Le contrôleur *Openflow Faucet*

Marc Bruyere
Internet Initiative Japan Institute of Innovation
Chiyoda
Tokyo, Japon

David Delevennat
École Polytechnique
Route de Saclay
91128 PALAISEAU Cedex

Brad Cowie
Waikato University
Hillcrest, Hamilton 3216
Nouvelle-Zélande

Josh Bailey
Vandervecken
1 rue du pont Cybernétique
Nouvelle-Zélande

Résumé

Initié il y a un peu plus de dix ans par des chercheurs, le changement de paradigme induit par le "Software Defined Networking" (SDN) a bien été suivi par l'industrie avec OpenDayLight ou Open Networking Operating System. Mais ces derniers, en dehors de n'être que des frameworks pour développer un contrôleur, n'ont que rarement pu être utilisés pour gérer des réseaux locaux de campus ou d'entreprise.

FAUCET est un contrôleur OpenFlow compact et open source, qui permet aux administrateurs d'exploiter leurs réseaux de la même manière que les clusters de serveurs.

Il est dédié pour les réseaux de campus et d'entreprise. Il n'y a qu'un seul et unique fichier au format YAML[1] pour prendre en charge autant de commutateurs OpenFlow que nécessaire.

Le déploiement de FAUCET ne nécessite aucun développement spécifique. Il possède toutes les fonctions nécessaires pour administrer les réseaux ainsi que les outils de visualisation adaptés. L'outil de tableau de bord Grafana est directement utilisable avec InfluxDB ou Prometheus.

Plusieurs organisations utilisent FAUCET et, depuis plusieurs années, l'université de Waikato à l'initiative de ce projet s'en sert pour opérer une partie de son réseau, constitué de matériel provenant de quatre constructeurs différents. Force est de constater que, malgré les promesses du SDN avec OpenFlow, il ne semblait pas exister

1. un exemple est donné en annexe.

de véritable contrôleur déployable et exploitable sans un gros effort de programmation ; FAUCET comble ce manque.

Cet article rappellera les fondements d'OpenFlow SDN, détaillera l'architecture de FAUCET, expliquera comment le déployer et enfin présentera des retours d'expériences.

Mots-clefs

SoftWare-Defined-Networking, OpenFlow, Réseau Campus, Réseau d'entreprise.

1 Introduction

1.1 Le « software defined networking »

La publication en 2008 de « *OpenFlow : Enabling Innovation in Campus Networks* »[1] a introduit l'idée que les réseaux (à l'origine les réseaux de campus et d'entreprise) peuvent être perçus comme des logiciels flexibles plutôt que comme une infrastructure rigide, permettant un déploiement rapide et sûr de nouveaux services réseau. Le « *Software Defined Networking* » (SDN), c'est avant tout la séparation du plan de contrôle et du plan de données. Cette séparation ouvre de nouvelles perspectives telles que la possibilité de centraliser le contrôle, l'orchestration et la possibilité de virtualisation des ressources physiques. Le SDN basé sur *OpenFlow* offre une flexibilité dans la manière dont le réseau est utilisé, exploité.

Le SDN favorise l'introduction rapide de service personnalisé, car les opérateurs réseau peuvent implémenter les fonctionnalités qu'ils veulent, sans avoir à attendre qu'un fournisseur les intègre dans ses propres produits.

Il réduit les frais d'exploitation, les erreurs et les temps d'arrêt du réseau, car il permet une configuration automatisée de ce dernier et réduit les interventions manuelles à risque.

Le SDN basé sur *OpenFlow* permet la virtualisation du réseau et donc l'intégration du réseau avec les services et le stockage. Cela permet à l'ensemble de l'exploitation informatique d'être gérée de manière plus élégante avec un seul point de vue et un seul ensemble d'outils.

OpenFlow est un standard reconnu qui favorise les marchés ouverts et multi-fournisseurs.

Depuis lors, beaucoup d'opérateurs ont partagé leurs expériences avec le SDN et *OpenFlow* pour une utilisation principalement dans les centres de données et les réseaux à grande distance. Cet article se concentre sur les réseaux d'entreprise et de campus, présentant un contrôleur SDN open-source qui leur est dédié : *Faucet*[2]. Le contrôleur *Faucet* permet de remplacer l'un des éléments les plus basiques du réseau : le commutateur du réseau local avec son plan de contrôle et de données. Il a été conçu pour amener facilement tous les avantages du SDN aux réseaux d'entreprise et campus d'aujourd'hui.

2. En anglais Faucet signifie robinet.

Le SDN permet le développement et le déploiement rapides et sûrs de fonctionnalités réseau grâce à des tests automatisés du matériel et des logiciels, sans avoir recours à des tests manuels fastidieux en laboratoire. Comme décrit plus loin dans cet article, une mise à jour complète du plan de contrôle (permettant par exemple de rajouter une nouvelle fonctionnalité), peut être effectuée en une fraction de seconde même si le réseau est en exploitation.

1.2 Les réseaux campus et entreprise d'aujourd'hui

La plupart des réseaux campus ou d'entreprise se composent de plusieurs couches (classiquement accès, distribution et agrégation ou seulement accès et agrégation) de commutateurs, le plus souvent avec des VLAN séparant les utilisateurs en différents domaines en fonction de leurs activités (par exemple, les ressources humaines de l'ingénierie).

De nos jours une diversité d'appareils et de dispositifs sont connectés[2], nécessitant une exploitation plus complexe incluant la mise en œuvre de politiques de sécurité. L'élaboration des configurations uniques par équipements devient délicate et complexe d'autant plus avec une syntaxe propriétaire. Les commutateurs non-SDN ne sont pas programmables, les règles de sécurité sont définies via un langage de configuration propriétaire à chaque fabriquant. Dans certains cas, un système externe, tel qu'un IDS (Système de Détection d'Intrusion), peut apporter des améliorations grossières à un réseau pour la mise en œuvre d'une politique de sécurité dynamique (par exemple, désactiver un port hôte si l'IDS détermine qu'un hôte sur ce port a été infecté par un malware).

Aujourd'hui, les administrateurs réseaux sont responsables de la gestion et de l'intégration de cette vaste gamme d'appareils et de dispositifs et il peut être difficile de mettre en œuvre une politique de sécurité spécifique si les dispositifs disponibles ne possèdent pas la bonne interface centrale de contrôle. L'exploitation et la maintenance d'un réseau sécurisé avec des outils inflexibles exigent des compétences et des efforts considérables. Les réseaux propriétaires ne pouvant pas être programmés, les possibilités d'automatisation sont limitées. C'est particulièrement vrai lorsque les fabricants n'offrent même pas, au minimum, d'interface de programmabilité ou ne fournissent que des technologies d'automatisation propriétaires qui ne fonctionnent, au mieux, que sur leurs propres équipements.

La sécurité est une préoccupation pour tous les opérateurs et utilisateurs réseaux. Une attaque de type « *zero-day* » sur le réseau lui-même est particulièrement inquiétante, car elle peut avoir un impact sur la sécurité de tous les utilisateurs et services du réseau. Par conséquent, il est essentiel que les opérateurs disposent d'un moyen de réaction rapide. Pour déployer de nouveaux dispositifs de sécurisations ou pour limiter les failles en cas d'attaque, il est nécessaire de ne pas interférer sur le bon fonctionnement du réseau. Le SDN renforce la flexibilité et uniformise le contrôle sur tout le réseau à un niveau très bas. Ce niveau de flexibilité est hors de portée des dispositifs de sécurité externes tels que les pare-feux qui n'ont qu'une vision d'un point unique sur le réseau.

Le design de Faucet prend en compte les contextes particuliers des réseaux campus et entreprise. Dans la section suivante nous présentons *Faucet* et comment le déployer.

2 Présentation de Faucet

Faucet est un contrôleur qui peut être déployé en production en quelques instants sans programmation. Il est utilisé dans des environnements divers, y compris l'*Open Networking Foundation*, qui l'utilise pour son propre réseau local. *Faucet* offre des performances élevées grâce aux commutateurs matériels. La bande passante n'est pas une limitation, car le traitement des paquets est assuré en « *hardware* » et ceci tout en permettant aux opérateurs d'ajouter rapidement et facilement des fonctionnalités. Cela peut se faire sans avoir besoin de changer (ou même de redémarrer) les commutateurs matériels. De plus l'interopérabilité avec les équipements externes non-SDN est assurée car *Faucet* ne programme pas de modifications non standard des champs des paquets.

Faucet a été construit sur la norme *OpenFlow* 1.3 qui est la « *golden* » version de l'*Open Networking Foundation*, la plus adoptée par les constructeurs. Sans la disponibilité d'équipements matériels commerciaux supportant cette norme, cela n'aurait pas été possible. Plusieurs constructeurs fournissent maintenant du matériel qui supporte *OpenFlow* 1.3 avec le support des « multi-tables » et d'IPv6. Afin de minimiser la logique propre aux constructeurs dans le contrôleur, ces derniers ont été encouragés à ne prendre en charge que les éléments obligatoires de la norme *OpenFlow* 1.3. Cela a réduit les coûts initiaux de développement et de support et a simplifié les rapports de bogues et les tests automatisés.

Alors que la technologie SDN continue d'évoluer et de devenir encore plus programmable (par exemple, avec le langage de programmation *P4* cf http://p4.org), *Faucet* et *OpenFlow* 1.3 sont suffisants dès maintenant pour en tirer des bénéfices. Cet article décrit spécifiquement comment tirer parti des pratiques de *DevOps* avec le « *push on green* » pour développer et déployer rapidement des fonctionnalités [3].

2.1 Déployer Faucet

Pour mettre en œuvre *Faucet*, il n'est pas nécessaire de développer mais de le configurer. Cependant, s'arrêter à un déploiement partiel d'un seul équipement de son réseau avec *Faucet* ne produira pas des résultats significatifs. En effet, le remplacement d'un seul commutateur au milieu d'un réseau non-SDN n'aura pas les avantages provenant de la centralisation du contrôleur et de la gestion uniforme de tout le réseau local (notez qu'un contrôleur peut contrôler plusieurs commutateurs).

L'installation[3] [4] de *Faucet* est possible de multiples manières :

- APT "*Advanced Packaging Tool*", le système de *package Debian/Ubuntu* ;
- installer un container *Docker* depuis *Docker-hub* ;
- via le gestionnaire de paquets PIP ;
- installer sur un *Raspberry Pi* ;
- télécharger la machine virtuelle *Faucet* ;

3. https://docs.faucet.nz/en/latest/tutorials/first_time.html

4. https://docs.faucet.nz/en/latest/installation.html

2.2 La redondance

Il est possible d'avoir plusieurs contrôleurs pour un même domaine, ceci pour une meilleure disponibilité et résilience. *Faucet* prend en charge la redondance des contrôleurs. La plateforme d'exécution de *Faucet* peut être de puissance variable selon les besoins. *Faucet* peut être utilisé en production en utilisant une plateforme comme le Raspberry Pi. Par design le contrôleur n'est sollicité qu'à minima. Ce sont les plate-formes matérielles de commutation qui font le travail sur tous les paquets et, de nos jours, elles disposent de toute la puissance nécessaire pour ne pas être bloquantes.

2.3 Rajouter ou corriger une fonctionnalité sans interruption de service

« *Push on green* » fait référence à une philosophie qui consiste à pouvoir créer et tester des logiciels de façon automatisée de telle sorte qu'il soit facile de décider quand les changements de code ou de configuration sont « *green* » et prêt à être déployés avec le moins de bogues possible. *Google* a publié certaines de ses approches pour déployer et gérer des logiciels plus fiables dans un ouvrage récent *Site Reliability Engineering* [4].

Dans cette philosophie, la suite logicielle *Faucet* inclut un module de tests unitaires, permettant de vérifier l'impact de nouvelles fonctionnalités. Les tests peuvent être exécutés aussi bien avec des commutateurs simulés (réseau virtuel *Mininet*) qu'avec du matériel. Ils valident si les modifications sont « *green* » et si l'opérateur peut pousser celles-ci en production avec un maximum d'assurance et de confiance sur le fonctionnement du système. Il est possible de détecter les problèmes de système et d'intégration au moment du développement et de ne pas avoir à mener de campagne de validation très coûteuse au moyen d'un « lab » matériel.

2.4 Comment est traité le « broadcast »

De nombreux commutateurs non SDN implémentent le « *flood unicast* » dans le cadre du processus d'apprentissage, afin que le commutateur puisse découvrir quels hôtes sont connectés à quels ports. L'apprentissage fonctionne de cette façon : un hôte envoie un paquet Ethernet avec une destination inconnue du commutateur et le commutateur « *broadcast* » le paquet à tous les ports dans l'espoir que l'hôte de destination répondra. Cela peut ne pas être souhaitable pour des raisons de sécurité et dans de nombreux commutateurs non-SDN ce comportement est codé en dur.

2.5 Le Multi table OpenFlow de Faucet

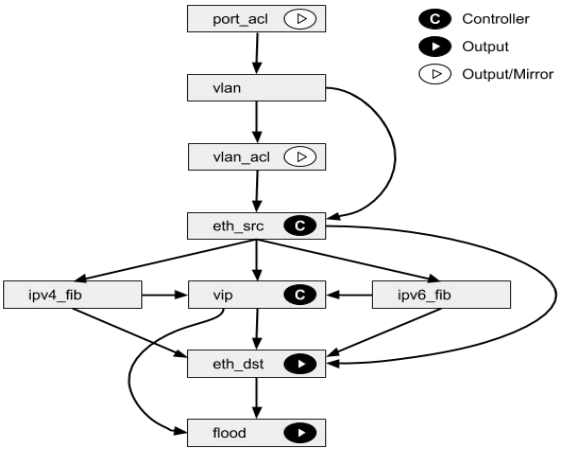

Figure 1 - Faucet « pipeline » multi tables

Toutes les « *pipeline* » *OpenFlow* à table unique sont rapidement limités par le nombre de fonctionnalités [5]. La figure 1 montre le pipeline de Faucet. Toutes les fonctionnalités de *Faucet* utilisent OpenFlow 1.3. Via les multiples tables, il supporte aujourd'hui notamment les VLAN, le routage IPv4 et IPv6 (statique et via le protocole BGP), les ACL (liste de contrôle des accès), le *port mirroring*, le *policy-based forwarding*, le *LACP*, le *802.1X*, *Eduroam*, le « *stacking* » de commutateurs. La liste complète des fonctionnalités est accessible via Internet. Le commutateur matériel se charge de toute la commutation en mode *OpenFlow* et sans aucune fonctionnalité de mode « hybride ». Le mode hybride est le mode dans lequel un commutateur utilise un traitement local non programmable, non SDN, avec un contrôle *OpenFlow*. *Faucet* n'a pas besoin d'un tel traitement local et selon notre expérience, le mode hybride augmente la complexité et limite la programmabilité en introduisant la possibilité d'un conflit entre le contrôle local et *OpenFlow*.

Une infime fraction du trafic est copiée vers le contrôleur pour qu'il puisse apprendre quels hôtes sont connectés sur quels ports et pour que le contrôleur lui-même puisse résoudre les sauts entre VLANs. Le contrôleur est en général inactif sauf si des hôtes sont connectés ou changent de ports (dans ce cas, le contrôleur reprogramme le pipeline comme il convient). *Faucet* dispose d'une sécurité de base contre les attaques à destination du contrôleur (en limitant les adresses MAC Ethernet « *spoofées* »). Comme le pipeline est entièrement programmé par le contrôleur, l'opérateur réseau peut, s'il le souhaite, apporter des modifications arbitraires au comportement du plan de données en modifiant le logiciel du contrôleur.

Lorsque le commutateur matériel démarre, il établit une connexion *OpenFlow* avec le contrôleur. Ce dernier programme le pipeline de base, y incluant les informations relatives aux balises VLAN et aux ACL si elles sont configurées et ajoute des règles de « filtrage par défaut » (tout trafic inconnu est explicitement rejeté). Lorsqu'un nouvel hôte est détecté, le commutateur envoie une copie de l'en-tête Ethernet au contrôleur et (si le « f*lood unicast* » est activé) l'envoie à tous les autres ports du même VLAN ou (si

le « *flood unicast* » est désactivé) l'envoie uniquement si le paquet est un paquet de découverte des hôtes voisins IPv6 ou ARP. Le contrôleur programme ensuite les flux pour que les futurs paquets provenant de cette adresse source Ethernet soient transmis par le commutateur. Ces flux sont périodiquement désactivés et sont réactualisés par le contrôleur si nécessaire, ce qui permet au commutateur de préserver des ressources.

2.6 Le réseau pour le plan de contrôle

Le CPN (*Control Plan Network)* connecte le contrôleur et le commutateur *OpenFlow* via un port dédié. Dans la plupart des déploiements, il s'agit simplement d'un câble court Ethernet. Dans les grands déploiements où un seul serveur exécutant *Faucet* pilote plusieurs commutateurs *OpenFlow*, le câble court peut être remplacé par plusieurs câbles Ethernet et un commutateur non-SDN. La connexion entre le contrôleur et les commutateurs peut être sécurisée via SSL/TLS ou via MACsec (norme de sécurité IEEE 802.1AE MAC.).

De nombreux commutateurs *Openflow* offrent la possibilité de configurer la prise en charge de la perte de la connexion avec le contrôleur via le mode « *fail secure* » (maintien des flux programmés jusqu'à leur expiration) ou « *fail standalone* » (le commutateur passe en mode non-SDN). *Faucet* applique des délais d'expiration à tous les flux, ce qui entraîne la suspension du « *forwarding* ». Si aucun contrôleur ne peut être trouvé pendant une période configurable, alors *Faucet* prévoit que le commutateur sera en mode « *fail secure* ». Il est possible de modifier le CPN (et le commutateur), même en production, sans interruption de service dans le délai d'expiration des flux. Par contre, il n'est généralement pas possible de faire de même avec des commutateurs non-SDN sans interruption de service.

2.7 Monitoring réseau avec Faucet

Figure 2 - Tableau de bord Grafana

Faucet s'écarte de la pratique courante en monitoring réseau en ce qu'il n'implémente pas le protocole SNMP (*Simple Network Management Protocol*). Au lieu de cela, le contrôleur *Faucet* pousse les statistiques (octets, paquets, entrées et sorties de chaque port) vers un système externe. L'approche « *push* » est de plus en plus généralisée et vient même a être considérée comme la bonne pratique. *Faucet* prend en charge

InfluxDB[5] ou *Prometheus*[6] qui sont des bases de données de séries chronologiques open source. En utilisant *Faucet* et un système de visualisation libre, *Grafana*[7], il est possible de créer des tableaux de bord et de lancer des requêtes sur les données actuelles et historiques, comme illustré à la figure 2. *Faucet* peut également produire un journal JSON (*JavaScript Object Notation*) des statistiques qui peuvent être converties et importées dans un autre système.

3 Retour d'expérience : SuperComputing Conference 2018

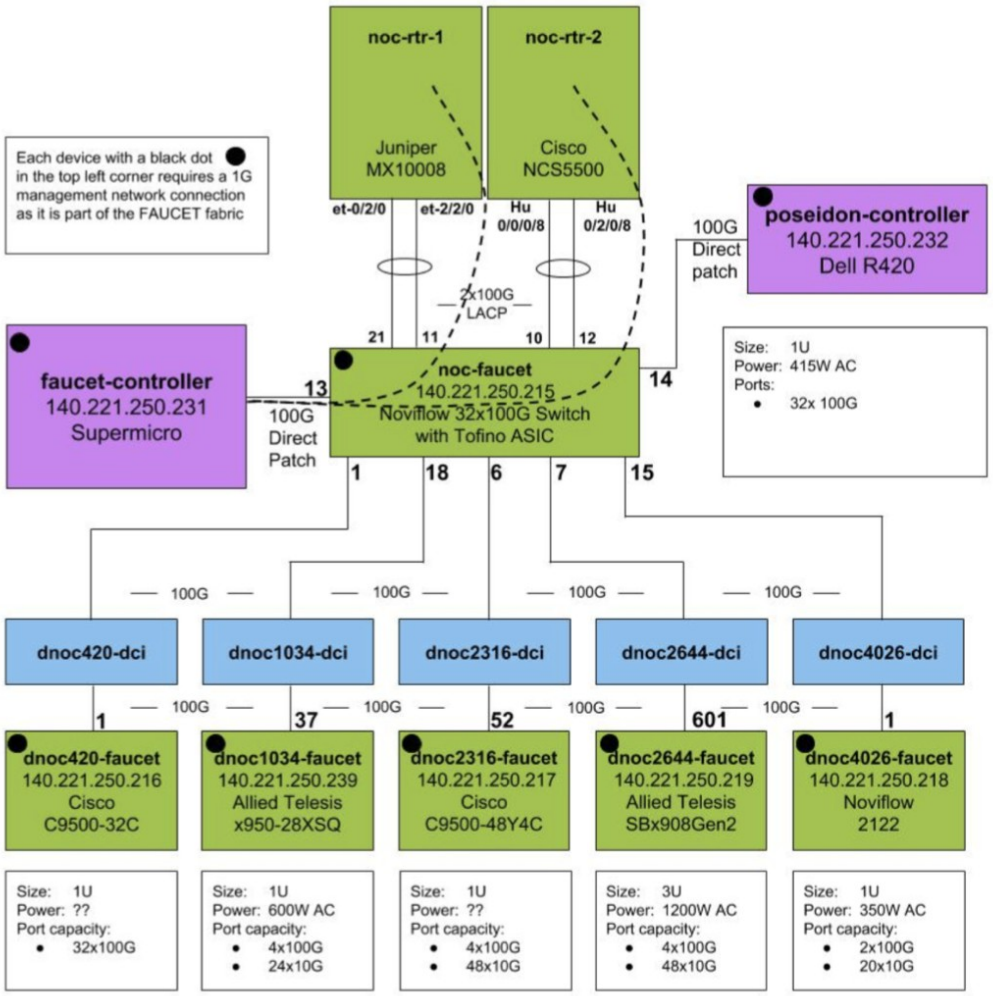

Figure 3 - Schéma du réseau pour SC18

Faucet a été déployé dans de nombreux réseaux, dans le monde entier. Plus récemment, nous nous sommes associés à la conférence SC18 (ACM/IEEE Supercomputing Conférence 2018) pour déployer l'imposant réseau SCinet qui fournit de l'Internet Multi Terabits aux stands sur le salon. Le réseau que nous avons construit comprenait des équipements d'*Allied Telesis, Cisco et NoviFlow* (cf figure 3).

5. https://influxdata.com
6. https://prometheus.io
7. http://grafana.org

Un équipement *NoviFlow* fonctionnant avec un ASIC (*Application Specific Integrated Circuit*) *Barefoot Tofino*, programmable via le language P4, assurait le routage central. *NoviFlow* avait développé en *P4* un plan de données *OpenFlow*, ce qui a permis avec *Faucet* d'utiliser l'API *OpenFlow* pour le piloter. Nous avons aussi écrit un module d'automatisation *Ansible* pour s'intégrer à la base de données de connexion de SC18. Ainsi nous pouvions générer la configuration *Faucet* à la volée. Les opérateurs de SC18 pouvaient, via la base de données, attribuer à chaque stand un VLAN et son adressage IP.

Faucet se chargeait du routage inter-VLAN et BGP (*Border Gateway Protocol*) avec les deux routeurs centraux de SCinet pour l'accès Internet. Avec LACP (*Link Aggregation Control Protocol*) supporté par *Faucet*, nous avons agrégé deux liens 100Gbps vers chaque routeur central afin de permettre une meilleure résilience. Chaque stand bénéficiait d'une connexion Internet soit de 1 Gbps, de 10 Gbps ou de 100 Gbps via son propre commutateur, lui-même contrôlé par *Faucet*. Tous les commutateurs d'accès appliquaient les règles de sécurité par port définie par le contrôleur *Faucet*. Une sécurité supplémentaire était assurée par le logiciel *Poséidon* de *CyberReboot*, un outil d'apprentissage machine, qui s'intègre à *Faucet* et peut identifier les hôtes suspects. Tous les services tels que le protocole DHCP (*Dynamic Host Configuration Protocol*) et BGP, ont été fournis par les solutions open source (dhcpd et BIRD) qui fonctionnaient sur un serveur *Linux*. Le déploiement SC18 a démontré la maturité de *Faucet* et d'*OpenFlow* pour des réseaux de grande taille.

4 Conclusion

Jusqu'à maintenant les avantages du SDN étaient difficiles à accéder en raison d'un manque de logiciels facilement déployable par la communauté des opérateurs réseaux. *Faucet* essaie avec pragmatisme de répondre à ce manque pour les réseaux de type campus ou entreprise. L'automatisation est là pour réduire les temps et les coûts. La gestion centralisée et unifiée multi-constructeurs du SDN de *Faucet* simplifie grandement cette voie.

Les avantages spécifiques du développement logiciel et du déploiement rapide sans interruption de service n'est pas accessible au monde non-SDN. *Faucet* rend possible la sécurisation globale des réseaux de campus ou d'entreprise avec la même flexibilité que celle du développement logiciel.

Annexe

Exemple d’un fichier de configuration au format YAML :

```
vlans:
    office:
        vid: 100
        description: "office network"
        acls_in: [office-vlan-protect]
        faucet_mac: "0e:00:00:00:10:01"
        faucet_vips: ['10.0.100.254/24', '2001:100::1/64',
'fe80::c00:00ff:fe00:1001/64']
        routes:
            - route:
                ip_dst: '192.168.0.0/24'
                ip_gw: '10.0.100.2'
    guest:
        vid: 200
        description: "guest network"
        faucet_mac: "0e:00:00:00:20:01"
        faucet_vips: ['10.0.200.254/24', '2001:200::1/64',
'fe80::c00:00ff:fe00:2001/64']

routers:
    router-office-guest:
        vlans: [office, guest]

dps:
    sw1:
        dp_id: 0x1
        hardware: "Allied Telesis"
        interfaces:
            1:
                name: "h1"
                description: "host1 container"
                native_vlan: office
                acls_in: [access-port-protect]
            2:
                name: "h2"
                description: "host2 container"
                native_vlan: office
                acls_in: [access-port-protect]
```

Bibliographie

[1] : McKeown, Nick and Anderson, Tom and Balakrishnan, Hari and Parulkar, Guru and Peterson, Larry and Rexford, Jennifer and Shenker, Scott and Turner, Jonathan, OpenFlow: Enabling Innovation in Campus Networks, 2008

[2] : Sherry, Justine and Ratnasamy, Sylvia, A Survey of Enterprise Middlebox Deployments}, 2012

[3] : Daniel V. Klein and Dina M. Betser and Mathew G. Monroe, Making “Push On Green” a Reality: Issues & Actions Involved in Maintaining a Production Service, 2014

[4] : Niall Murphy, Betsy Beyer, Chris Jones, Jennifer Petoff, Site Reliability Engineering, 2016

[5] : Open Networking Foundation, The Benefits of Multiple Flow Tables and TTPs, 2008, https://www.opennetworking.org/wp-content/uploads/2014/10/TR_Multiple_Flow_Tables_and_TTPs.pdf