selon le principe "plug & play" : l'administrateur installe automatiquement les règles de flux pour tout router « approuvé », car le contrôleur connaît déjà sa configuration. Par conséquent, le temps entre le moment où un commutateur est connecté, et son passage en mode opérationnel est négligeable. A l'inverse, si un routeur est connecté sans l'approbation de l'administrateur, tout son trafic est éliminé. Comme le TouSIX-Manager opère la topologie avec des procédures proactives, si le lien est perdu au niveau du canal de contrôle, toutes les règles déjà en place sur le commutateur demeurent. Nous avons testé l'effet de l'arrêt du TouSIX-manager et avons vérifié que les statistiques sur le trafic restent les mêmes sans avoir d'impact sur les opérations de la fabrique.

	TouIX	TouSIX
Max. (Pkt/s)	14,96	3
Average (Pkt/s)	8,51	1,18
Min. (Pkt/s)	1,1	0

Tableau 2. Trafic ARP (en paquets par seconde) dans TouIX et dans TouSIX.

	Commutateur 1	Commutateur 2	Commutateur 3
Peak (B/s)	4874601415	5367083113	268646
Average (B/s)	1975571	43955853	103

Tableau3. Trafic moyen et crête (en octets par seconde) dans chacun des commutateurs TouSIX.

4. Amélioration de la robustesse des SFC dans un IXP

L'apport d'un traitement particulier des paquets dans le réseau a toujours fait partie du paysage des architectures réseaux. Dans la plupart des cas, ce traitement a conduit à l'apparition de middlebox pour exécuter des services spécifiques. Des exemples de ces middlebox sont les firewall, IDS, DPI, … Ces middlebox sont désormais confrontées à des problèmes de flexibilité. En effet, elles sont très peu malléables, et demandent une intervention humaine afin de fonctionner dans des conditions changeantes. Pour proposer une meilleure flexibilité dans le traitement du trafic, il est possible de déporter les différents services pour la propagation des paquets vers des instances virtuelles. Cela a pour but de faciliter la gestion du réseau [35]. Il n'en demeure pas moins que pour mettre en œuvre cette approche, il faut trouver un moyen flexible de mettre en relation les différentes classes de trafic et le traitement que l'on souhaite leur appliquer.

Pour palier ce problème, il est possible d'utiliser l'approche SFC (Service Function Chaining) [36]. Cette approche propose des avantages pour gérer dynamiquement les besoins de traitement de flux réseau et nous l'avons appliquée à la technologie SDN sur les réseaux IXP, et mis en œuvre sur TouSIX. En effet, la chainage de fonctions (SFC) est un attribut clé des SDN. Avec cet attribut, les SDN peuvent dynamiquement connecter des machines virtuelles (VMs) proposant différentes fonctions pour établir de nouveaux services sur la base des besoins utilisateurs et de l'état du réseau [40]. Par exemple, une chaine de service de sécurité avancée peut demander au firewall et à l'IDS d'inspecter séquentiellement les données, alors qu'un service de sécurité classique ne pourra faire appel qu'à l'un au plus de ces équipements. Ainsi, le chainage de services est le processus de déploiement requis pour construire une suite de fonctions de service individuelles permettant de réaliser une tâche complexe. Sur l'exemple de sécurité avancée considéré plus haut, on peut imaginer un service de sécurité avancé qui fasse appel à une fonction de détection et de prévention d'intrusion, une autre de firewall, une fonction de filtrage de contenu, et une dernière sur la mise en œuvre de mécanismes d'optimisation. Ces services de sécurité seront positionnés en plusieurs points du réseau (e.g. dans les commutateurs et/ou routeurs et/ou machines standard), et le routage à mettre en place devra permettre aux paquets à inspecter de traverser chacune de ses fonctions (simples) de sécurité [41]. Ainsi, si un flux de données provient d'une source non fiable, il devra traverser les fonctions et/ou équipements de sécurité en accord avec la politique du réseau, aussi appelée « Rule Based Forwarding » [42]. De plus, un membre ou un utilisateur peut spécifier quels composants ou fonction réseau ses paquets doivent traverser. Le fournisseur de services réseaux doit alors faire en sorte que les paquets passent dans les équipements spécifiés par l'utilisateur. Avec les progrès actuels en matière de virtualisation des machines, les services réseaux sont implémentés à l'aide de VMs distribuées exécutant une ou un ensemble de fonctions de service [43]. Ainsi, la combinaison des services implémentés sur des VMs et la possibilités de les chainer est de nature à permettre d'améliorer la scalabilité et la robustesse de ces réseaux [1][2].

Cette partie va donc montrer comment l'intégration des SFC dans un IXP comme TouSIX est possible. Pour cela, nous allons en outre proposer une architecture pour répondre aux besoins de gestion des politiques SFC [37] qui permette de proposer une plus grande flexibilité et un catalogue de services par les membres pour les membres [38]. Cela implique cependant que l'opérateur de l'IXP connaisse les échanges de politiques entre les différents membres, afin de les appliquer. Il aura donc une responsabilité à prendre sur la partie de déploiement des SFC. Une garantie supplémentaire de robustesse doit être assurée sur ces politiques pour garantir le bon traitement du trafic.

Architecture SFC

Les membres d'un IXP pouvant être à l'origine de la mise en place de certain chainages, en plus de l'opérateur lui même, deux types de services de chainage doivent être proposés : les services de chainage gérés par l'IXP et ceux demandés plus ou moins ponctuellement par les membres. Cela amène donc à une architecture de gestion des communications de l'IXP intégrant un orchestrateur SFC (Figure 6). Ainsi, avant d'établir un chaînage et de déployer le service voulu, un des membres (le membre A ou B) doit envoyer une requête de SFC. Une fois reçue par l'orchestrateur, ce dernier demande son accord à l'autre membre. Les principales raisons de cette demande d'acquittement sont :

- de notifier l'autre membre d'une modification du trafic entrant/sortant. Ce membre est en effet en droit de savoir si l'intermédiaire de connexion effectue des modifications sur les flux et quelles sont ces modifications.

- de pouvoir fournir des informations supplémentaires sur la gestion du trafic.

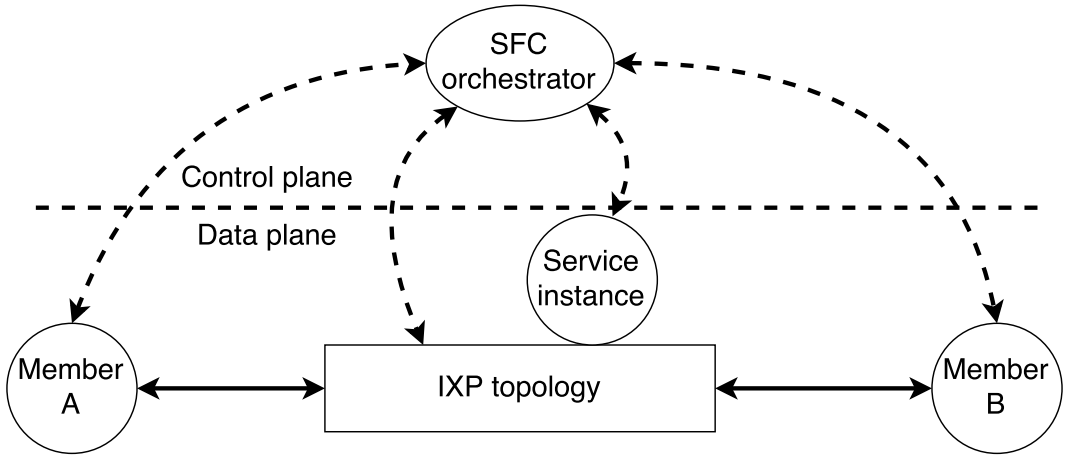

Figure 6. Principes de communication entre le réseau et l'orchestrateur SFC de l'IXP

Etant donné la conception de TouSIX, il est à noter que certains services ne peuvent pas être appliqués directement. Par exemple, considérons un membre de l'IXP, qui souhaite utiliser une middlebox pour proposer un service qui remplisse la fonction de proxy HTTP. Des modifications des entêtes de niveau 3 sont forcément requises pour ce type de service. Or elles ne sont pas compatibles avec le statut d'un IXP.
Donc, pour les différents types de services qui sont proposés dans le réseau, nous devons prendre en compte la nature des champs qui doivent être analysés par les routeurs. Si les routeurs échangent essentiellement des informations de niveau 3, cela exclut toute modification de ces champs pour un service, de part la fonction même d'un IXP. Au contraire, des modifications intervenant sur les autres niveaux sont acceptables.
Concernant la gestion de la politique de SFC, l'IXP sera donc en charge de :

- Appliquer des modifications en fin de chaînage (notamment restaurer les champs d'adressage qui ont pu être modifiés pour la durée de leur passage dans l'IXP) afin que le routeur de destination puisse récupérer un trafic exploitable.
- Appliquer la politique sur la topologie de l'IXP.

Renforcement de la robustesse d'un chainage SFC

Les IXP ont pour contrainte de fournir un service ayant une fiabilité forte. En ajoutant une fonctionnalité de SFC en parallèle de leurs contraintes de commutation, cela impose de rendre robuste aussi la fonction de chainage.

La figure 7 illustre le problème qui peut se produire avec les SFC. Dans ce cas, un chemin a déjà été établi pour un chaînage de service dans la topologie, en bleu. Si un des liens est désactivé, une procédure de secours sera enclenchée pour le trafic passant par ce lien. Par conséquent, le chemin suivi par le chaînage de service est donné en rouge dans la figure 7. Si le chemin de secours représente une solution viable pour le trafic régulier d'un IXP, il dégrade le fonctionnement de la politique SFC dans la topologie, en utilisant un même lien pour finir la liaison avec l'instance de service B. Un autre chemin devrait donc être établi pour ce chaînage ; il est représenté en vert sur la figure 7. Ainsi, un meilleur équilibre de trafic serait atteint pour une meilleure QoS.
C'est pourquoi, il est utile de prendre en compte ce risque avant le déploiement, et de fournir

un chemin alternatif qui serait préventivement déjà déployé dans la topologie. En utilisant un chemin alternatif sur la topologie, qui serait dédié à la gestion de la politique de SFC, il est possible de proposer un meilleur chemin que celui prévu pour le trafic normal. Bien sûr, le comportement dégradé peut être utilisé jusqu'a ce que l'incident soit résolu, mais dans d'un réseau métropolitain comme les IXPs, l'intervention peut arriver plusieurs heures après sa détection. Or, en utilisant la technologie SDN, il est possible de séparer le traitement de chaque politique de façon claire et efficace, et donc de ne pas subir de perturbation en cas de panne de lien, et ce grâce à la flexibilité du routage qui peut être mis en œuvre.

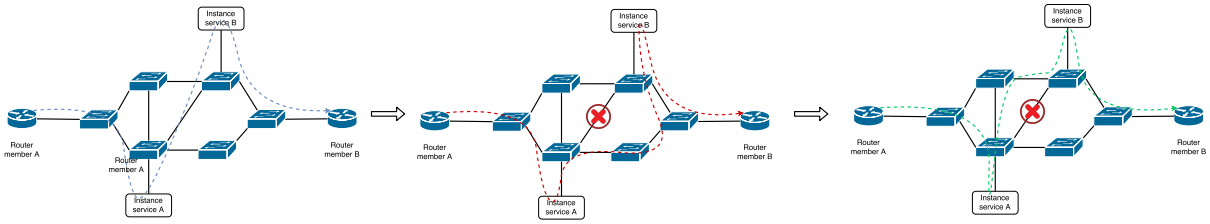

Figure 7. Scénario conduisant à un service SFC sous optimal

La conception du module de SFC que nous proposons pour l'IXP TouSIX doit prendre en compte 3 contraintes :

- Maintenir l'indépendance avec le plan de contrôle du point de vue des commutateurs. Cela implique la construction d'une procédure proactive pour l'application de la politique SFC.
- Prêter attention aux possibles pannes pouvant survenir dans le SFC.
- Avoir la possibilité de comprendre et de modifier facilement les SFC déployés.

La mise en œuvre d'un chainage impose de respecter les besoins de l'IXP ou de ses membres. Une première étape consiste pour l'opérateur à valider les politiques SFC envoyées par les membres. Une fois les politiques SFC validées, les SFC sont prêts à être déployée. Les polices de déploiement des SFC sont illustrées sur l'exemple du Listing 1. Elles comportent 3 types d'information :

- Les premiers éléments d'information portent sur les caractéristiques des hôtes. Ces informations sont généralement données par l'opérateur de l'IXP.
- La police de déploiement comporte ensuite les éléments permettant de classifier le flux SFC dans le plan de données. Ce sont des champs propres à OpenFlow. Ils servent pour le SFC à identifier les flux au niveau des switches de bordure / accès à la topologie de l'IXP.
- Enfin, un descriptif du chaînage des différents services. Dans notre approche, cela comprend le type et la localisation du service dans un ordre strict.

```
{
”host src”: ”host1”,
”host dst”: ”host2”,
        ”mac dst”: ”82:17:14:C6:2E:10”,
        ”flow”: {”udp dst”: 99,
                ”ip proto”: 17,
                ”eth type”: 2048,
                ”ipv4 dst”: ”10.2.0.2”},
        ”services”: [
        {”image”: ”local:img srv 1”, ”entry”: ”edge–4”},
        {”image”: ”local:img srv 2”, ”entry”: ”edge–3”} ],
```

```
        ”redirect”: true
}
```

Listing 1. Informations envoyées au module SFC (au format JSON)

Comme déjà présenté, si nous souhaitons avoir des changements de chemins lors de pannes de liens, et ce sans impliquer le contrôleur, nous devons placer les règles de routage pro-activement pour anticiper ces pannes. Un des aspects du chaînage dans notre approche est la sélection du lien sur lequel effectuer le prochain saut. A chaque saut, il y a une possibilité de ré-évaluer et de corriger le chemin avant d'envoyer le flux vers le prochain service. Cette détermination du prochain saut peut s'effectuer sur les switches d'accès en utilisant la fonctionnalité de groupes définie dans le protocol OpenFlow [3]. Il est ainsi possible de choisir des chemins alternatifs en vérifiant le statut du lien pour le prochain saut.
Les groupes OpenFLow sont des entrées sur les switches compatibles OpenFlow qui permettent d'améliorer les possibilités de commutation. Il est possible de préparer sur une entité de sortie, plusieurs instructions à exécuter selon des conditions définies dans le groupe. Dans notre cas, nous utilisons des groupes de type Fast-Failover. Ils sont utilisés pour vérifier l'état des ports de sortie du switch ou l'état d'autres groupes de ports de sortie dans le switch. Les instructions sont alors appliquées selon le statut du port ou du groupe. La première entrée active (correspondant au premier lien en ordre de marche rencontré dans le groupe) sera sélectionnée, selon l'ordre qui a été défini dans le groupe.
Le but ici est de proposer un moyen d'insérer des instructions de source-routing en fonction du statut du lien à chaque saut du chaînage de service. Cela permet de changer les routes rapidement sans avoir à solliciter le contrôleur pendant la défaillance d'un ou plusieurs liens.
Dans le contexte des SFC, il y a la possibilité d'utiliser des groupes pendant l'encapsulation du flux, et à chaque étape de traitement par un service. Cela nous autorise donc à proposer cette méthode à chaque fois que le flux est traité par un switch d'accès.

Pour constituer les ensembles de meilleurs chemins, au moment de la configuration du réseau SDN préalable à sa mise en route, nous évaluons un scénario de déploiement de SFC pour chaque ensemble d'instructions que l'on souhaite construire. Sur ces scénarios, nous allons ajouter une panne probable sur le chemin vers le prochain saut, et trouver le chemin optimal pour chaque défaillance possible. De cette manière, dans chaque cas de figure, nous déterminons un chemin qui peut être appliqué. Nous allons ensuite encoder ces chemins optimaux dans des champs du paquet non-utilisés par le service, et qui peuvent être restaurés à l'arrivée au prochain saut. Typiquement les adresses peuvent être modifiées librement dans le domaine d'un IXP, vu que le positionnement et les caractéristiques des machines sont connus d'avance.

Group Number	Type	Buckets
Id group	Fast Failover	Bucket 0 : Instruction optimale Bucket 1 : Première instruction de secours Bucket 2 : Seconde instruction de secours …

Tableau 4. Définitions des groupes pour le module SFC

Le tableau 4 détaille le principe de changement des routes SFC en cas de panne d'un lien sur le chemin SFC. Le groupe couvre trois scénarios qui seront sélectionnées selon l'état des liens de sortie possibles à l'arrivée d'un paquet sur le switch. Ainsi, si le lien classique est en

panne, le chemin SFC basculera sur le premier lien de secours, et ainsi de suite.
Pour mettre en relation la classification du flux et les décisions de commutation, nous avons besoin au final que d'une seule règle de flux OpenFlow. Cette règle, décrite dans le tableau 5, consiste donc lorsqu'un paquet d'une requête SFC est reçu et reconnu à lui faire respecter les instructions décrites pour le groupe SFC..

OpenFlow Match	OpenFlow Actions
Packet matches SFC flow	goto service group

Tableau 5. Correspondance entre les règles sur les flux OpenFlow et SFC

Au final, lorsque le flux doit être redirigé vers le destinataire final, c'est le module SFC qui s'en charge. A l'initialisation, un chemin additionnel a justement été créé pour rediriger le flux. Ensuite, des instructions de ré-écriture des champs modifiés par le module SFC est effectuée pour restaurer les adresses et autres caractéristiques du routeur de destination.
Il y a donc seulement deux règles OpenFlow par service à déployer :

- une pour définir le groupe avec les différentes situations de pannes de liens ordonnées.
- une règle de flux pour identifier le flux.

Les règles de flux doivent être placées sur les switches d'accès. Pour les switches de cœur dans le domaine de l'IXP, l'identification des flux n'est pas à faire, car cette information est fournie sur la base des instructions de commutation qui ont été définies dans les groupes du switch d'accès, et donc propagée de proche en proche à chaque saut.

Evaluation

L'évaluation proposée se décline en 3 parties :

- Un premier test est une validation de notre approche sur différentes topologies.
- Le second test est vise à évaluer notre solution sur des procédures de fonctionnement d'un IXP réel. Cette partie est effectuée en utilisant le TouSIX-Manager, une plateforme de contrôle SDN pour un IXP.
- Le troisième test consiste à vérifier l'impact du module SFC sur les règles de flux.

1- Vérification

Ce premier test consiste à nous assurer que les politiques fournies par des membres différents ne rentrent pas en conflit avec des règles déjà établies, pour les règles de production ou les politiques SFC. Plusieurs topologies ont été évaluées. Ces topologies sont des graphes classiques qui décrivent de façon générique les topologies des réseaux réels. Elles permettent de couvrir largement les situations possibles en matière de recherche de chemins optimaux en situations réelles. Pour couvrir ce spectre large, nous considérons donc quatre topologies différentes : hypercube, fat tree, multi-hop et linéaire.
Pour ces topologies, nous avons émulé des membres d'IXP inter-connectés avec les autres membres présents sur la topologie. Chaque membre de l'IXP transmet au module SFC (et donc à l'équipement d'accès) un ensemble de requêtes SFC sous la forme de politiques valides.
Deux vérifications sont à effectuer pour cette validation :

- En premier lieu, il faut vérifier que la connectivité entre les membres est complète, pour

montrer qu'il est toujours possible d'échanger du trafic non classifié dans le module SFC. Il faut également vérifier que les flux non classifiés ne sont pas impactés par le déploiement du chainage SFC.
- Ensuite, pour chaque flux classifié par le module SFC, nous devons vérifier que le chemin qui est suivi dans la topologie est le chemin optimal (sachant qu'il existera par construction toujours au moins un chemin SFC défini sur chaque hôte/switch).

Ces deux vérifications sont effectuées, avec et sans incidents sur la topologie. Chaque incident correspond à ce qui a été défini dans le groupe.

En ce qui concerne l'environnement de simulation, il se compose de deux machines différentes. Le contrôleur fonctionne sur une machine avec un processeur Intel E5-2620 @ 2.00GHz, avec les librairies par défaut de la distribution Linux Ubuntu 16.04.3 LTS. Dans le même temps, le plan de données est installé sur une machine virtuelle ; 4 cœurs virtuels sont alloués sur une machine équipée d'un processeur Intel E5-2699 v3 @ 2.30GHz, avec 8GB de RAM. L'environnement de test représente un IXP typique. Les hôtes sont représentés par des routeurs qui ont des hôtes transmettant du trafic à travers l'IXP. Les instances de services sont gérées à l'aide de conteneurs qui sont créés pour chaque politique.
Le prototype a été créé sur la base du contrôleur Ryu. Tous les switches dans la topologie sont gérés par OpensVSwitch dans sa version 2.5.2.

Après une large campagne de tests, aucun bogue n'a été observé : les politiques SFC sont parfaitement appliquées et nous avons pu vérifier que la connectivité est toujours assurée entre les membres de l'IXP. De plus, même en cas de panne de un ou deux liens, le chemin SFC est toujours le chemin optimal tel que défini dans les groupes. On observe également qu'avec cette approche, en production, le nombre de règles OpenFlow est réduit, à la fois au niveau des commutateurs de bordure que des commutateurs de cœur. Cela va dans le sens d'une plus grande scalabilité de l'IXP.

2. Test de procédure complète sur un IXP

Le principal objectif de cette évaluation est de montrer que le calcul de chemins de secours n'a qu'un impact minime sur le temps de déploiement d'une politique avec le module SFC.
Pour ce test, nous allons nous focaliser sur une seule topologie, celle de TouSIX. Le TouSIX-Manager [39] est mis en place sur le plan de contrôle. Le module SFC a, quant à lui, été développé dans le TouSIX-manager.
L'opérateur de l'IXP peut offrir des services qui fonctionnent sur son infrastructure, sur les switches d'accès, et tous les membres peuvent demander une politique spécifique pour leur trafic.
Pour cette vérification sur la validité du filtrage des politiques, nous utilisons les données de TouSIX sur ce test. Plus précisément, les données de peering établies en production nous servent pour le processus de vérification des routes BGP au niveau du module SFC. Et pour éviter de perturber les services sur un IXP en fonctionnement opérationnel, nous avons réalisé ces évaluations à l'aide de Mininet[3] pour représenter les hôtes et les switches de la topologie TouSIX, avec des scripts pour la génération de trafic préalablement capturé sur TouSIX.
Lors du déploiement des politiques SFC, l'impact du calcul des chemins de secours ne peut même pas être mesuré avec nos outils, tant ce temps est court. Ce calcul des chemins est donc transparent sur le processus de déploiement d'une politique de SFC.

[3] http://mininet.org/

3. Répartition des règles SFC sur la topologie

L'objectif de ce test est de mesurer l'impact de l'utilisation du module SFC sur le nombre de règles installées lors. Nous reprenons la topologie utilisée lors du test précédent, et les requêtes SFC sont directement transférés au module SFC (e.g. pour simplifier, elles ne sont pas transmises par les membres).

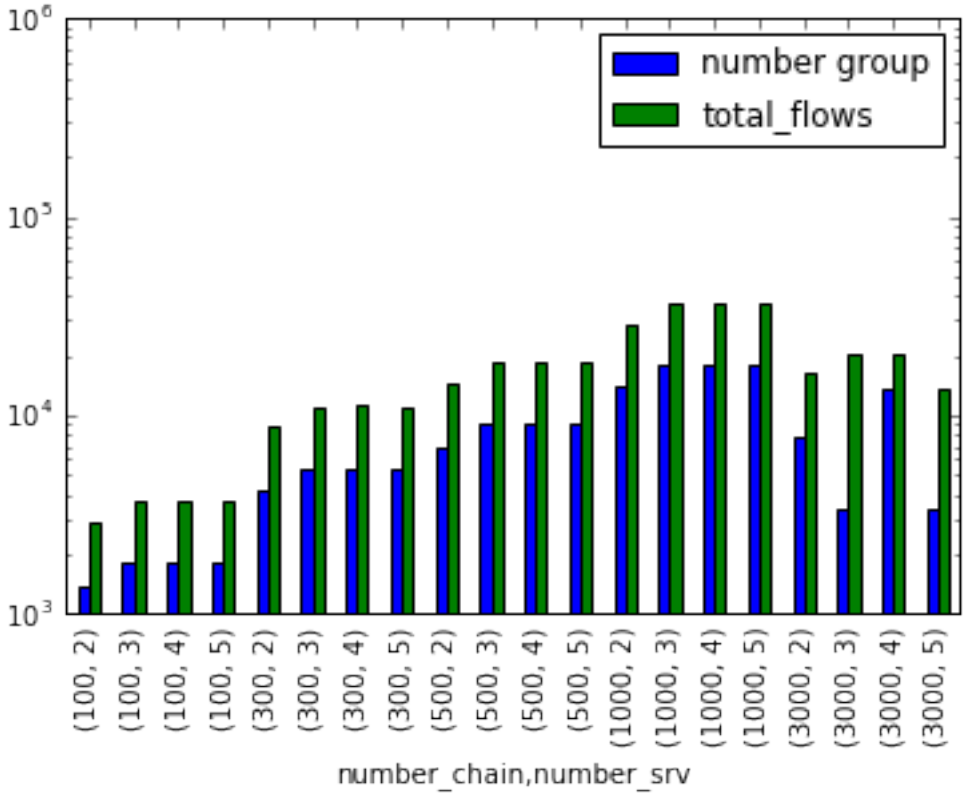

Figure 8. Nombre de groupes/flux déployés dans l'IXP

La figure 8 montre le nombre de règles et de groupes installés sur la topologie pour les requêtes SFC. La figure 8 affiche le nombre de requêtes couplé avec le nombre d'éléments dans le chaînage de service. Le regroupement du nombre de services est dû à l'ajout de règles de flux et de groupes en sortie de service, pour identifier le prochain service, ou proposer une redirection vers le destinataire du flux. Nous remarquons que la courbe évolue de manière quasi linéaire avec le nombre de requêtes envoyées.

5. Etat de l'art

Introduire OF dans le monde des IXP est une idée très récente. Gupta et al. [2], [3] ont développé un point d'échange utilisant SDN (SDX) pour permettre la mise en place de règles plus évoluées que les règles de forwarding hop-by-hop conventionnelles. Cette solution a montré qu'il est possible d'implémenter des polices représentatives pour des centaines de participants tout en garantissant des temps de réponse à des changements de configuration et des modifications de routage inférieurs à la seconde. Pour y parvenir, la version multi-tables

du prototype iSDX considère un scénario dans lequel tous les participants sont connectés à un unique commutateur. En réalité, cependant, il peut y avoir des sauts multiples dans l'infrastructure de l'IXP. Umbrella, avec son approche par commutation de labels, peut servir de support fiable de l'architecture iSDX en forwardant directement les paquets de découverte de la localisation à tous les participants, ce qui rend Umbrella et iSDX complémentaires. En particulier, après qu'iSDX ait pris sa décision quant au port de sortie, l'adresse MAC de destination peut être ré-écrite en utilisant les mécanismes d'umbrella. Les paquets seront alors délivrés en suivant le chemin encodé dans le champ d'adresse MAC destination. Cependant, l'intégration proposée nécessiterait qu'un commutateur OF supplémentaire s'occupe de des requêtes ARP pour la recherche des prochains sauts virtuels et qui vont être pris en charge par un proxy-ARP. Comme Umbrella transforme les paquets broadcast ARP en unicast, les requêtes ARP doivent avoir le chemin vers le proxy-ARP encodé au niveau de l'adresse MAC de destination. Cela améliorerait la flexibilité, car le proxy-ARP n'a pas besoin d'être connecté à un commutateur spécifique de la fabrique de l'IXP.[4]

Le projet cardigan [1] a implémenté en hardware une politique de déni par défaut permettant de filtrer à partir d'une vérification RPKI les routes annoncées par des équipements connectés à la fabrique. Alors que cette approche offre le niveau de protection requis pour une fabrique d'IXP stable, il est moins fiable pour les IXPs qui voudraient rester neutres par rapport aux principes de forwarding du trafic.

L'architecture B4 de Google [31] essaie aussi de régler les problèmes de scalabilité d'OF en forwardant les flux de façon proactive sur le modèle d'Umbrella. Toutefois, l'architecture B4 requiert que le contrôleur traite les décisions de façon active, alors qu'Umbrella est une solution pour les IXPs dont la fabrique ne prend pas part au processus de décision de peering.

Le routage au niveau MAC dans les réseaux OF est un domaine de recherche nouveau. Schwabe et al. [32] ont montré que l'adresse MAC de destination peut être utilisée comme un label universel dans les environnements SDN, et les caches ARP des hôtes peuvent être utilisés comme une table de labels entrants, réduisant ainsi les tables de forwarding des équipements réseaux. Agarwal et al. [33] ont démontré qu'en utilisant les adresses MAC comme des labels de forwarding opaques, un contrôleur SDN peut utiliser les grandes tables de forwarding MAC pour gérer pléthores de chemins grain fin. Même si ces approches présentent des propriétés intéressantes pour les réseaux à grande échelle, elles reposent sur un mécanisme de proxy-ARP, ce qui implique les limitations déjà discutées dans ce rapport.

Concernant le chainage de services se répartissent en deux catégories : (1) les chaines de services transverses dans le plan de données, et (2) celles avec décision de chainage de services dans le plan de contrôle.

- *Chaines de services transverses dans le plan de données* : Les commutateurs dans le plan de données peuvent être conçus pour permettre le chainage de services dans les datacenters des clouds. La commutation à base de politiques a été proposée pour intégrer nombre de fonctions réseau dans les commutateurs déployés dans les

[4] La fonction de proxy-ARP dans iSDX est essentielle pour compresser les flux en classes d'équivalence de forwarding.

datacenters [44].Ce commutateur maintient une table de règles politiques prédéfinies. Quand il reçoit des paquets, le commutateur force les paquets à traverser une séquence de fonctions réseau, puis les envoie vers le saut suivant. Il sépare les fonctions réseau des chemins de routage pour améliorer la flexibilité et les opérations réseau. Cependant, Il n'est pas scalable et manque de résilience/robustesse. Une architecture de sécurité hybride combinant des fonctions réseau hard et des fonctions réseaux implémentées sur des VMs a été proposée dans [45]. Un mécanisme de propagation basé sur des labels dans les fonctions réseau sur VMs permet de déterminer le prochain saut à partir du label encapsulé dans l'entête du paquet. Ces fonctions réseau sur VMs sont scalables pour fournir des chainages de services. Gember et al. [46] ont montré que les fonctions réseau logicielles (sur VMs) distribuées ne sont pas efficaces, et ils ont proposé un plan de contrôle unifié pour gérer les flux de données au travers de fonctions réseau distribuées et chainées. Dans [47], un problème d'optimisation du débit dans les chainages de services et sur différentes topologies a été étudié. Gushchin et al. [48] a proposé un algorithme de routage pour maximiser le nombre de requêtes de service, en prenant en compte les contraintes de mémoire du commutateur et da capacité de calcul.

- *Décision de chainage de services dans le plan de contrôle* : malgré ces travaux sur le chainage de services dans le plan de données, peu ont abordé les problèmes de délai des décisions de chainage de service dans le plan de contrôle. Tootoonchian et al. [49] ont mis en œuvre des benchmarks pour évaluer les performances des contrôleurs OpenFlow disponibles publiquement. Il est mis en évidence que le temps de réponse des contrôleurs affecte de façon significative les délais des SDN. La latence des communications entre un commutateur Openflow et son contrôleur a été évaluée dans [50]. Blenk et al. [51] ont comparé les latences des plans de contrôle de plusieurs architectures d'hyperviseurs SDN et différentes topologies. De façon similaire, une approche mathématique basée sur le Network Calculus pour caractériser le comportement de contrôleurs OpenFlow a été présentée dans [52]. Grâce au Network Calculus, il est possible de calculer la borne supérieure des délais pour les paquets et la capacité nécessaire pour le buffer du plan de contrôle. Même si des travaux ont analysés les délais des plans de données et de contrôle des SDN, ces travaux traitent les questions d'évaluation de délais et de routage séparément. Il existe donc toujours un manque entre la compréhension des limites théoriques des délais et la prise de décisions optimales pour le réseau. Notre approche vise à combler ce vide en proposant une solution qui prenne en considération à la fois le besoin de délais faibles dans les plans de contrôle et de données et la robustesse de niveau de performances élevées. L'originalité tient dans le fait de proposer pour tous les chainages des chemins optimaux en fonction de l'état des liens, réunis sur des groupes avec des commutations immédiates, et un calcul de chemins optimaux reposant uniquement sur la détermination du prochain saut de proche en proche.

6. Conclusion

Scalabilité & fiabilité : Umbrella améliore la scalabilité globale de la fabrique car elle ne nécessite qu'un petit nombre de règles au niveau des commutateurs de cœur et bordure. De plus, cela évite au contrôleur de traiter les paquets de découverte de la localisation, et par conséquent réduit la dépendance entre les plans données et contrôle. Plus avantageux encore,

cela permet au plan de contrôle de continuer à fonctionner même si le niveau de performance du canal de contrôle est faible. De même pour une plus grande flexibilité, le module SFC permet de chainer de nombreuses fonctions réseau de façon optimale et robuste, pouvant également fonctionner dans des conditions très défavorables.

IXP orientés services : La commutation orientée labels des mécanismes de forwarding rend la fabrique de l'IXP orientée services. Un IXP orienté service peut créer des catalogues de ressources réseaux avec leurs politiques de gestion associées (e.g., paramètres de QoS et bande passante) qui peuvent être appliquées aux applications actives dans le réseau. Comme les chemins dans la fabrique de commutation sont configurés au niveau du commutateur de bordure, il est possible de configurer différents chemins selon les applications, ou de rediriger certains flux sur différents chemins suivant les services activés (e.g., firewal, QoS ou monitoring). Notons que c'est seulement une facilité qui peut être activée si besoin. Umbrella peut évidemment être utilisée de façon neutre au niveau de la couche 3 (et au dessus), comme cela est fait traditionnellement aujourd'hui dans les réseaux d'IXP.

TouSIX, un réseau unique au monde : TouSIX est une première européenne et, à ce jour, le seul réseau au monde à exploiter complètement la technologie OpenFlow pour ses opérations quotidiennes Il utilise pour cela un logiciel spécifiquement conçu au LAAS dans ce but : TouSIX-manager (https://github.com/tousix/tousix-manager). Le logiciel permet de configurer automatiquement les règles OF en fonction des besoins des membres, assure la supervision et la métrologie du réseau, et génère les règles de commutation lors de chaque modification du réseau. Il intègre également le module SFC.

9. Références

[1] J. Stringer, D. Pemberton, Q. Fu, C. Lorier, R. Nelson, J. Bailey, C. Correa, and C. Esteve Rothemberg, "Cardigan: SDN distributed routing fabric going live at an Internet exchange," in *ISCC*. IEEE, 2014.

[2] A. Gupta, L. Vanbever, M. Hahbaz, S. Donovan, B. Schlinker, N. Feam- ster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett, "SDX: A Software Defined Internet Exchange," in *SIGCOMM*. ACM, 2014.

[3] A. Gupta, R. MacDavid, R. Birkner, M. Canini, N. Feamster, J. Rexford, and L. Vanbever, "iSDX: An Industrial-Scale Software Defined Internet Exchange Point," in *NSDI*. USENIX, 2016.

[4] B.Ager,N.Chatzis,A.Feldmann,N.Sarrar,S.Uhlig,andW.Willinger, "Anatomy of a Large European IXP," in *SIGCOMM*. ACM, 2012.

[5] N. Chatzis, G. Smaragdakis, J. Bo¨ttger, T. Krenc, A. Feldmann, and W. Willinger, "On the Benefits of Using a Large IXP as an Internet Vantage Point," in *IMC*. ACM, 2013.

[6] H. D. Vu and J. But, "How rtt between the control and data plane on a sdn network impacts on the perceived performance," in *International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE Computer Society, 2015.

[7] K. He, J. Khalid, A. Gember-Jacobson, S. Das, C. Prakash, A. Akella, L. E. Li, and M.

Thottan, “Measuring control plane latency in sdn- enabled switches,” in *Symposium on Software Defined Networking Research (SOSR)*. ACM, 2015.

[8] B.Ager,N.Chatzis,A.Feldmann,N.Sarrar,S.Uhlig,andW.Willinger, “Anatomy of a Large European IXP,” in *SIGCOMM*. ACM, 2012.

[9] N. Hilliard, E. Jasinska, R. Raszuk, and N. Bakker, “Internet Exchange Route Server Operations,” Internet- Draft, Tech. Rep. [Online]. Available: https://tools.ietf.org/html/ draft- ietf- grow- ix- bgp- route- server- operations- 03

[10] P. Richter, G. Smaragdakis, A. Feldmann, N. Chatzis, J. Boettger, and W. Willinger, “Peering at Peerings: On the Role of IXP Route Servers,” in *IMC*. ACM, 2014.

[11] M. Wessel and N. Sijm, “Effects of IPv4 and IPv6 address resolution on AMS-IX and the ARP Sponge,” Master’s thesis, Universiteit van Amsterdam, the Netherlands, 2009.

[12] “FranceIX website,” https://www.franceix.net/en/events- and- news/news/ franceix-outage- notification/, [Online; accessed 29-Jan-2016].

[13] J. C. Cardona and R. Stanojevic, “IXP Traffic: A Macroscopic View,” in *LANC*, 2012.

[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling Innovation in Campus Networks,” *CCR*, 2008.

[15] V. Boteanu and H. Bagheri, “Minimizing ARP traffic in the AMS-IX switching platform using OpenFlow,” Master’s thesis, Universiteit van Amsterdam, the Netherlands, 2013.

[16] I. Pepelnjak, “Could IXPs Use OpenFlow to Scale?” MENOG, 2012.

[17] B. Schlinker, K. Zarifis, I. Cunha, N. Feamster, E. Katz-Bassett, and M. Yu, “Try Before you Buy: SDN Emulation with (Real) Interdomain Routing,” in *ONS*. USENIX, 2014. [Online]. Available: https://www. usenix.org/conference/ons2014/technical-sessions/presentation/schlinker

[18] “Ryu SDN controller,” http://osrg.github.io/ryu, [Online; accessed 29- Jan-2016].

[19] AMS-IX, “Allowed Traffic Types on Unicast Peering LANs,” http://ams-ix.net/technical/specifications-descriptions/allowed-traffic, [Online; accessed 29-Jan-2016].

[20] M. Hughes, M. Pels, and H. Michl, “Internet Exchange Point Wishlist, ”https://www.euro- ix.net/ixps/ixp- wishlist/, 2015, [Online; accessed 29- Jan-2016].

[21] Open-IX, “IXP Technical Requirements OIX-1,” http://www.open-ix.org/standards/ixp-technical- requirements, [Online; accessed 29-Jan-2016].

[22] “OpenFlow Switch Specification,” http://www.openflow.org/documents/openflow-spec- v1.3.0.pdf, [Online; accessed 13-Nov-2017].

[23] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, “Neighbor Discovery for IP

version 6 (IPv6)," Tech. Rep., 2007. [Online]. Available: http://tools.ietf.org/html/rfc4861

[24] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Enabling Fast Failure Recovery in OpenFlow Networks," in *DRCN*. IEEE, pp. 164–171.

[25] N. L. V. Adrichem, B. J. V. Asten, and F. a. Kuipers, "Fast Recovery in Software-Defined Networks," *EWSDN*, pp. 61– 66, 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/ wrapper.htm?arnumber=6984053

[26] "PeeringDB website," http://www.peeringdb.com, [Online; accessed 29- Jan-2016].

[27] A. Lodhi, N. Larson, A. Dhamdhere, C. Dovrolis, and k. Claffy, "Using peeringDB to Understand the Peering Ecosystem," *CCR*, 2014.

[28] B. Augustin, B. Krishnamurthy, and W. Willinger, "IXPs: Mapped?" in *SIGCOMM*. ACM, 2009.

[29] N. Chatzis, G. Smaragdakis, A. Feldmann, and W. Willinger, "There Is More to IXPs than Meets the Eye," *CCR*, vol. 43, no. 5, pp. 19–28, 2013.

[30] "ONOS official website," http://onosproject.org, [Online; accessed 29- Jan-2016].

[31] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Ho ̈lzle, S. Stuart, and A. Vahdat, "B4: Experience with a Globally-deployed Software Defined Wan," in *SIGCOMM*. ACM, 2013.

[32] A.SchwabeandK.Holger, "UsingMACAddressesAsEfficientRouting Labels in Data Centers," in *HotSDN*. ACM, 2014.

[33] K. Agarwal, C. Dixon, E. Rozner, and J. Carter, "Shadow MACs: Scalable Label-switching for Commodity Ethernet," in *HotSDN*. ACM, 2014.

[34] ENDEAVOUR official website, « http:// », [Online ; accessed 7-July-2017]

[35] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making Middleboxes Someone Else's Problem : Network Processing as a Cloud Service."

[36] J. Halpern and C. Pignataro, "Service function chaining (sfc) architecture," Tech. Rep., 2015

[37] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, "Service Function Chaining in Next Generation Networks : State of the Art and Research Challenges," IEEE Communications Magazine, vol. 55, no. 2, pp. 216–223, 2017.

[38] O. Krieger, D. Oran, and R. Fonseca, "Towards a Network Marketplace in a Cloud," USENIX Workshop on Hot Topics in Cloud Computing, 2016.

[39] R. Lapeyrade, M. Bruyère, and P. Owezarski, "Openflow-based migration and management of the touix ixp," in NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, April 2016, pp. 1131–1136.

[40] S. D'Horo, L. Gallucio, S. Palazzo, and G. SChembra, "Exploiting congestion games to achieve distributed service chaining in NFV networks", IEEE J. Selected Areas Commun, vol. 35, No 2., pp 407-420, Feb. 2017.

[41] Cisco data Center Infrastructure 2.5 Design guide. Accessed Dec. 10, 2013. [Online]. Available : http://www.cisco.com/univer-cd/cctd/doc/solution/dcidg21.pdf

[42] I. Ahlad, S. Namal, M. Yianttila, and A. Gurtov, "Security in Software Defined Networks : A survey", IEEE Commun. Surveys Tuts, vol. 17, No 4. Pp 2317-2346, 4th quart 2015

[43] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization : Challenges and opportunities for innovations", IEEE Commun. Mag., Vol. 53, No 2., pp 90-97, Feb. 2015.

[44] D.A. Joseph, A. Tavakoli, and I. Stoica, "A policy-aware switching layer for data centers", ACM SIGCOMM Comput. Commun. Rev., vol 38 ., No 4., pp 51-62, 2008

[45] H.Y. Lam, S. Zhao, K. Xi, and H.J. Chao, "Hybrid security architecture for data center networks", in Proc. IEEE int. Conf. Commun., Ottawa, ON, Canada, 2012, pp 2939-2944.

[46] A. Gember, T. Benson, and A. Akella, "Challenges in unifying control of middlebox traversals and functionality," in *Proc. Large Scale Distrib. Syst. Middleware*, 2012, pp. 1–2.

[47] L. Guo, J. Pang, and A. Walid, "Dynamic service function chaining in SDN-enabled networks with middleboxes," in *Proc. IEEE Int. Conf. Netw. Protocols*, Singapore, 2016, pp. 1–10.

[48] A. Gushchin, A. Walid, and A. Tang, "Enabling service function chain- ing through routing optimization in software defined networks," in *Proc. IEEE Annu. Allerton Conf. Commun. Control Comput.*, Monticello, IL, USA, 2015, pp. 573–581.

[49] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proc. USENIX Workshop Hot Topics Manag. Internet Cloud Enterprise Netw. Services*, San Jose, CA, USA, 2012, p. 10.

[50] K. Phemius and M. B. Thales, "OpenFlow: Why latency does matter," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag.*, Ghent, Belgium, 2013, pp. 680–683.

[51] A. Blenk, A. Basta, J. Zerwas, M. Reisslein, and W. Kellerer, "Control plane latency with SDN network hypervisors: The cost of virtualiza- tion," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 366–380, Sep. 2016.

[52] S. Azodolmolky *et al.*, "An analytical model for software defined networking: A network calculus-based approach," in *Proc. IEEE Glob. Commun. Conf.*, Atlanta, GA, USA, 2013, pp. 1397–1402.