Le SDN pour les nuls

Jérôme Durand
Cisco Systems
11 rue Camille Desmoulins
92130 Issy-les-Moulineaux

Résumé

Le SDN – Software-Defined Networking – est certainement le sujet chaud qui agite le monde du réseau depuis ces dernières années. Le SDN est reconnu aujourd'hui comme une architecture permettant d'ouvrir le réseau aux applications. On est donc loin de la définition rigide initiale dans laquelle il était seulement question de dissocier les plans de contrôle et de données. Openflow n'est donc qu'une composante du SDN et le marché semble aujourd'hui davantage réfléchir sur les solutions offrant réellement la programmation et la simplification des infrastructures. A ce niveau plusieurs modèles de SDN se développent en parallèle pour mieux répondre à des usages spécifiques. Les travaux portent globalement sur la programmabilité intrinsèque des équipements (nouveaux composants programmables…), les contrôleurs SDN et orchestrateurs qui ont pour mission de fournir une couche d'abstraction du réseau, et la virtualisation des fonctions réseau qui vise à s'affranchir partiellement de la complexité du réseau physique sous-jacent. Le Software Defined Networking offrira de nouveaux usages. L'enjeu pour les administrateurs réseau est d'accompagner cette nouvelle étape afin de pouvoir tirer profit de ces nouvelles capacités.

Mots-clefs

SDN, API, Contrôleur, Openflow, Netconf, Overlay, NfV, FOG, Orchestration

1 Introduction

Le SDN – Software-Defined Networking – est certainement le sujet chaud qui agite le monde du réseau depuis ces dernières années. Il est impossible d'avoir un article, un tweet, un blog, une publication ou une conférence sans ces 3 lettres qui semblent vouloir tout solutionner sur nos infrastructures. Dans une telle agitation, les administrateurs réseau sont le plus souvent perdus. Qu'est-ce que le SDN ? Qu'est-ce qu'OpenFlow ? Qu'est-ce que cela apporte pour moi ? Est-ce une technologie qui restera dans les laboratoires ? N'est-ce pas une technique des constructeurs pour me faire acheter leurs derniers commutateurs ?

Cet article fait une présentation simple du SDN, afin de permettre à chacun de mieux comprendre cette évolution actuelle des réseaux, et de se faire sa propre opinion en toute connaissance de cause.

2 Une définition pour le SDN ?

SDN signifie littéralement Software-Defined Networking, c'est-à-dire le réseau défini par l'application. On comprend donc immédiatement que le sujet est vaste et qu'il va être difficile d'avoir une définition unique. De plus, la définition même du SDN a changé avec le temps.

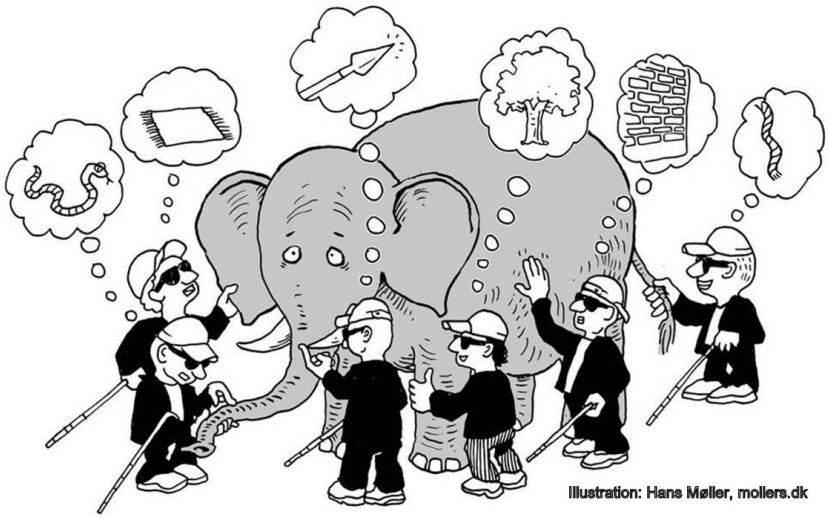

Figure 1 - Plusieurs définitions pour le SDN

2.1 La définition initiale

La définition académique, qui a depuis largement évolué, consistait à voir le SDN comme une architecture qui découplait les fonctions de contrôle et de transfert des données du réseau (*data plane*) afin d'avoir une infrastructure physique complètement exempte de tout service réseau. Il est possible de lire sur le site de l'open networking foundation la définition suivante :

« *Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services* ».

Dans ce modèle, les équipements réseau se contentent d'implémenter des règles, injectées par les applications, de traitement des flux de données. Plus besoin d'avoir sur ces équipements de protocoles de routage, spanning-tree, etc. : une entité intelligente, appelée « contrôleur » voit le réseau dans sa globalité et injecte directement les règles de traitement des données sur chaque équipement.

2.2 Un sujet beaucoup plus vaste

Assez rapidement, le sujet est devenu beaucoup plus vaste. La raison principale est qu'en soit un changement d'architecture n'intéresse personne si ce ne sont les experts réseau et les équipes de recherche et développement. Le marché s'est rapidement approprié le SDN comme ensemble de solutions/architectures permettant de supprimer les frontières existantes entre les mondes des applications et du réseau. Alors que le déploiement d'applications est toujours plus aisé et dynamique, notamment grâce à la virtualisation et au Cloud, le réseau reste parfois perçu comme un frein. Ce dernier ne semble pas avoir évolué et chaque modification nécessite encore souvent des interventions manuelles sans orchestration dans la plupart des organisations. La mise en place d'un simple VLAN peut parfois prendre plusieurs jours, avec la nécessité de configurer un par un de nombreux équipements : commutateurs, routeurs, firewalls…

La discussion a donc naturellement dévié sur les réelles problématiques réseau : simplifier le déploiement d'applications sur le réseau, mettre en place des politiques de sécurité / QoS homogènes globalement pour les applications, automatiser des fonctions réseau directement à partir des applications...

Le SDN est donc plus globalement reconnu aujourd'hui comme une architecture permettant d'ouvrir le réseau aux applications. Cela intègre les deux volets suivants :

- permettre aux applications de **programmer le réseau** afin d'en accélérer le déploiement ;
- permettre au réseau de mieux **identifier les applications** transportées pour mieux les gérer (qualité de service, sécurité, ingénierie de trafic...).

Il n'y a donc pas un seul SDN avec une solution ou un protocole unique. Chacun aura à cœur de se faire sa propre définition selon ses problématiques et ses éventuels intérêts. Cela peut sembler complexe mais c'est la réalité aujourd'hui et il est important de bien comprendre cela pour survivre dans la jungle médiatique du SDN : annonces marketing faites par les constructeurs, articles dans de nombreux blogs, tweets... Martin Casado, qui est reconnu aujourd'hui comme l'inventeur du SDN, pouvait déclarer dans un article de Enterprise Networking Planet, (Kerner, 2013) : "*I actually don't know what SDN means anymore, to be honest*". Cela résume assez bien la situation : ne perdons pas notre temps à nous demander ce qu'est le SDN, mais concentrons-nous sur les problématiques à résoudre.

2.3 Pourquoi maintenant ?

Avoir un réseau automatiquement programmé est une des nombreuses légendes bien connues par les administrateurs système et réseau. La première fois, on écoute un peu méfiant en se demandant si notre avenir est menacé, et les fois suivantes, on n'écoute même plus : « le réseau est tellement complexe et critique, impossible de faire autrement... »

Mais les choses ont changé…

Tout d'abord les applications et le réseau jouent un rôle toujours plus critique pour l'entreprise. Ils ne sont plus seulement là pour accompagner les métiers (ERP, portails de vente en ligne…) Ils sont au cœur même du métier de chaque organisation qui aujourd'hui réfléchit plus que jamais à sa stratégie de « digitalisation ». Les métiers « traditionnels » se font largement concurrencer par des modèles disruptifs utilisant des applications innovantes ; par exemple Uber® concurrence les taxis, Airbnb® met à mal les hôtels, Youtube® retransmet les matchs de football de la Ligue 2 et vient ainsi marcher sur les plates-bandes des chaînes de télévision… Chaque organisation prend donc sa « digitalisation » avec beaucoup plus de sérieux et la mise en place d'applications évolue beaucoup plus rapidement. Chacun se développe pour éviter de se faire « Uberiser », terme à la mode assez explicite.

Dans ce contexte les réseaux traditionnels montrent encore davantage leur faiblesse. Même quand aucune application n'est à déployer localement (utilisation de services dans le cloud), il faut bien que le réseau assure la connexion entre les usagers et celle-ci, avec les niveaux appropriés de qualité de service et de sécurité. Cela peut prendre beaucoup de temps et la situation n'est souvent plus tenable. Cette pression est devenue trop forte pour les équipes réseau et c'est la raison pour laquelle le SDN est pris au sérieux aujourd'hui. Dans ce contexte certains experts vont même jusqu'à estimer que la CLI traditionnelle sur les équipements réseau (Command Line Interface – paramétrage en lignes de commande au format texte) aura quasiment disparu dans 5 ans !

Les chiffres suivants démontrent bien que l'engouement pour le SDN n'est pas juste une mode ou un simple intérêt technologique :

- 4800 % de croissance des investissements depuis 2007
- 419 entreprises se proclament comme fournissant des solutions « SDN »
- Marché estimé à 35 milliards de dollars pour 2008

Il y a une réalité SDN et cela conforte les analystes dans leurs prédictions : le métier d'administrateur réseau devrait prochainement changer.

2.4 Le SDN est donc différent d'Openflow ?

La plupart du temps, il y a une confusion entre SDN et Openflow, comme rappelé dans l'article « The Road to SDN: An Intellectual History of Programmable Networks » traçant l'histoire du SDN : *« A commonly held misconception is that SDN and OpenFlow are equivalent; in fact, OpenFlow is merely one (widely popular) instantiation of SDN principles »*

Aussi, il est important de bien comprendre dès à présent la différence entre les deux. Dans l'architecture SDN initiale, expliquée en section 2.1, des règles de traitement des flux de données sont injectées par un contrôleur intelligent dans des équipements réseau sans fonction de contrôle. Openflow est le protocole défini par l'ONF (Open Networking Foundation) pour transférer ces règles. Il permet par exemple à un contrôleur d'injecter des règles sur des commutateurs ou des routeurs.

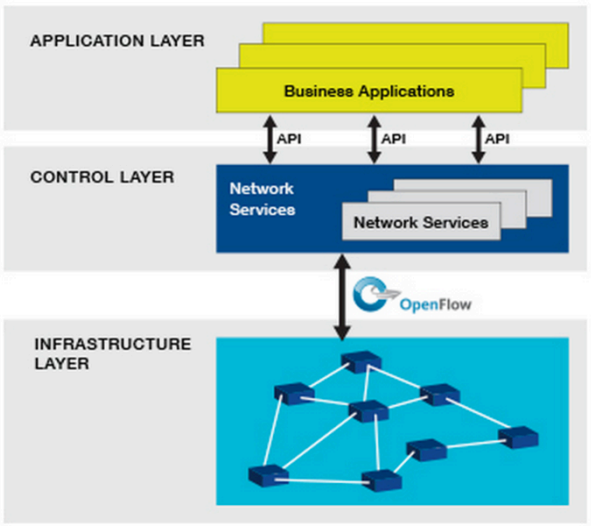

Figure 2 - Openflow dans l'architecture SDN

Openflow transmet des ordres au niveau du plan de données, par exemple :

- Transférer les trames à destination de l'adresse MAC X vers l'interface Y en rajoutant le VLAN Z dans l'en-tête 802.1Q
- Jeter les paquets ayant pour source ou destination l'adresse IPv6 A
- Encapsuler en GRE à destination de l'adresse IPv4 C tous les paquets IPv4 provenant de l'adresse D

Comme cela a été clarifié précédemment, le SDN va au-delà de la capacité d'injecter des règles, et donc des états sur les équipements pour chaque action nécessaire. Openflow en soi ne suffit donc pas à répondre aux problématiques réelles des organisations. Aussi, cela pose de nombreuses questions quant à la capacité de résister au facteur d'échelle quand le nombre de règles/flux s'accroit du fait de l'augmentation du nombre d'utilisateurs, de terminaux et d'applications.

Openflow est donc une composante du SDN mais c'est loin d'être la seule puisque cette technologie ne répond pas directement aux problématiques rencontrées. Rob Sherwood, CTO de Big Switch Network, acteur historique sur le marché du SDN avec une solution de contrôleur Openflow, déclarait (Sherwood, 2014) : "*Openflow is assembler language for networking and that's definitely not for everyone*". Cela illustre bien qu'Openflow est un outil au milieu de nombreux autres.

Depuis 2013, on constate d'ailleurs assez peu d'avancées sur Openflow. Il semble que l'ONF se cherche une nouvelle orientation à travers des sondages envoyés à la communauté. Mike Robuck, rédacteur senior sur SDxCentral.com, apporte l'anayse suivante :

« When the ONF was founded four years ago, the OpenFlow standard that it supports was pretty much the genesis of software-defined networking (SDN). Since then, other open source organizations and implementations have come into play in support of SDN. In short, OpenFlow is no longer the big player that it once was »

2.5 Modèle de programmation impératif ou déclaratif

Dans un modèle impératif, un programme connaît exactement toute la suite des actions à réaliser pour achever une action. Appliqué au SDN, cela revient à devoir injecter sur l'ensemble des équipements tous les états nécessaires pour réaliser une action.

Dans un modèle déclaratif, le programme se focalise sur l'objectif. Chaque entité appelée par le programme (un élément réseau dans le cas du SDN) va faire tout le nécessaire pour arriver à cet objectif en reposant éventuellement sur une décision locale.

Ce second modèle semble aujourd'hui gagner du terrain sur le SDN pour de nombreuses raisons :

- il est plus simple à mettre en œuvre puisqu'on va utiliser une intelligence déjà présente sur les équipements ;

- il est plus robuste puisque les équipements réseau eux-mêmes peuvent réagir. Prenons simplement le cas d'un protocole de routage : une décision prise localement sur un routeur sera toujours plus rapide que reposer sur un ordre transmis par un contrôleur sur le réseau qui peut justement subir une panne ;
- il change moins les habitudes des équipes réseau puisqu'on va s'appuyer sur des bases maîtrisées. Cet élément est décisif puisqu'une solution ne peut être déployée en production que si des personnes sont capables de l'opérer et de résoudre des problèmes rapidement quand ils surviennent ;
- il est préféré par les constructeurs qui acceptent plus facilement d'exposer l'intelligence de leurs équipements s'ils restent complètement maîtres de leur technologie.

Ce mode déclaratif requiert la modélisation du réseau et de ses services. On va parler de « policy-models » qui peuvent être décrits via des langages comme YANG.

3 Différents modèles pour le SDN

Comme expliqué dans la section précédente, le SDN englobe toutes les solutions permettant une programmation du réseau, afin de mieux interagir avec les applications. Diverses solutions coexistent, adaptées selon les besoins des utilisateurs. On comprend aisément que les solutions permettant de simplifier le déploiement d'une application dans un datacenter ne sont pas les mêmes que celles qui permettront de mieux contrôler l'éclairage d'une ville à travers le réseau.

On peut noter différents modèles de programmabilité :

1) Programmabilité individuelle de chaque équipement. Dans ce modèle une application interagit directement avec chaque équipement via des API. L'application est centralisée ou peut être localisée directement sur l'équipement réseau pour réaliser des tâches spécifiques.

2) Programmabilité via un contrôleur. Dans ce modèle, une application donne un ordre abstrait et global à un contrôleur, qui à son tour traduit cette requête en une suite d'ordres auprès des équipements du réseau concerné. Ce modèle est certainement le plus populaire puisqu'il permet de simplifier le réseau. Le contrôleur masque la complexité du réseau. On peut distinguer plusieurs cas selon le type d'ordres échangés entre le contrôleur et les équipements. Si, au départ, il était question d'avoir une programmabilité du plan de données (via Openflow par exemple), des modèles plus récents implémentent des modèles dans lesquels des ordres plus abstraits sont donnés aux équipements, ces derniers restant libres de les implémenter au mieux. On parle dans ce cas d'un modèle « policy-intent ». La suite de l'article reviendra sur ces différentes alternatives.

3) Création d'un réseau virtuel au dessus du réseau physique. Dans ce modèle, les applications créent leur propre réseau « overlay », s'affranchissant des contraintes du réseau physique sous jacent. Ce dernier n'a pour mission que la simple connectivité entre les nœuds d'extrémité des tunnels, et le réseau d'overlay assure l'intégralité des services. On parle également de virtualisation des fonctions réseau (NFV – Network Function Virtualization) quand les routeurs, commutateurs, firewalls, etc. sont des éléments virtualisés sur des serveurs.

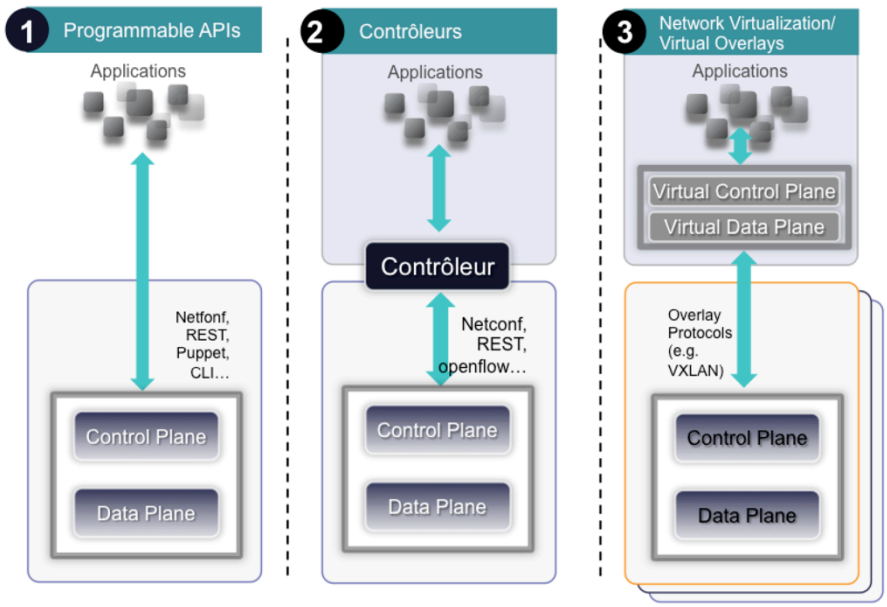

Figure 3 - Modèles SDN principaux

Cette section liste les diverses facettes du SDN succinctement afin de guider le lecteur avant des recherches poussées.

3.1 La programmation des équipements

3.1.1 Les API

La programmation des équipements réseau nécessite sur ces derniers la capacité de recevoir des directives de l'extérieur. Pour cela des interfaces de programmation sont nécessaires : des API (Application Programming Interface). Il existe de nombreuses API, standards ou propriétaires, pouvant agir sur différents éléments de l'équipement (plan de données, plan de management…) Il est communément admis qu'une seule API universelle ne suffira pas pour résoudre toutes les problématiques réseau. Aussi les équipements modernes implémentent souvent plusieurs API.

La suite de cet article décrit quelques API les plus communément supportées sur les équipements réseau :

- CLI

L'accès en ligne de commande aux équipements via telnet/ssh est bel et bien une API et il est important de ne pas l'oublier ! C'est d'ailleurs encore celle qui est encore le plus souvent utilisée pour programmer/configurer le réseau. La CLI agit sur le plan de management.

- OpenFlow

OpenFlow est une API permettant la programmation du plan de données. Des actions sont programmées au niveau des flux (un ensemble de critères sur les paquets/trames : par exemple IP source, MAC source, DSCP…) Tout flux correspondant à une entrée dans la table OpenFlow de l'équipement sera traité selon les actions demandées.

- Netconf/YANG

Netconf est un standard IETF (RFC 6241) visant à standardiser les directives réseau auprès des équipements. YANG est un langage permettant la modélisation des services réseau. Une directive Netconf pourra contenir un modèle YANG et ainsi configurer de la même manière des équipements de constructeurs différents, si tant est qu'ils implémentent bien le service demandé et cette API de configuration. Ce modèle standardisé est le plus souvent privilégié par les *service providers* qui cherchent au maximum à être indépendants des constructeurs, sans pour autant perdre la maîtrise du réseau.

- RESTful API (REpresentational State Transfer)

On parle de RESTful API quand les directives fournies sont faites via des directives de type HTTP (GET, POST, DELETE…) La principale caractéristique d'une API RESTful est qu'elle est sans état. En outre, chaque requête correspond à une demande. Il n'y a pas de dialogue possible entre les extrémités de la chaîne via une connexion

préalablement établie. On dénote l'API RESTconf qui permet d'échanger via une API RESTful un modèle YANG. Ces API sont plutôt privilégiées par les communautés de développeurs web et seront plus souvent utilisées dans le contexte d'un datacenter puisque cela correspond au language naturel de programmation des équipes en charge des applications

- Opflex

Opflex est un protocole permettant à un *Policy Repository* (PR) de dialoguer avec des *Policy Elements* (PE) pour leurs demander de réaliser une action abstraite. Il est adapté à un modèle de programmation déclaratif puisqu'on va se focaliser ici sur un objectif. Opflex intègre une boucle de feedback permettant à un équipement réseau de fournir des informations clés au *Policy Repository* afin de l'aider à prendre les bonnes décisions. Ce protocole est en cours de standardisation à l'IETF. Il est davantage appliqué au niveau du datacenter puisque c'est aujourd'hui sur cet écosystème qu'il a été déployé par divers constructeurs.

- Autres…

Il existe de très nombreuses autres interfaces de programmation, plus ou moins ouvertes et adaptées à des environnements spécifiques. Nous noterons dans le désordre et sans explication particulière : OVSDB, PCEP, BGP-LS, Python API, SNMP, LISP, CAPWAP, HTTP, OVSDB…

3.1.2 Quelle intelligence dans les équipements ?

Une des premières idées qui a émergé lors des débuts du SDN était que les équipements réseau allaient perdre toute intelligence, ces derniers se contentant d'exécuter des ordres émis par les applications et les contrôleurs, entités surdouées capables de tout faire. Quelques années plus tard, on observe que presque personne ne conserve cette vision. Le contrôleur est avant tout un facilitateur pour répondre à de nouveaux besoins plus rapidement et il permet de simplifier globalement le réseau. Chacun conçoit que chaque équipement réseau doit conserver une intelligence locale non seulement pour réaliser des actions qu'il fait bien aujourd'hui, et ce de manière distribuée (routage, spanning-tree, agrégats de lien, fonctions de sécurité, etc.) ; mais aussi pour implémenter les directives complexes transmises par les applications.

Effectivement programmer le réseau n'est possible que si chaque équipement offre une certaine dose de programmabilité. Or, les équipements réseau sont très divers. Alors que certains ne doivent gérer que quelques kilobits par seconde pour des applications spécifiques (distributeurs de billets, capteurs divers…), d'autres routeurs sur les cœurs de réseau d'opérateurs doivent commuter plusieurs terabits par seconde. Les différences entre les performances requises sur ces matériels imposent des architectures radicalement différentes. La plupart des constructeurs intègrent et/ou développent des ASICs (Application-Specific Integrated Circuits) permettant d'assurer les performances à coût relativement bas. Comme leur nom l'indique, ces composants sont définis pour réaliser des tâches prédéfinies et ne sont donc pour la plupart pas programmables. Ils sont généralement un frein à l'innovation puisque chaque nouvelle fonctionnalité non prévue nécessite ce que l'on appelle un « re-spin », c'est-à-dire le développement d'une nouvelle version de l'ASIC. Certains systèmes combinent un ASIC pour les opérations fixes qui délègue certaines fonctions complexes à des FPGA (Field Programmable Gate Array) ou processeurs plus coûteux. Dans ce cas aussi, il convient à l'avance d'anticiper ce qui doit être envoyé à ces éléments intelligents.

Aussi certains équipementiers ont créé des Network Processing Units (NPU) introduisant une dose de programmabilité tout en conservant les performances réseau attendues. Ces composants, globalement plus coûteux que les ASICs, peuvent être reprogrammés pour implémenter les dernières innovations, comme par exemple un nouveau type d'encapsulation. Des ASICs programmables sont également apparus sur le marché. De leur côté, les processeurs x86 ont aussi évolué pour mieux gérer les paquets ; ils peuvent assurer des performances toujours plus élevées, ouvrant la voie à la virtualisation des fonctions réseau (ou NFV) qui sera décrite dans la suite de cet article.

Nous pourrons retenir les 2 éléments suivants :

- les approches SDN n'ont pas pour objectif d'éradiquer tout plan de contrôle/management sur les équipements réseau. Au contraire elles cherchent à mieux les exploiter ;
- la programmation du réseau impose de nouvelles contraintes sur le plan de données des équipements réseau qui sont amenés à réaliser des actions plus complexes.

3.1.3 P4 (Programming Protocol-Independent Packet Processors)

Ce récent projet regroupant divers académiques et industriels illustre bien les conclusions de la section précédente car il vise la programmation complète du traitement des données sur un équipement réseau. Au lieu de programmer des états

comme le fait Openflow, on va réellement programmer via un langage abstrait baptisé P4 le comportement d'un équipement réseau. Un algorithme de traitement d'un paquet IPv4 pourra être simplement construit. Le format même des paquets est simplement défini dans le programme et rajouter/modifier un en-tête relève d'une simplicité enfantine. L'exemple ci-dessous montre comment analyser (parser) une trame Ethernet. Il suffit de quelques lignes pour définir un nouveau format de données.

```
parser ethernet {
  switch(ethertype) {
    case 0x8100: vlan;
    case 0x9100: vlan;
    case 0x0800: ipv4;
    case 0x86dd: ipv6;
  }
}
```

Une fois le programme créé, il suffit de le compiler pour qu'il puisse fonctionner sur le matériel désiré (ASIC, FPGA, Network Processor, CPU…). A nouveau, le degré de programmabilité du matériel est la clé pour réaliser les actions demandées avec les performances requises.

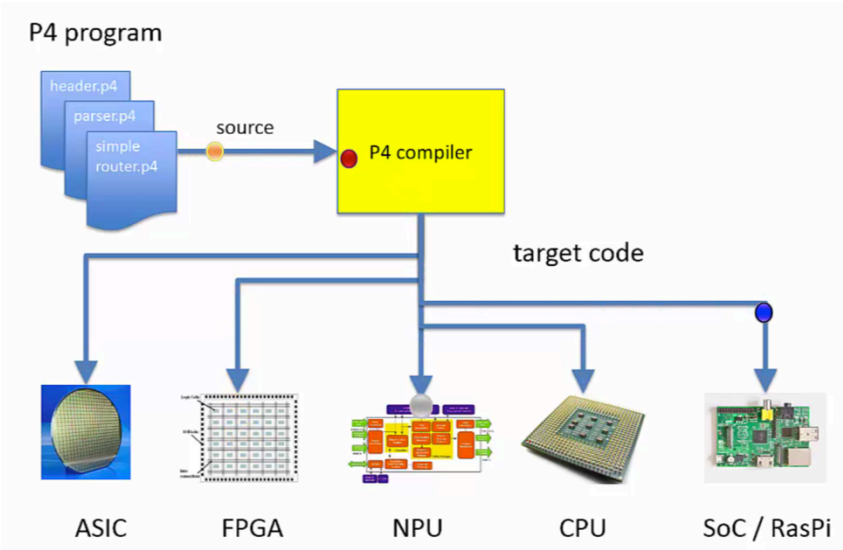

Figure 4 - Compilation de code P4 sur différents matériels

3.2 Les contrôleurs SDN

3.2.1 Définition

Certaines solutions SDN reposent sur la mise en place de contrôleurs. Ces derniers ont pour mission de fournir une couche d'abstraction du réseau et de présenter ce dernier comme un système. Le contrôleur SDN permet d'implémenter rapidement un changement sur le réseau en traduisant une demande globale (par exemple : prioriser l'application X) en une suite d'opérations sur les équipements réseau (requêtes Netconf, ajouts d'états Openflow, configuration en CLI…)

Les ordres sont donnés au contrôleur par une application via une API dite « Northbound » ou nord. Les éditeurs logiciels de contrôleurs publient la documentation de l'API afin de permettre d'interfacer des applications.

Le contrôleur communique avec les équipements via une ou plusieurs API dites « Southbound » ou sud. Openflow se positionne comme une API sud agissant directement sur le plan de données. D'autres API permettent d'agir sur le plan de management ou de contrôle. Netconf est par exemple une API sud permettant au contrôleur de configurer un équipement. Un contrôleur pourra même parler directement en CLI avec un équipement pour actionner une fonctionnalité.

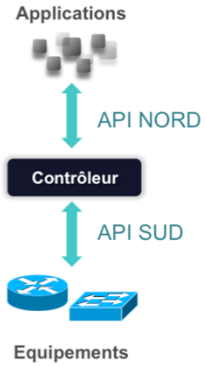

Figure 5 - APIs Nord et Sud

Afin de pouvoir interagir avec le réseau, le contrôleur a besoin d'une vue précise de ce dernier. C'est ainsi que le concept de NIB (Network Information Base) a vu le jour. Cette NIB est construite au niveau du contrôleur et permet à ce dernier de savoir comment implémenter chaque ordre abstrait, trouver les équipements qui doivent être reconfigurés, s'assurer de la capacité de ces derniers à implémenter une directive, les API supportées par l'équipement…

La plupart des équipementiers principaux travaillent aujourd'hui sur des contrôleurs SDN. Certaines start-ups, la plus connue étant sans doute Big Switch Network, se sont créées pour se focaliser sur cette tâche.

3.2.2 Contrôleur SDN ou outil de management ?

Le contrôleur pouvant agir sur le plan de management, la frontière avec les outils de management traditionnels peut s'avérer bien mince.

Nous noterons les principales différences suivantes :

- Le contrôleur est conçu pour exécuter un changement sur le réseau. Il n'a pas pour objectif principal d'assurer une traçabilité du réseau ni de vérifier de l'état de santé de chaque nœud. L'outil de management en revanche a pour première mission de vérifier que le réseau fonctionne correctement et générer des alarmes en cas de besoin. Afin de bien mettre en évidence cette première distinction essentielle, nous conviendrons qu'un outil de management restera indispensable, ne serait-ce que pour s'assurer que le contrôleur garde le réseau dans un état opérationnel !
- Le contrôleur agit tant sur le plan de management que sur les plans de contrôle et de données. L'outil de management reste exclusivement sur le plan de management (configuration de l'équipement).
- Dans un modèle SDN à base de contrôleur, il y a un découplage entre l'application, qui donne un ordre abstrait, et le contrôleur, qui va implémenter cet ordre au niveau du réseau. Dans un outil de management, il n'y a pas de dissociation entre ces 2 éléments. L'outil de management peut très bien évoluer pour devenir une des applications utilisant les services d'un contrôleur. Dans ce cas, l'outil de management se concentre sur la GUI et des opérations globales abstraites, et laisse le/les contrôleur(s) réaliser les changements sur le réseau.
- Un contrôleur est ouvert puisqu'il est là pour exécuter des ordres fournis via une API Nord. Les outils de management ne sont pas toujours ouverts et ne peuvent réaliser que les missions que pour lesquelles ils ont été conçus. La mission du contrôleur va donc bien au-delà du management puisque les applications peuvent être très variées : gestion de la QoS par un serveur de téléphonie, load balancing contrôlé directement par l'application, auto-configuration…

3.2.3 Open Daylight

Alors que le consortium ONF (Open Networking Foundation) s'est penché sur l'API Openflow permettant le contrôle du plan de données, le consortium Open Daylight a été créé pour se pencher sur la problématique du contrôleur, qui est un élément prépondérant de nombreuses architectures SDN. L'objectif de ce consortium est de regrouper les efforts d'industriels et d'académiques pour développer un contrôleur réellement utilisable dans des environnements de production (au-delà des quelques laboratoires de recherche ou de sociétés dans le domaine de l'informatique ou le réseau). Ce projet collaboratif, placé au sein de la Linux Foundation, a permis le développement d'un contrôleur, aujourd'hui dans sa troisième version :Lithium.

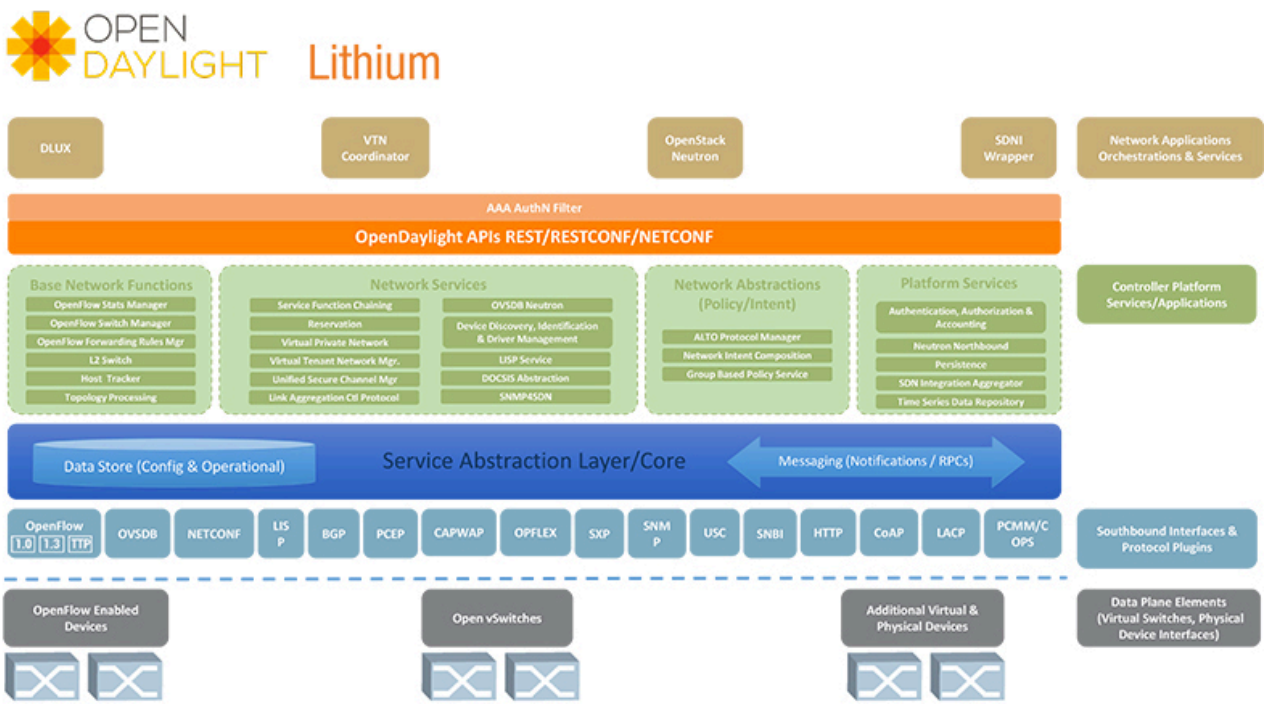

Figure 6 - Architecture du contrôleur Open Daylight (version Lithium)

Nous pouvons noter la quantité d'interfaces sud (Openflow, OVSDB, Netconf, BGP, PCEP, SNMP, HTTP, Opflex…) mettant en évidence qu'une seule API focalisée sur le dataplane ne suffit pas quand il est nécessaire d'assurer de nombreux services. Pour assurer l'intégration simple d'une nouvelle API sud, une couche d'abstraction appelée SAL (Service Abstraction Layer) a été conçue et reste un élément clé du contrôleur Open Daylight, permettant un développement de nouveaux services sans lien direct avec les API utilisées pour les implémenter.

3.3 La virtualisation des fonctions réseau (NFV)

NFV (Network Function Virtualization) est une approche consistant à réaliser certaines fonctions réseau, traditionnellement effectuées sur du matériel dédié, sur des serveurs x86. C'est une facette importante du SDN particulièrement étudiée chez les fournisseurs de services qui voient ici une solution pour mieux ajuster l'investissement selon les besoins de leurs clients. Au lieu de déployer du matériel ad-hoc pour chaque demande, l'opérateur va piocher dans une capacité de calcul globale, facilement extensible via l'ajout de serveurs.

Il y a 3 grandes problématiques associées aux approches NFV :

- il n'y a plus un réseau à gérer mais plusieurs réseaux (dont le réseau physique). A première vue cela semble contraire à l'un des objectifs primaires du SDN : la simplification des opérations réseau ;
- l'interconnexion du réseau virtuel au reste du monde, à l'Internet. Les overlays restent le plus souvent confinés à certains domaines, comme le datacenter, et l'interconnexion reste nécessaire via des passerelles qui doivent supporter les protocoles du modèle overlay tout en garantissant les performances requises ;
- le réseau sous-jacent ne voit plus grand chose de ce qu'il transporte, puisque le trafic est encapsulé en amont dans un protocole ad-hoc comme par exemple VXLAN ou NVGRE.

Une approche NFV nécessite souvent un niveau d'expertise additionnelle en amont et une modification dans l'organisation des équipes en charge des serveurs/applications et du réseau. Les équipes réseau ne peuvent accepter de reposer sur des serveurs que s'ils ont la garantie de leur bon fonctionnement. Pour beaucoup d'organisation aujourd'hui, cela est un frein. Les *service providers* sont les plus avancés sur le NFV car ils ont davantage les capacités de faire face aux challenges du NFV : le réseau est leur métier. Pour autant les déploiements massifs d'architectures NFV n'en sont encore qu'à leurs prémices.

4 Programmation de bout en bout : le rôle de l'orchestrateur

Comme explicité précédemment, les solutions SDN vont varier selon le type d'environnement (WAN, LAN, Datacenter…) et comme c'est le cas aujourd'hui pour les solutions de management traditionnelles, des solutions indépendantes sont souvent déployées pour chaque domaine.

Or, la nécessité de programmation existe de bout en bout. Si l'application est déployée dans un datacenter, les usagers peuvent être connectés en Wi-Fi, connecté au LAN, derrière le WAN et des firewalls ou autres dispositifs réseau. Il est illusoire de penser qu'un seul contrôleur pourra programmer l'ensemble de la chaîne. La première raison est que les cycles d'acquisition de tous ces éléments sont différents aujourd'hui dans les organisations. L'autre raison est que la plupart des organisations veulent cloisonner ces domaines pour garder davantage de liberté, mais également parce qu'elles sont elles mêmes organisées en silos avec des équipes en charge directement d'un sous-ensemble du réseau.

Dans les modèles SDN actuels, ce n'est pas le contrôleur mais l'orchestrateur qui va offrir cette fonction de programmation de bout-en-bout. Ce dernier reçoit un ordre depuis une application (par exemple un portail de fourniture de services) et réalise ensuite la suite d'actions nécessaires pour mener à bien la tâche demandée. Pour réaliser sa mission, l'orchestrateur va pouvoir s'appuyer pour chaque élément de la chaîne sur le/les contrôleurs déployés.

L'orchestrateur travaille généralement en mode dit « transactionnel ». Il doit être capable de valider chaque étape quand il réalise une action afin de revenir en arrière en cas d'échec. Il ne peut pas laisser le réseau dans un mode bancal où seule une partie aurait été configurée avec un nouveau service.

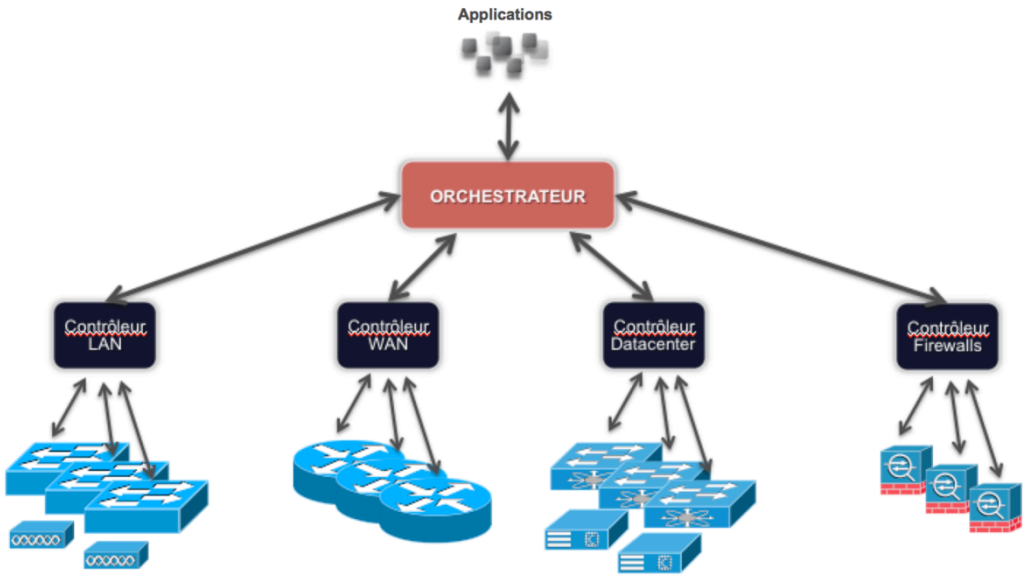

Figure 7 - Programmation de bout-en-bout avec un Orchestrateur

5 L'administrateur réseau doit-il apprendre à développer ?

Si les travaux principaux actuels portent surtout sur les contrôleurs et orchestrateurs afin de simplifier le management et accélérer le changement, il ne faut pas perdre de vue la cible : la programmabilité du réseau par les applications. Dans ce contexte, nous en sommes encore aux prémices puisque peu de solutions réelles sont encore présentes. L'application reste encore le plus souvent un outil de management centré sur les missions du réseau. Les administrateurs réseau ne sont donc pas perdus, mais ils commencent à comprendre qu'ils ne peuvent utiliser la puissance des outils à leur disposition que s'ils savent les personnaliser. En d'autres termes il est illusoire de penser qu'une application fera toujours le travail nécessaire et répondra parfaitement à la demande. Les contrôleurs et orchestrateurs sont de véritables plateformes de développement et savoir en tirer profit va devenir indispensable.

Dans ce contexte se distinguent aujourd'hui ceux qui se lancent dans ce que l'on nomme le « *devops* », c'est-à-dire le développement d'applicatifs, souvent simples, capables de réaliser des actions spécifiques sur le réseau (configuration, supervision…) Ils représentent aujourd'hui ce qui est certainement une première étape dans le métier d'administrateur réseau de demain. Ils peuvent plus rapidement tirer profit du réseau et gagnent en agilité. Ce n'est certainement qu'une

première étape tant les applications sont aujourd'hui au cœur des discussions là où nous parlions IP et protocoles de routage auparavant.

Les administrateurs réseau n'échapperont sans doute pas au Darwinisme : ceux qui resteront demain ne seront pas les meilleurs, mais bien ceux qui auront su s'adapter à temps.

6 Conclusion

Le Software Defined Networking annonce des changements importants sur les réseaux dans les années à venir. Ceux-ci vont voir leur architecture profondément évoluer, facilitant des nouveaux usages. Tout cela sera permis grâce à la programmabilité, l'ouverture, la virtualisation et l'orchestration. Aucun domaine ne semble épargné : WAN, datacenters, campus, sécurité… L'enjeu pour les administrateurs réseau est d'accompagner cette nouvelle étape afin de pouvoir tirer profit de ces nouvelles capacités.

7 Bibliographie

A. Bierman, M. B. (2015). draft-bierman-netconf-restconf - RESTCONF Protocol. *IETF Draft* .

Bjorklund, M. (2010). RFC 6020 - YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). *IETF RFC* .

Consortium, P. L. (s.d.). Récupéré sur P4: p4.org

Feamster, N., Rexford, J., & Zegura, E. (2013, 09 30). *The Road to SDN: An Intellectual History of Programmable Networks.* Consulté le 09 29, 2015, sur Princeton University web site: https://www.cs.princeton.edu/courses/archive/fall13/cos597E/papers/sdnhistory.pdf

Foundation, O. N. (2013, 01 01). *Software-Defined Networking (SDN) Definition*. Consulté le 08 18, 2015, sur Open Networking Foundation: https://www.opennetworking.org/sdn-resources/sdn-definition

Kerner, S. M. (2013, 4 29). *OpenFlow Inventor Martin Casado on SDN, VMware, and Software Defined Networking Hype*. Consulté le 8 18, 2015, sur Enterprise Networking Planet: http://www.enterprisenetworkingplanet.com/netsp/openflow-inventor-martin-casado-sdn-vmware-software-defined-networking-video.html

Linux Foundation. (s.d.). Consulté le 10 2, 2015, sur Open Daylight: https://www.opendaylight.org/

ONF. (s.d.). *Openflow Specifications v1.4.* Récupéré sur https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf

Robuck, M. (s.d.). *What's next for the ONF ?* Récupéré sur SDXcentral: https://www.sdxcentral.com/articles/news/whats-next-for-the-onf/2015/10/

Sherwood, R. (2014, 05 02). *Modern OpenFlow and SDN - Part I*. Consulté le 08 18, 2015, sur Big Switch network: http://bigswitch.com/blog/2014/05/02/modern-openflow-and-sdn-part-i