CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS

PARIS 2019-2020, semestre 2

Mémoire présenté en vue d'obtenir
l'UE ENG221 « Information et communication pour ingénieur »
Spécialité : INFORMATIQUE — Parcours Architecture
et Ingénierie des Systèmes et des Logiciels (A.I.S.L.)
Par Virginie Vélitchkoff

Sujet n° 37
L'évolution du SDN avec P4

Soutenu le 16 juin à partir de 17 h
Lien de la soutenance : À cette adresse sur Teams
Jury :
PRÉSIDENT : Kamel Barkaoui
MEMBRES : Faten Atigui, Éric Soutil

Sujet

L'évolution du SDN avec P4

Le SDN (Software-Defined Networking) est un nouveau paradigme permettant une programmation de la pile réseau et favorisant sa virtualisation. Récemment, ce paradigme a été revu avec le « Next-Gen-SDN » qui essentiellement vise à ne pas mettre des limites aux opérations qu'un commutateur peut effectuer dans le traitement du trafic. Une brique essentielle de cette évolution est le langage P4 qui permet une programmation des commutateurs tout en donnant des garanties de rapidité dans l'exécution du programme compilé et installé dans les commutateurs SDN. Il s'agit de présenter cette évolution du SDN et de toutes les nouvelles briques logicielles impliquées dans l'opération d'un réseau avec P4.

Mots-clés : SDN, P4, NFV

Bibliographie d'entrée :

https://www.opennetworking.org/p4/
Bosshart, Pat, et al. "P4: Programming protocol-independent packet processors." ACM SIGCOMM Computer Communication Review 44.3 (2014): 87-95.

Problématique

L'évolution du Software-Defined Networking (SDN) s'accompagne d'objectifs ambitieux répondants à des problématiques actuelles toujours plus novatrices.
Nous verrons dans ce document comment le SDN a répondu, tenté de répondre ou réponds actuellement à ces problématiques découpées en les trois phases de l'évolution du SDN comme l'explique McKeown dans le Keynote de ONF Connect 2019 [1]:

1. La création du SDN dont les objectifs étaient de pouvoir encourager l'innovation et d'accélérer la prise en compte de ces changements en donnant accès aux équipements de routage qui étaient propriétaires (côté matériel et logiciel) coûteux et rigide. L'enjeux était de prendre possession des spécifications et des logiciels de ces équipements pour pouvoir les programmer. L'architecture du SDN était de bas en haut.
2. Considérée comme la première phase de la prochaine génération SDN (NG-SDN), la seconde phase du SDN, dans laquelle nous sommes, a pour but d'améliorer OpenFlow en s'abstrayant totalement du matériel et en améliorant et en facilitant la programmation des équipements avec de nouveaux outils tel que le langage P4 et le compilateur P4Runtime. L'enjeu est d'installer une architecture du haut vers le bas : de s'abstraire complètement du matériel et de contrôler le traitement des paquets. L'innovation doit venir des programmeurs et non plus des équipementiers.
3. La troisième phase, NG-SDN, déjà initiée avec quelques interfaces développées ici, s'offre des objectifs ambitieux avec pour but de ne plus faire appel aux humains pour gérer un réseau qui lui est trop important et rapide à contrôler. Pour cela, il faut s'en remettre à la télémétrie (qui observe et vérifie le bon comportement de notre réseau, identifier et traquer les bugs...), à une programmation dynamique qui peut réagir aux erreurs ou aux comportements non désirés et enfin à un processus vérification du code et de son déploiement : une sorte d'Intégration continue et de déploiement continu (CI/CD). L'enjeu est des réseaux gérés par un contrôle en boucle fermée vérifiable.

Liste des abréviations

Abréviation	Terme français	Terme anglais
P4	P4	P4 for "**P**rogramming **P**rotocol-independent **P**acket **P**rocessors"
NG-SDN **Next-Gen-SDN**	Prochaine génération SDN	Next Generation SDN
SDN	Réseau défini par logiciel / Réseau programmable	Software-Defined Networking
NFV	Virtualisation des fonctions réseaux	Network Function Virtualization
FIB	Table de routage	Forwarding information base forwarding table MAC table
API	Interface de programmation	Application Programming Interface
IoT	Objets connectés	Internet of Things
NOS	Système d'exploitation réseau	Network Operating System
RVP	Réseau virtuel privé	Virtual Private Network (VPN)
LAN	Réseau local	Local-Area Network
OF	OpenFlow	OpenFlow
ONF	Open Networking Foundation	Open Networking Foundation
CI/CD	Intégration continue / déploiement continu	Continous Integration / Continuous Deployement
INT		In-band Network Telemetry

Glossaire technique

Terme français	Terme anglais	Définition
Commutateur réseau	Switch	C'est un équipement qui relie plusieurs segments (câbles ou fibres) dans un réseau informatique et de télécommunication et qui permet de créer des circuits virtuels. La commutation est un des deux modes de transport de trame au sein des réseaux informatiques et de communication, l'autre étant le routage. Dans les réseaux locaux (LAN), il s'agit le plus souvent d'un boîtier disposant de plusieurs ports RJ45 (entre quatre et plusieurs centaines). **[2]**
OpenFlow	OpenFlow	C'est un protocole standardisé par L'ONF utilisé entre les couches de systèmes d'exploitation et la couche d'infrastructure. Elle permet l'abstraction du matériel. (Source : Virginie Vélitchkoff) Ces notions seront développées dans cette étude**.**
ONF	Open Networking Foundation	C'est un consortium non lucratif dirigé par des opérateurs pour la transformation de l'infrastructure réseau répondant aux modèles commerciaux des opérateurs. Ils sont initiateurs du SDN, du P4 et de bien d'autres choses… **[3]** L'ONF sera développé dans cette étude**.**
P4	Programming Protocol-independent Packet Processors	P4 est un langage de programmation permettant d'exprimer la logique de traitement des paquets au sein d'un élément de communication de réseau informatique tel qu'un commutateur, une carte réseau, un routeur ou une d'un appareil afin d'y exécuter des actions. **[4]** Ces notions seront développées dans cette étude**.**
P4Runtime	P4Runtime	L'API P4Runtime est une spécification de plan de contrôle pour contrôler les éléments du plan de données d'un périphérique défini ou décrit par un programme P4. L'API P4Runtime peut être utilisée pour des applications de plan de contrôle local ou distant. [5] En d'autres termes, c'est une API d'exécution pour gérer les entrées de table P4. (Source : Vélitchkoff Virginie)
Pile réseau	Networking stack	La couche de réseau ajoute la notion de routage des paquets d'information depuis une adresse source et en les transférant de proche en proche vers une adresse destination (c'est par exemple à ce niveau qu'interviennent les adresses IP). **[6] (voir section réseau)**
• Plan de commutation • Plan de données • Plan de transfert	• Data plane • Forwarding Plane	Dans le domaine du routage, le plan de commutation parfois appelé plan de données (Forwarding Plane ou Data Plane en anglais) définit la partie de l'architecture d'un routeur qui décide ce qu'il faut faire avec les paquets arrivant sur une interface d'entrée. Le plus souvent, il se réfère à un tableau dans lequel le routeur recherche l'adresse de destination contenue dans le paquet entrant et récupère les informations nécessaires pour déterminer le chemin, depuis l'élément récepteur, à

		travers la structure de commutation interne du routeur, jusqu'à l'interface(s) de sortie appropriée(s). **[7]**
Plan de contrôle	Control plane	Dans le domaine du routage, le plan de contrôle est la partie de l'architecture du routeur qui est concernée par l'élaboration de la topologie réseau ou les informations d'une table de routage (éventuellement augmentée) qui définit ce qu'il faut faire avec les paquets entrants. Les fonctions du plan de contrôle, comme la participation à des protocoles de routage, tournent dans l'architecture plan de contrôle. Dans la plupart des cas, la table de routage contient une liste d'adresses de destination et l'interface(s) de sortie(s) associée(s). La logique du plan de contrôle peut également définir que certains paquets doivent être rejetés, ou au contraire doivent être traités de manière préférentielle afin d'obtenir un haut niveau de qualité de service. Ceci est permis grâce à des mécanismes de différenciation de services. En fonction de la mise en œuvre spécifique du routeur, il peut y avoir une base d'informations de transmission distincte qui est peuplée (c'est-à-dire chargée) par le plan de contrôle, mais utilisée par le plan de commutation pour rechercher les paquets, à très grande vitesse, et de décider comment les gérer**. [8]**
Prochaine génération SDN	Next-Gen-SDN	Une plateforme open source SDN de nouvelle génération (NG-SDN) intégrant plusieurs technologies pour fournir une solution de réseau indépendante du matériel, programmable, vérifiable, sans contact humain pour sa gestion et sa configuration. Le tout reposant sur des logiciels open source. **[9]** Ces notions seront développées dans cette étude**.**
• Réseau défini par logiciel • Réseau programmable	Software-Defined Networking	Le Software-Defined Networking (SDN) est un modèle d'architecture réseau qui permet la séparation physique du plan de contrôle et du plan de commutation sur le réseau. Le plan de contrôle contrôle plusieurs équipements. **[10]** Ces notions seront développées dans cette étude.
Virtualisation des fonctions réseaux	Network Function Virtualization	La virtualisation des fonctions réseau (NFV) permet de virtualiser les services réseau (routeurs, pare-feu, modules d'équilibrage de charge, etc.) traditionnellement exécutés sur du matériel propriétaire. Ces services sont regroupés dans des machines virtuelles sur du matériel standard, ce qui permet aux opérateurs de services de faire fonctionner leur réseau sur des serveurs standards, plutôt que propriétaires. **[11]**
Table de routage	• Forwarding information base • Forwarding table • MAC table	Il vise à trouver l'interface de réseau de sortie appropriée à laquelle l'interface d'entrée doit transmettre un paquet. Il s'agit d'un tableau dynamique qui met en correspondance les adresses MAC avec les ports. **[12]**
ONOS	ONOS	Open Network Operating System (ONOS) est le contrôleur SDN open source leader pour la création de solutions SDN / NFV de nouvelle génération. **[13]** Ces notions seront développées dans cette étude**.**
µONOS	µONOS	C'est l'évolution de la plateforme de contrôleur ONOS SDN prenant en charge un plan de contrôle unifié, une

		architecture basée sur les microservices et une orchestration commune du plan de contrôle et de commutation. **[9]** Ces notions seront développées dans cette étude**.**
Stratum	Stratum	Stratum est un agent d'un commutateur fournissant le contrôle, la configuration et les opérations via P4Runtime, gNMI et gNOI de ce commutateur. **[9]** Ces notions seront développées dans cette étude**.**
Commutateur de boîte blanche	White Box Switches	Ces commutateurs ont vu le jour avec l'apparition de Stratum pour compenser les matériels propriétaires onéreux. Ils peuvent être programmés avec OpenFlow ou tout autre Southbound API. Ils prennent également en charge des architectures de processeurs réseau plus diverses. **[14]**
La télémétrie réseau en bande	Inband Network Telemetry	C'est un cadre conçu pour permettre la collecte et la production de rapports de l'état du réseau, par le plan de données, sans intervention ni travail du plan de contrôle. **[15]**
Pipeline de plan de commutation	Pipeline of match-action table	Enchaînement de tableaux d'action de correspondance (Source : Virginie Vélitchkoff).
Yang	Yang	Un langage de modélisation de données écrit en Yang utilisé pour décrire la structure des données. Il est néanmoins possible d'extraire des données si elles sont conformes à la structure de données définie **[16]**
OpenConfig	OpenConfig	OpenConfig a la volonté de compiler un ensemble cohérent de modèles de données indépendants des fournisseurs (écrits en YANG). Ces modèles sont décidés en fonction des besoins opérationnels réels. **[17]**
gRPC	gRPC	gRPC est un framework RPC (Remote procedure call) open source initialement développé par Google. Il utilise le protocole HTTP/2 pour le transport, Protocol Buffers comme langage de description d'interface et offre des fonctionnalités telles que l'authentification, la transmission bidirectionnelle et le contrôle de flux, par le blocage ou non des communications par annulation ou délais d'attente. Il permet la construction de liaisons client/serveur multiplateforme pour de nombreux langages. **[18]**

Table des figures et tableaux

Tableaux

Figures

Table des matières

1. Introduction

1.1 Explication du sujet

Le sujet porte sur un nouveau paradigme qui introduit une nouvelle architecture réseau, le SDN qui remet en cause la rigidité d'un matériel propriétaire à l'architecture verticale qui intègre plan de contrôle et plan de commutation en un seul et même équipement. Cette nouvelle architecture, le SDN, a pour but de séparer le matériel de l'intelligence logicielle pour accélérer l'innovation et la recherche et favoriser l'apparition, l'amélioration de nouvelles applications ou fonctionnalités réseaux.

L'API OpenFlow en est une brique importante puisqu'elle a permis cette abstraction du matériel en proposant une table fixe d'en-têtes compatibles avec tous les équipements en se basant sur ces derniers. Mais l'évolution de l'internet et des usages a conduit à une augmentation trop abondante pour pouvoir continuer de faire évoluer cette table motivant ainsi l'évolution et la naissance d'une NG-SDN initiée par P4 et P4Runtime.

Le langage P4 permet de traiter tous les protocoles sur tous les équipements avec en plus la possibilité de réutiliser un programme d'un équipement à un autre. Très vite, d'autres outils naissent comme le P4Runtime, implémenté sur le plan de contrôle, qui offre la possibilité de changer dynamiquement le contenu des tables de routage. Il est désormais possible de contrôler le processus de traitement des paquets. Vient ensuite d'autres outils présentés dans la dernière phase du NG-SDN qui ont pour but de rendre le fonctionnement du réseau de plus en plus indépendant du contrôle humain.

1.2 Annonce du plan

Mon plan s'est imposé de façon historique afin que le lecteur puisse apprécier l'évolution et les avancées du SDN.

Dans la première partie de ce document, j'explique les origines du SDN en général, ce qui a nécessité sa création, ses définitions, son publique. J'introduis son architecture et sa composition puis OpenFlow son élément phare. J'évoque ses domaines d'applications, ses apports et ses bénéfices et enfin les défis qui lui reste à relever.

Il existe plusieurs variations de l'architecture SDN et surtout plusieurs solutions d'API. Cette étude étant consacrée à l'évolution du SDN avec P4 je me suis focalisée sur les produits de l'Open Networking Foundation (ONF) qui est d'ailleurs la référence dans le domaine la créatrice du SDN…

Cet état des lieux fait, la seconde partie aborde l'évolution du SDN qui se fait en deux phases. La première avec l'arrivée révolutionnaire du langage P4 et du P4Runtime de leurs avantages et inconvénients puis dans une seconde phase de la NG-SDN je détail les nouvelles interfaces, je reviens sur le processus du P4 et P4Runtime dans ce nouvel environnement et je présente quelques solutions d'architecture adaptées à des technologies cloud ou 5G par exemple. Je conclue enfin sur la nouvelle architecture du NG-SDN.

En conclusion, je résume cette étude, j'expose ses éventuelles lacunes et j'énumère les thèses et sujets de thèses en cours qui me paraissent les plus prometteurs et intéressants aux vues de mes recherches.

1.3 Apports du document

Autant que faire possible j'ai essayé de résumer, comparer et critiquer les articles et études de recherches actuelles et passées pour permettre au lecteur d'avoir un sens critique, de donner une vue d'ensemble du SDN et de la révolution actuelle du P4 tout comme de lui communiquer toutes les informations qu'ils seraient intéressants ou utile d'approfondir selon des problématiques ou des objectifs précis ou plus personnels.
Plus encore, cette note de synthèse a pour objectif de faire prendre conscience de la nécessité de ces évolutions pour l'avenir des réseaux devant l'arrivée d'un trafic toujours plus important et plus compliqué à gérer. Et puisque les solutions proposées sont open source, j'espère encourager le lecteur à participer à ces projets.

2. « Software-Defined Network » (SDN)

2.1 Son histoire

Lors de la création du réseau internet l'architecture étroitement couplé du matériel réseaux répondait à un besoin de performance du réseau, avec une augmentation rapide du débit des lignes. Malheureusement, ces équipements à l'architecture verticale : plan de commutation et plan de contrôle sur la même machine posait des problèmes de performance, de complexité, d'isolation et surtout freinaient considérablement l'innovation : l'implantation d'un nouveau protocole pouvait mettre jusqu'à 10 ans selon Kreutz et al. (2015) [19].

La nécessité d'évoluer pour répondre aux ressources toujours plus importantes, à la complexité des réseaux avec l'apparition de nouvelles technologies et de nouveaux enjeux c'est donc fait sentir dès les années 80 mais c'est à la fin des années 90 que les premiers travaux ont prouvé la nécessité de la séparation du plan de commutation et du plan de contrôle.

Nick McKeown et al. (2008) publie un article sur le SDN et OpenFlow [20]. Professeur et chercheur à l'Université de Stanford, il explique en 2009 [21] son inspiration pour imaginer le SDN d'après le principe de l'architecture d'un ordinateur (fig1.): Peu importe la marque de l'ordinateur et ses caractéristiques, le système d'exploitation tourne dessus grâce à une couche de virtualisation qui permet de s'abstraire de l'infrastructure. Le système d'exploitation offre des applications qui répondent aux fonctionnalités désirées. Une isolation parfaite des couches et des rôles qui permet de faire évoluer une couche sans impacter les autres. C'est sur ce principe qu'il propose les pierres angulaires du SDN : la séparation du plan de commutation, du plan de contrôle, une forte isolation des couches et OpenFlow une API qui permet de contrôler le plan de contrôle.

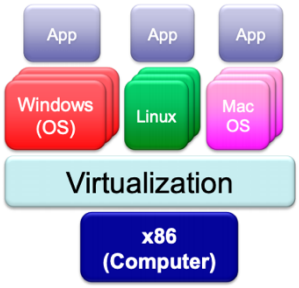

Figure 1 - Architecture d'un ordinateur x86 (source McKeown 2009 [21])

Rapidement suivi par la plupart des fournisseurs ceux-ci incluent la prise en charge de l'API OpenFlow dans leur équipement.

En 2011, l'Open Networking Foundation (ONF) [10] « un consortium non lucratif dirigé par des opérateurs pour la transformation de l'infrastructure réseau répondant aux modèles commerciaux des opérateurs »[1][3] est créé avec à son board notamment Nick McKeown et les plus grands opérateurs tel que AT&T, China Unicom, Comcast, Deutsche Telekom, Google and Turk Telekom. Ce consortium est à l'origine des projets SDN, P4, P4Runtime et bien d'autres.

L'histoire du SDN par l'ONF [10] :

- **2011 : SDN** permet de découpler le plan de control du plan de commutation
- **2012 : OpenFlow** devient la première interface standard pour séparer les plans de contrôle du plan de commutation.
- **2014 : ONOS** est le principal SDN Controller ou Network Operating System (NOS) pour les opérateurs
- **2017 : CORD** est une solution « Edge Cloud » destinés aux opérateurs qui projettent de l'utiliser à 70%

2.2 Ses définitions

Les définitions du SDN sont plus ou moins précises. Il me semble important de relever deux définitions, celle de l'ONF [22] et celle de Kreutz et al. (2015) [19] qui font toutes deux références dans le milieu.

2.2.1 La définition selon L'ONF [10]

Il s'agit de « la séparation physique du plan de contrôle et du plan de commutation sur le réseau. Le control plane contrôle plusieurs équipements. »[2].

2.2.2 La définition selon Kreutz et al. (2015) [19]

Plus précis, le SDN est défini par quatre piliers qui comme nous le verrons dans le chapitre suivant se découpent en trois couches :

1. Le plan de contrôle et le plan de commutation son découplé. Le plan de commutation intégré à l'appareil devient un simple élément de transmission de données alors que le plan de contrôle en est retiré.
2. Les décisions de transmissions sont basées sur le flux et non sur la destination : un flux est défini par un ensemble de paquets avec un critère de correspondance et des instructions ou actions qui s'y rapportent.
3. La logique de contrôle est déplacée vers une entité externe : le contrôleur SDN ou NOS (Network Operating System). Le NOS est une plateforme logicielle qui fonctionne sur la technologie des serveurs de base et fournit les ressources et abstractions essentielles pour faciliter la programmation des périphériques de transfert sur la base d'une vue de réseau abstraite et centralisée logiquement. Son objectif est donc similaire à celui d'un système d'exploitation traditionnel.
4. Le réseau est programmable grâce à des applications logicielles exécutées au-dessus du NOS qui interagissent avec les dispositifs de plan de données sous-jacents. Il s'agit

[1] "The Open Networking Foundation (ONF) is a non-profit operator led consortium driving transformation of network infrastructure and carrier business models." Traduction : V.V.
[2] "The physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices." Traduction : V.V.

d'une caractéristique fondamentale de SDN, considérée comme sa principale proposition de valeur.

Tous ces points seront développés dans le chapitre suivant.

2.3 Sa composition, son architecture

Selon Kreutz et al. (2015) [19], le SDN est composé de trois couches principales énumérées ici. S'ensuit une représentation graphique de cette architecture ainsi qu'une explication détaillée.

Les trois couches principales :

1. Application ou Management plane
2. Contrôle ou plan de contrôle
3. Transmission ou data plane

Ces couches sont reliées entre elles par plusieurs API :

1. Northbound interface
2. East/West interface
3. Southbound interface

2.3.1 Représentation graphique de l'architecture SDN

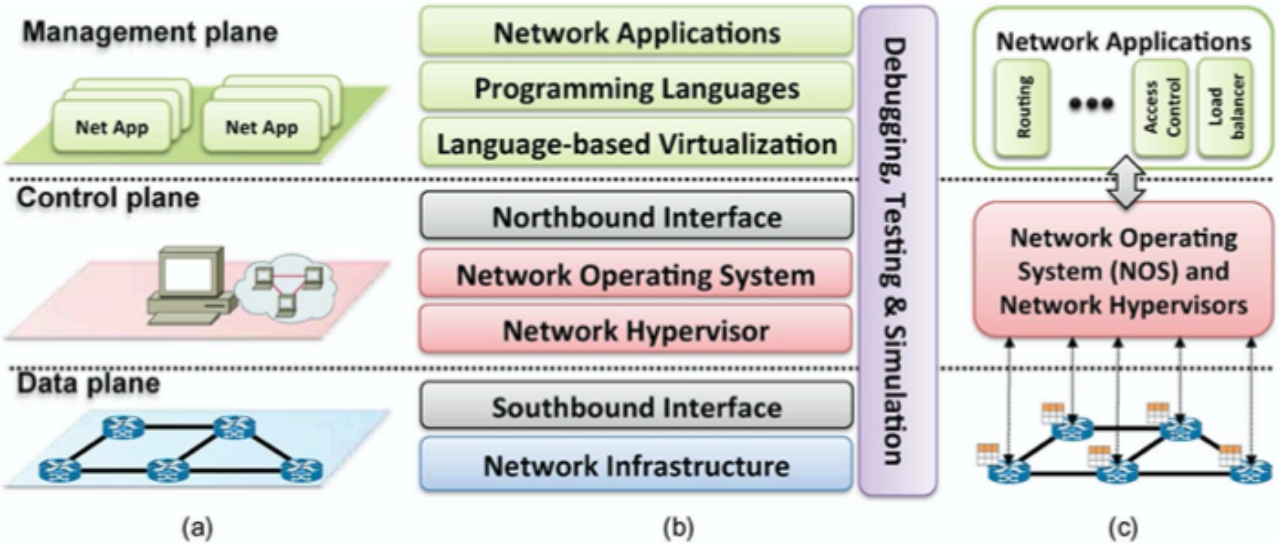

Figure 2 - Software-Defined Networks in (a) planes, (b) layers, and (c) system design architecture. (Source : Kreutz et al. (2015) [3])

2.3.2 Explications détaillées de l'architecture SDN

J'ai utilisé plusieurs sources pour être au plus précis : [2], [6], [7]

1. **La couche Application réseau ou Plan de gestion (Management plane)** permet de déployer des nouvelles fonctionnalités réseau à travers des applications (la qualité de service, la sécurité, l'orchestration, l'automatisation, la virtualisation etc….) en fonction des ressources (prévision du trafic, gestion dynamique des ressources, surveillance du réseau ou des équipements…). Son rôle est de déterminer une politique de plan de contrôle centralisé qui sera renforcé par les plans de contrôle.

L'API Northbound fait le lien entre les couches 1 et 2.

2. **La couche de système d'exploitation réseau ou plan de contrôle** est composé de plusieurs plans de contrôle distribués qui gère le flux de données. Leur rôle est de définir comment gérer le trafic réseau et ses équipements en fonction de la politique du plan de gestion.),
    - Il doit définir une politique de traitement des paquets de données (rejet, transmission, redirection, modification des en-têtes, envoi à un ou plusieurs ports, etc…).
    - Il représente les protocoles utilisés pour peupler les tables de routage des plans de commutations
    - Il configure ou éteint les data planes.

    Ses qualificatifs attendus sont la capacité d'adaptation et de personnalisation, sa gestion des politiques de gestion de flux, sa capacité à gérer les situations exceptionnelles et à faciliter le traitement du plan de commutation.

L'API East/West est destinées à faire le lien entre plusieurs plans de contrôle de la couche contrôle.

3. **La couche d'infrastructure réseau ou le plan de commutation** comprend le matériel : routeurs, switches physiques ou virtuelles, commutateurs... Son rôle est d'exécuter les décisions du plan de contrôle tel que :
    - Rechercher les données.
    - Traiter, aiguiller et mettre en action la politique de gestion de plan de control sur les requêtes de données.
    - Collecter des statistiques

    Ses qualificatifs attendus sont la rapidité, la simplicité et la régularité du traitement.

L'API Southbound fait le lien entre les couches 2 et 3. Les contrôleurs interagissent avec le matériel via l'API et un protocole. Le protocole standardisé et le plus répandu à ce jour est OpenFlow de L'ONF. D'autres alternatives existent tel que ForCes ou Open vSwitch Database.

2.4 OpenFlow : Une brique essentielle

2.4.1 Ses origines

OpenFlow (OF) est le premier standard officiel du SDN [25] présenté par l'ONF qui propose certifications et validations de compétences. Il constitue l'API Southbound qui fait le lien entre la couche 2 (système d'exploitation réseau) et la couche 3 (d'infrastructure). Ce protocole est un élément fondamental pour la construction de solutions SDN puisqu'il permet de programmer un plan de contrôle pour n'importe quel équipement et logiciels réseau. Sans lui pas d'abstraction de l'équipement.

2.4.2 Son architecture

Selon Choukri, al (2019) [23] L'API OpenFlow se retrouve à chaque plan de l'architecture SDN : le plan de gestion a une chaîne de sécurité en lien avec le commutateur, le plan de contrôle à un contrôleur OF (ex. : OpendayLight, NOX…) et le plan de données à un commutateur OF nommé tout simplement OpenFlow.
OpenFlow est intégré dans le commutateur où ils possèdent plusieurs tables de flux qui contiennent chacune des règles (composées d'en-tête, d'action et de compteur statistiques). Ces règles définissent des sous-ensembles de données à traiter auxquelles sont attribuées des actions.
Voici quelques illustrations de cette architecture tirées d'une présentation d'une présentation de McKeown (2009) [21]:

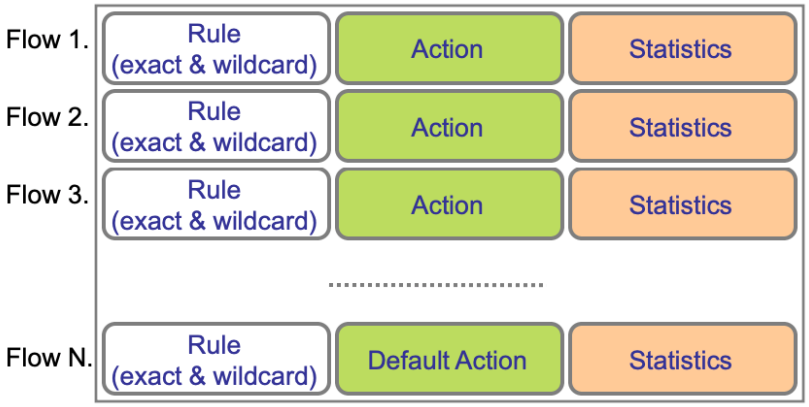

Figure 3 - Architecture d'une table de flux composée de règles (source : McKeown (2009) [21])

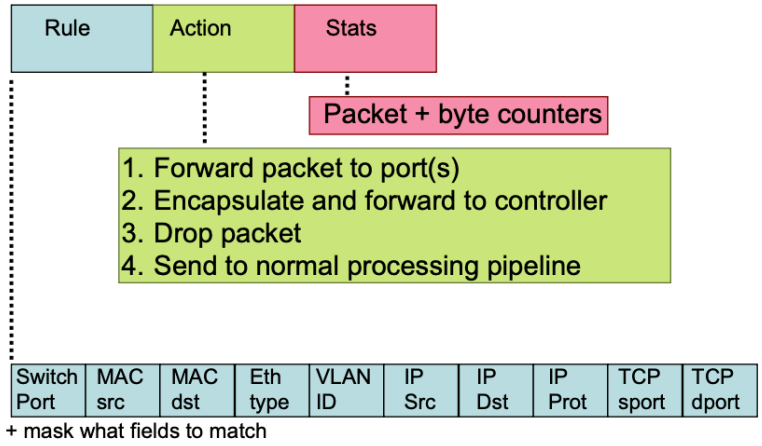

Figure 4 - Détails d'une règle (source : McKeown (2009) [21])

2.4.3 Son fonctionnement

Le fonctionnement est simple : un paquet entre dans le commutateur qui vérifie que son en-tête est contenu dans l'une des règles de ces tables puis exécute l'action qui lui est attribuée. Si l'en-tête est inconnue, le contrôleur récupère l'information et la traite soit en supprimant le paquet soit en créant une nouvelle règle.
Les Api OpenFlow control des OpenFlow switch et permettent de se comporter comme des router, switch, firewall, ou avoir d'autres rôles.

2.5 Son public

Le public est large ! Des constructeurs d'équipements réseaux, des opérateurs, des fournisseurs d'accès réseaux et des administrateurs réseaux.

2.6 Ses domaines d'applications

Initialement testé sur des réseaux locaux et des data center, le SDN est aujourd'hui utilisé à l'échelle d'internet.
Boucadair et Jacquenet (2013) [24] ont montré que certains déploiements n'étaient pas appropriés : implémentations logicielles complètement flexible, implémentations complètement modulaires, ou un plan de contrôle exclusivement centralisé.

A contrario, les réseaux virtuels privés (RVP) sont les plus adaptés de par leur complexité intrinsèque qui incite à l'automatisation de la gestion des ressources importantes et des configurations qu'elles nécessitent. Ainsi, la couche application doit notamment gérer les besoins topologiques, l'adressage, le routage, les politiques de sécurité et enfin de qualité de service.

En outre, le SDN est fait pour répondre à des problématiques de enjeux multi-protocoles, multi-technologies, multi-services et besoins mobiles comme le souligne Choukri, al (2019) [23] l'architecture découplée, dynamique, adaptable et rentable ainsi que la vue globale du réseau quelle offre rend le SDN idéale pour les défis actuelles dans les domaines du Big Data (possibilité de reconfiguration à volonté), Machine Learning (prévision du trafic, sécurité….), 5G, IoT et Smart cities (scalabilité).

2.7 Ses apports et bénéfices

Les bénéfices présentés par l'ONF sont :

- Une administration du réseau centralisée qui permet une gestion globale et intelligente sécurisé, optimisé voir automatisé du réseau.
- Le plan de contrôle, programmable, gère et contrôle un ou plusieurs appareils via des API (Application Programming Interface). Des programmes qui ne dépendent pas de logiciels propriétaires grâce à l'abstraction du matériel
- Le découplage du plan de commutation permet d'abstraire le matériel sans impacter sur la politique du plan de control. Ainsi chaque couche de l'architecture a sa propre couche avec son propre rôle qui n'impacte pas les autres en cas de changement.

A mon sens, la force de la solution SDN par L'ONF est d'avoir travaillé à la fois avec des chercheurs universitaires et des opérateurs pour mettre en place une architecture et un standard open source qui continue de favoriser l'innovation et la recherche. C'est ce mouvement commun qui a favorisé l'acceptation de cette technologie par le plus grand nombre.

3. La nouvelle génération du SDN (NG-SDN)

Le SDN a été largement adopté, des fournisseurs comme Google ont implanté et mis en production cette architecture. Toutes ces expérimentations, ont amenées à d'autres objectifs et améliorations qui s'installent actuellement en deux phases :

- La phase 1 : Un changement d'architecture avec la révolution P4 et P4Runtime pour l'abstraction complète de l'équipement et la propagation de la programmation dans toutes les strates du domaine du réseaux. Cette phase à commencer et toujours en cours.
- La phase 2 : L'indépendance des réseaux programmables avec la télémétrie, le CI/CD, la conteneurisation et l'orchestration… L'aboutissement de cette phase est prévu pour 2030…

3.1 Phase 1 : la révolution P4 et P4Runtime

3.1.1 La révolution P4

Selon Bosshart et al. (2014) [26], je vais vous présenter P4 à travers ses origines, ses différences avec OpenFlow, ses objectifs, son fonctionnement succinct et ses avantages.

3.1.1.1 Ses origines

OpenFlow a été prévue pour des réseaux locaux et des data center gérant une douzaine de champs dans ses tables de flux. Son succès et la généralisation du SDN a conduit à la

prolifération des champs de ces tables et à la diversité des protocoles. OpenFlow n'est donc plus en mesure de couvrir l'ensemble des possibilités et des évolutions. Ce constat remet ainsi en cause de façon partielle l'abstraction du matériel sur laquelle repose le SDN.
Pour répondre à cette problématique et tendre vers une abstraction totale des équipements, Bosshart et al. (2014) [26] crée Le langage open-source de haut niveau P4 (**P**rogramming **P**rotocol-Independent **P**acket **P**rocessors) afin de faire correspondre les paquets avec n'importe quels champs d'en-têtes qui s'y rapportent pourvu qu'ils soient définis dans un programme P4.
P4 c'est aussi un Langage dédié qui définit formellement le pipeline du plan de commutation tel que : les en-têtes de protocole, les tables de correspondances, les actions, les statistiques, etc. La vitesse du pipeline peut être configurée selon l'équipement qui lui destiné.
P4 est rétroactif avec les commutateurs à fonction fixes puisqu'il définit un contrat entre le plan de contrôle et le plan de commutation et bien sûr il est compatible avec les commutateurs programmables !

3.1.1.2 Ses objectifs

- Des commutateurs indépendants des protocoles réseaux.
- Le programmeur peut traiter les fonctionnalités et actions des paquets indépendamment des spécificités des équipements.
- Le programmeur doit pouvoir reprogrammer, reconfigurer le processus des paquets déployés sur les équipements : définir leurs comportements.

3.1.1.3 Les différences avec OpenFlow ou les évolutions depuis OpenFlow…

Tableau I – Les différences avec OpenFlow (source : Bosshart et al. [26] et Virginie Vélitchkoff)

OpenFlow	P4
Champs d'en-tête fixe contraint par la conception des commutateurs ⇨ **Abstraction du matériel limitée**	Les champs d'en-tête sont programmables ⇨ **Pas de limites à l'innovation**
Les correspondances et actions sont fait en série	Les correspondances et actions sont fait en série ou/et en parallèle ⇨ **Gain de temps et d'efficacité**
Les actions dépendent des protocoles proposés par le commutateur. Ils ne sont pas détaillés voir même inexistant pour certains protocoles.	Les actions sont composées de primitives indépendantes du protocole prise en charge par le commutateur. ⇨ **Isolation complète**
Impossible de reconfigurer le processus d'un protocole	Reconfiguration de la correspondance des champs et de leurs traitements ⇨ **Liberté de tester, d'évoluer, de changer**
	P4 est indépendant du matériel et ses programmes peuvent donc être utilisés, réutilisé sur n'importe quel équipement pourvu que le compilateur puisse s'accorder avec l'équipement. ⇨ **Abstraction de l'équipement complète**

3.1.1.4 Programmer un fichier P4

- Mise à disposition par le fournisseur du commutateur.
    - Le fichier d'architecture : Ce fichier définit quel bloc sont disponibles et leurs capacités (quelques fonctions fixes : paquet, file d'attente, réplication, planification). Il est aussi nécessaire dans le cas d'un commutateur programmable.
    - Le compilateur P4 qui permet de générer un binaire compatible avec le commutateur.
    - Le plan de commutation intégré au commutateur dans lequel sera logé les instructions du programme P4

- Dans le fichier de programmation, spécifier les formats des en-têtes de protocoles en listant, dans l'ordre, tous les champs nécessaires et leur largeur.
- Programmer un comportement pour les pipelines du plan de commutation et d'autres fonctions tels que des compteurs, registres d'état, fonctions d'hachage….
- Compiler le programme.
- Le plan control prendra ensuite le relais pour peupler les tables définies par le programme. C'est là qu'intervient P4Runtime…

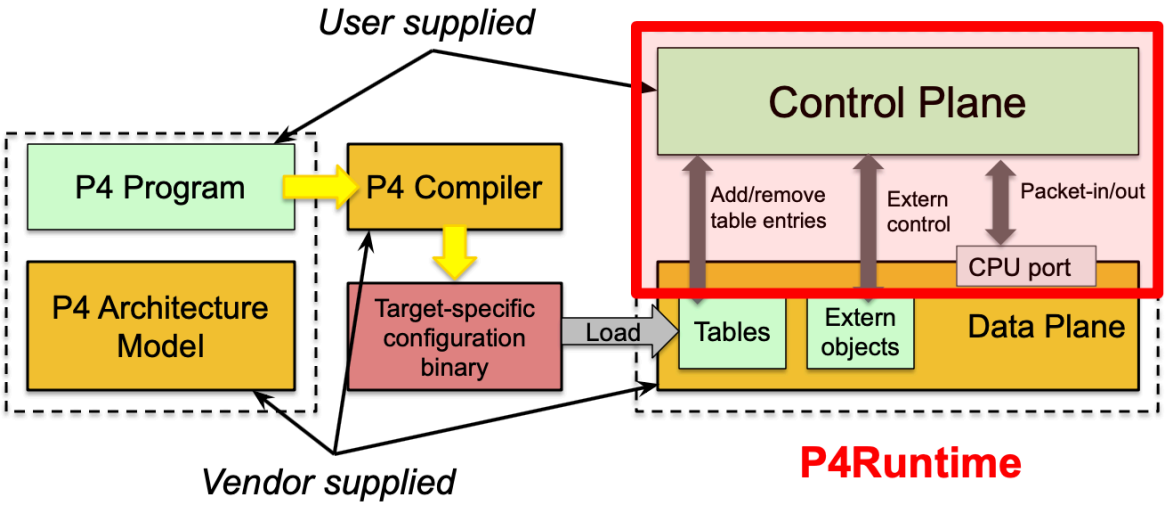

Figure 5 - Déroulement du processus de P4 (Source : Cascone [27])

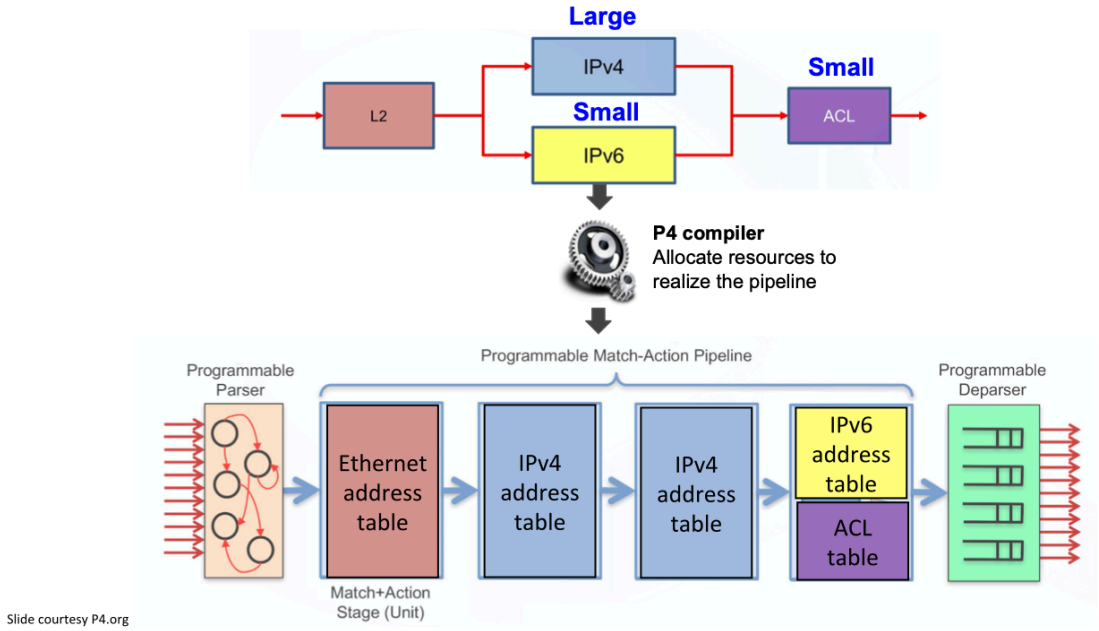

Figure 6 - Compiling P4 on a programmable switch (Source : Cascone [27])
Pour cet exemple IPV4 est programmé avec une table de correspondances plus grande que pour IPV6 ou ACL selon des besoins fictifs de cette démonstration.

3.1.2 Les bénéfices et inconvénients de P4

✚ Pas de limites aux opérations qu'un commutateur peut effectuer dans le traitement du trafic

- ✚ Langage P4 permet une programmation des commutateurs tout en donnant des garanties de rapidité dans l'exécution du programme compilé et installé dans les commutateurs SDN.
- ✚ Une interface et une personnalisation du comportement pour l'équipement
- ✖ Un programmable P4 n'est pas toujours portable d'un équipement à un autre
- ✖ Il semblerait que des bugs existent comme dans tous programme. D'où le développement en cours d'une interface pour la vérification du code.

3.1.3 P4Runtime

Initié par L'ONF et Google en 2016, les premières spécifications ont été définis en 2019 [28]. L'API P4Runtime est une spécification de plan de contrôle pour contrôler les éléments du plan de commutation d'un périphérique défini ou décrit par un programme P4.

3.1.3.1 Quelques caractéristiques

- P4Runtime est le seul outil qui propose l'indépendance des protocoles : Contrairement à OpenFlow qui dépend de ce que le matériel à contient, P4Runtime peut contrôler le plan de commutation de n'importe quel protocole standard ou personnalisé.
- L'API est indépendante des commutateurs et de ses fournisseurs et indépendante des pipelines de plan de commutation qui ont été spécifiquement formalisé et programmer en P4.
- L'API se met automatiquement à jour dès que de nouvelles fonctionnalités sont introduites dans un programme P4 et est exposée à un schéma de contrôle externe. La programmation est dynamique et cela nous donne une API unique qui peut être utilisée pour contrôler plusieurs interrupteurs polyvalents.
- L'API permet de mieux contrôler des contrôleurs distribués.
- P4Runtime est le cœur de l'interface de Stratum.

3.1.4 Conclusion sur cette première phase

Les objectifs définis ont été atteints. L'ONF a installé une architecture du haut vers le bas : Le SDN dépend de la programmation et non plus du matériel permettant la personnalisation, la souplesse et l'innovation. En outre, l'API P4Runtime offre la programmation dynamique et donc le pouvoir de tester plus facilement et surtout n'importe quel protocole peut être traité sur n'importe quel équipement !

3.2 Phase 2 : La nouvelle architecture et ses interfaces

Toutes les définitions et les informations de ce chapitres sont principalement tiré du site de L'ONF [9].
La plateforme open source de nouvelle génération (NG-SDN) intègre plusieurs technologies pour fournir une solution de réseau indépendante du matériel, programmable, vérifiable et sans contact humain, basée sur des commutateurs de boîte blanche (créées par l'ONF afin de limiter les coûts et de ne pas dépendre d'un matériel propriétaire onéreux) et des logiciels open source.
Les composants open source de cette nouvelle architecture comprennent :

- µONOS (une évolution de ONOS)
- Stratum
- Un Moteur de vérification NG-SDN
- NG-SDN Opérationnalisation d'un socle d'applications

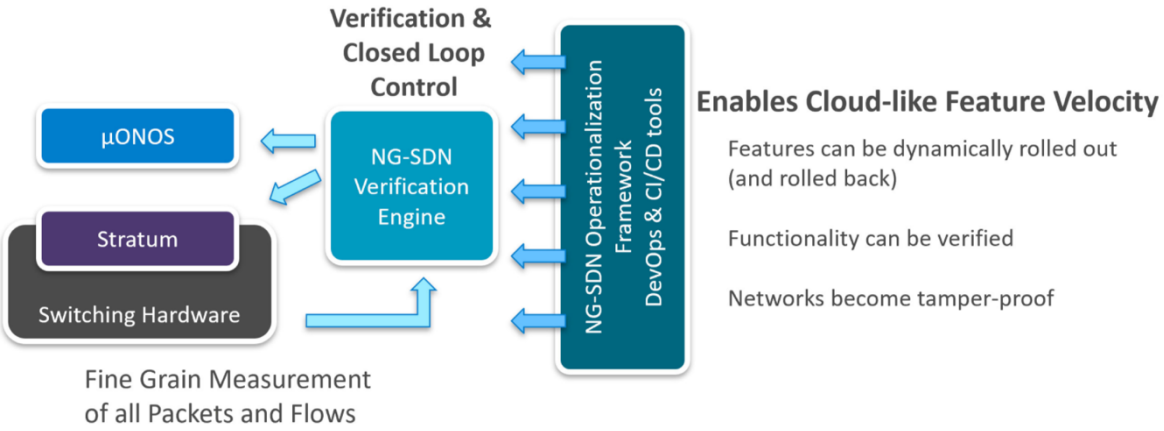

Figure 7 - Composants de la NG-SDN (source : L'ONF [9])

3.3 La présentation des interfaces de la NG-SDN

3.3.1 Stratum

Stratum est un système d'exploitation open source destinés pour le SDN. C'est une distribution légère qui intègre le minimum des interfaces P4Runtime, OpenConfig et des outils de configuration et de télémétrie (décris par la suite) et destiné à des commutateurs de boîte blanche.

3.3.1.1 Les outils de configuration et de télémétrie

- **Le protocol gNMI** est utilisé pour changer, gérer télémétrer les données d'un matériel.
- **gNOI** (Network Operations Interface) gère les services d'opérations ponctuelles, tels que redémarrage, test ou ping d'appareils.

Ces deux composants sont basés sur gRPC qui utilise OpenConfig.

3.3.1.2 Ses avantages

- Il permet l'interchangeabilité des périphériques de routage et la programmation de leurs comportements.
- Il évite le verrouillage des fournisseurs des plans de commutation (interfaces silicium propriétaires, API logicielles fermées).
- Il permet une intégration facile des appareils dans les réseaux des opérateurs (réduisant ainsi les erreurs et les coûts.

3.3.2 Onos

Onos une plateforme de contrôle et de configuration cloud native SDN qui gère un réseau de commutateurs Stratum de manière logiquement centralisée.
ONOS prend en charge à la fois la configuration et le contrôle en temps réel du réseau, éliminant ainsi la nécessité d'exécuter des protocoles de contrôle de routage et de commutation à l'intérieur de la structure du réseau. En déplaçant l'intelligence dans le contrôleur cloud ONOS, l'innovation est encouragée : les utilisateurs finaux peuvent facilement créer de nouvelles applications réseau sans avoir à modifier les systèmes de plan de données.
Les avantages d'Onos en font un élément de choix pour les réseaux Edge, les réseaux de transports ou encore les Datacenter.

3.3.2.1 Ses avantages

- Scalabilité
- Haute performance
- Résilience
- Prise en charge des anciens périphériques comme les appareils de nouvelle génération

3.3.3 Exemple d'un processus de P4 et P4Runtime dans cette nouvelle architecture

- Programmer son fichiers P4
- Compiler le programme avec le compilerP4 pour s'assurer qu'ils répondent aux exigences de l'équipement réseau. Deux fichiers sont générés :
    - un contrat, *.p4info, destiné au plan de control (ex. : ONOS) et au système d'exploitation du commutateur (ex. : Stratum). Ce fichier est indépendant de l'équipement et donc réutilisable dans le cas de plan de contrôle distribué avec ou pas du matériel différent.
    - Un binaire, *.bin, qui alloue les ressources pour exécuter l'enchaînement des actions à menées et générer un mappage de ces exécutions. Ce fichier est destiné à un commutateur dédié et à son driver.

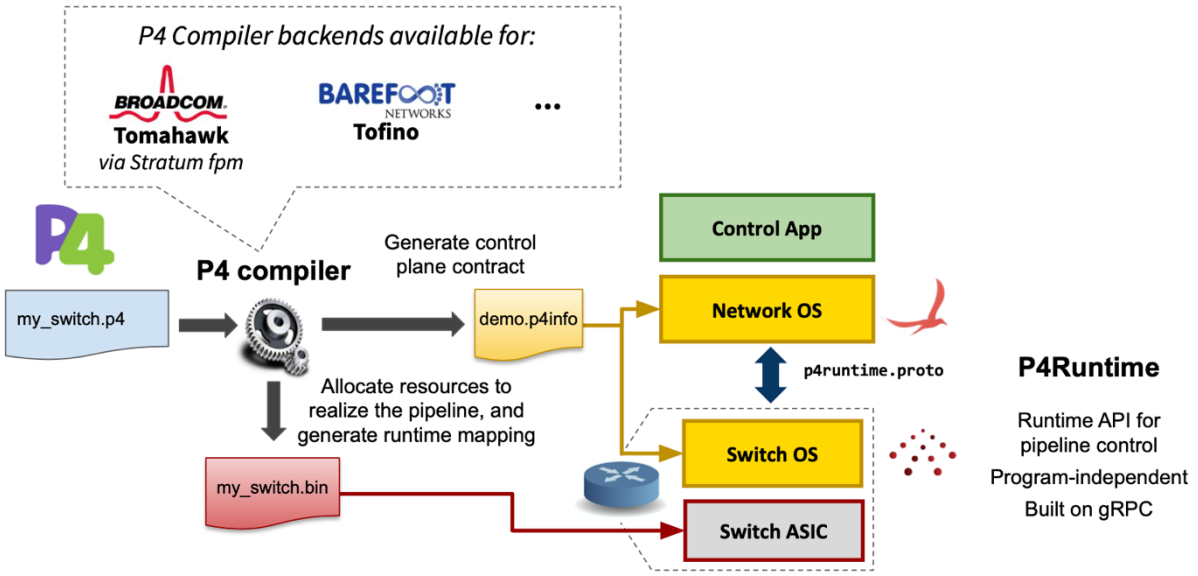

Figure 8 - Processus de la programmation P4 (Source : L'ONF (2019) [29])

3.3.4 Moteur de vérification NG-SDN

La meilleure granularité de mesure du réseau de chaque paquet et du flux permet une vérification continue du comportement précis du réseau. Le réglage et les mises à jour automatisés garantissent que le réseau offre le comportement précis défini par l'opérateur, garantissant ainsi que le réseau fonctionne comme spécifié.

3.3.5 NG-SDN Opérationnalisation d'un socle d'applications

Le réseau est géré selon la méthode Continuité d'intégration et de déploiement (CI/CD), permettant des mises à jour rapides, des déploiements automatisés, une vérification et des restaurations automatisées. Le réseau devient une ressource dynamique et agile qui peut être exploitée et mise à jour sans craindre que les changements ne dégrade l'infrastructure de l'opérateur.
Docker peut être utilisé pour déployer des images de configuration et Kubernetes pour orchestrer les différents commutateurs comme le montre l'exemple ci-dessous.

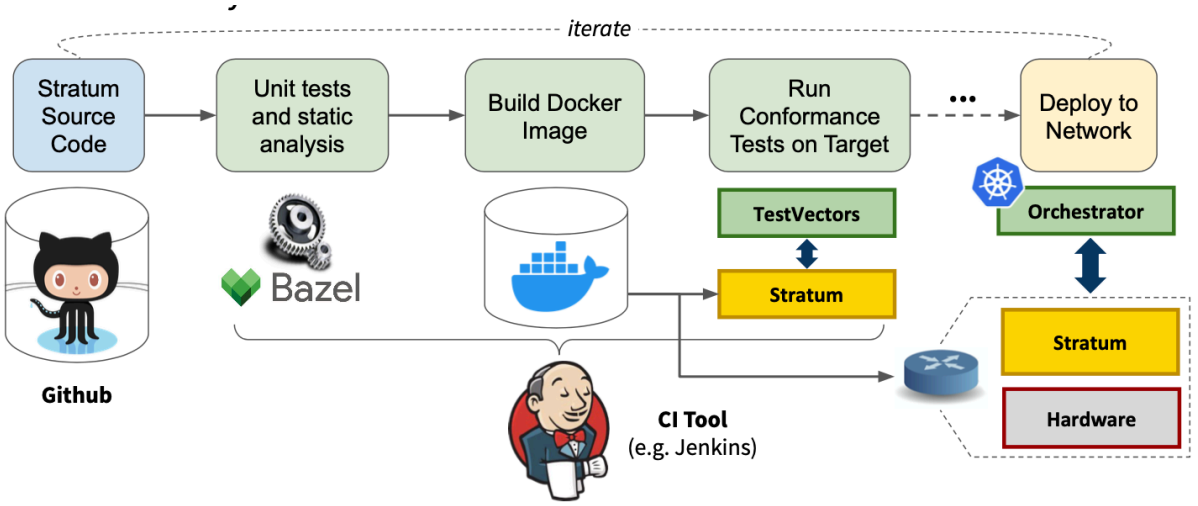

Figure 9 - SDN and CI/CD (Source : Sloane [29])

3.3.6 MININET – un environnement de test

Mininet est un environnement de test virtuel qui permet de tester l'architecture SDN. Il permet de prototyper rapidement de la solution la plus simple à la plus complexe sans avoir besoin d'équipements physique. Par la suite, il semble assez aisé de déployer la solution travaillée sur cet environnement de test à un environnement de production.

3.4 Plusieurs architectures reposant sur Stratum sont possibles en voici quelques 'une

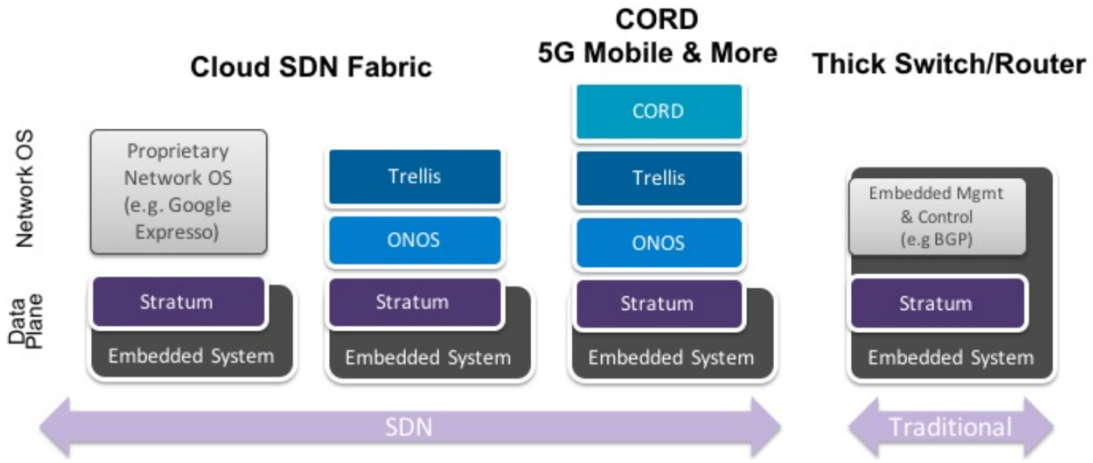

Figure 10 - Exemples d'architectures NG-SDN (source : L'ONF [30])

3.5 Les bénéfices et les inconvénients de cette nouvelle architecture

- Une indépendance complète au matériel
- Un système de couche isolé
- Une interface de contrôle et de gestion automatisé
- La télémétrie qui fait son apparition…
- Selon McKeown « Je pense qu'un réseau doit être aussi facilement programmable qu'un serveur. En faisant passer les protocoles du matériel au logiciel, nous allons déclencher une explosion cambrienne d'innovations dans le domaine des réseaux, comme nous l'avons vu dans le domaine de l'informatique. Grâce à son utilisation de

P4Runtime, Stratum ouvre la porte à chaque opérateur de réseau pour qu'il adapte son réseau à ses besoins, tout en bénéficiant d'un riche écosystème de logiciels libres et d'un choix de fournisseurs de silicium. »

4. Conclusion et Perspectives

4.1 Résumé de cette étude et de ses éventuelles lacunes

4.1.1 Résumé de cette étude

La première phase de l'évolution du SDN était de pouvoir encourager l'innovation et d'accélérer la prise en compte de ces changements en simplifiant le rapport aux équipements de routage tout en réduisant les coûts. La première version du SDN réussit à mettre en production son nouveau paradigme qui sépare le plan de contrôle du plan de commutation avec l'utilisation d'OpenFlow et de centraliser le contrôle des plans de contrôle. Le monde du réseau peut désormais prendre le contrôle sur la partie logicielle des équipements.
La nouvelle génération du SDN se compose en deux phases répondant à des objectifs distincts. La première phase, actuellement en cours, a pour but d'améliorer les expériences d'OpenFlow et d'inverser la logique de l'architecture : On doit pouvoir programmer en partant du plan de control et non plus des commutateurs. L'isolation complète du matériel de la logique de contrôle est donc renforcée par de nouveaux outils que sont P4 et P4Runtime. Et parce que dans cette version du SDN on donne la main aux programmeurs et non plus aux constructeurs, on intègre aussi le contrôle du traitement des paquets.
La deuxième phase, prévue pour 2030, commence déjà à poser les strates nécessaires à la programmation tout entière du monde du réseau pour que le système s'autogère un maximum et que l'humain n'intervienne qu'un minimum. Les infrastructures essentielles pour cette évolution, sont le développement des systèmes d'exploitation Stratum pour les commutateurs et Onos pour le plan de contrôle qui permette de virtualiser et d'intégrer les outils indispensables au SDN. La télémétrie, la couche de vérification, l'architecture orientée CI/CD sont encore peu développée aujourd'hui mais auront un grand rôle à jouer pour donner les indications nécessaires et permettre cette autogestion.
L'ONF est très active et semble toujours en mouvement, une nouvelle version du compiler de P4 est en cours par by Tomasz Osiński, un micro-livre vient de sortir aujourd'hui "Software-Defined Networks: A Systems Approach," par Larry Peterson, Carmelo Cascone, Brian O'Connor, and Thomas Vachuska [31].

4.1.2 Les lacunes que j'aurais aimé traiter

- Faire une démonstration et tester l'architectures et ses infrastructures.
- Aborder le débogage, les anomalies et les problèmes de mise en place du SDN.
- Une vue d'ensemble des alternatives pour pouvoir comparer les solutions de L'ONF à ses concurrents.
- Explorer le rapport entre le SDN et l'IoT et l'Edge Computing par curiosité personnelle.

4.2 Défis encore à relever, recherches en cours et alternatives

Boucadair et Jacquenet (2013) [24] soulignent la complexité du paramétrage de plusieurs fonctions combinées, les problèmes de sécurité tel que le déni de service et les politiques de filtrage, la question de la performance et de la scalabilité dépendant du service à fournir et de la masse de trafic à gérer, la capacité à faire cohabité des architecture SDN avec d'autres plus classiques, le besoin de standardisation.

Quelques années plus tard, Choukri, al (2019) [23], souligne plusieurs défis qu'il catégorise selon la performance, la scalabilité et la fiabilité. On constate que certains défis ont persistés comme les problèmes de sécurité et d'autres comme le besoin de standardisation a été résolu. Il ne s'agit pas ici de détailler tous ces points mais on peut toutefois noter que certains défis persistent et que des solutions ont été trouvées ou sont en développement comme il est précisé dans cet État de l'art.

A mon sens, l'un des défis les plus intéressant et qui contribuerait grandement à la performance, la scalabilité, la fiabilité, la sécurité et la gestion du réseau serait l'aspect du Machine Learning pour la couche application réseau. Une liste non exhaustive des recherches en cours sur le sujet est à disposition dans les notes bibliographiques sur des thèses françaises actuelles qui me semble intéressantes, mais l'ONF intégrant une équipe de recherche et un laboratoire c'est plutôt de ce côté que tout se passe et ils publient régulièrement leurs recherches et leurs avancées dans leur blog que je n'ai pas eu le temps de consulter. Cela dit, je n'ai pas beaucoup vu d'approches sur les problèmes de sécurité et de fiabilité… A vérifier.

Nick McKeown dans son Keynote de l'ONF Connect 2019 [1] nous annonce des objectifs très ambitieux et une vision du monde du réseau pour 2030 comme la disparition des protocoles très osée… (Non, ils n'auront pas disparu mais il ne seront juste plus visible par les applications réseaux qui permettront de gérer le réseau). Mais en même temps ces propos me semblent très actuelle, son but ultime est de programmer toute l'infrastructure du réseau pour que de plus en plus nous n'intervenions qu'au minimum et à un niveau élevé. Si nous reprenons son idée de départ : nous n'avons pas besoin de savoir comment fonctionne un ordinateur pour se servir de ses fonctionnalités et cela même si elle gère l'ordinateur ! Et c'est cette même idée, paradigme, architecture qu'il reprend pour créer le réseau de demain. Comme a dit Lavoisier « Rien ne se perd, rien ne se crée, tout se transforme ». Ou alors tout se répète ? En tout cas, même si la route semble longue et les défis de tailles, ils me semblent à la pointe des technologies actuelles et très motivants.

5. Bibliographie

5.1 Liens internet pour les traducteurs

Quand il me semblait important de ne pas dénaturer les mots de la langue d'origine, notamment pour les définitions, j'ai mis la traduction française dans le texte et la citation d'origine en bas de page. Les traductions notées par des guillemets peuvent être issues de mes propres connaissances, ou de l'un des sites ci-dessous voir d'un mixte afin de privilégier la fiabilité de la traduction.
Pour les mots et le vocabulaire technique : https://www.linguee.com/
Pour la traduction de texte :
https://www.deepl.com/translator
https://translate.google.fr/?hl=fr

5.2 Bibliographie pour cette étude

La bibliographie est présentée dans l'ordre d'apparence dans le texte. C'est ce qui me semble le plus facile à la lecture de ce document et ce qui est le plus fréquent dans les lectures que j'ai pu faire. J'ai utilisé le standard IEEE spécialisé en informatique.

[1] O. N. Foundation, *How We Might Get Humans Out of the Way - Keynote by Nick McKeown, Professor at Stanford University - ONF Connect 19*. 2019.
[2] « Commutateur réseau », *Wikipédia*. mai 05, 2020, Consulté le: mai 06, 2020. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Commutateur_r%C3%A9seau&oldid=170461579.
[3] « About the Open Networking Foundation | Mission, Members, Training, Partners », *Open Networking Foundation*. https://www.opennetworking.org/mission/ (consulté le juin 03, 2020).
[4] « P4 (langage) », *Wikipédia*. avr. 20, 2020, Consulté le: mai 06, 2020. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=P4_(langage)&oldid=169804676.
[5] « P4Runtime ». https://p4.org/p4-runtime/ (consulté le juin 07, 2020).
[6] « Pile de protocoles », *Wikipédia*. juill. 14, 2019, Consulté le: mai 06, 2020. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Pile_de_protocoles&oldid=160903388.
[7] « Plan de commutation », *Wikipédia*. nov. 07, 2016, Consulté le: mai 06, 2020. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Plan_de_commutation&oldid=131620447.
[8] « Control plane », *Wikipedia*. avr. 11, 2020, Consulté le: mai 08, 2020. [En ligne]. Disponible sur: https://en.wikipedia.org/w/index.php?title=Control_plane&oldid=950258695.
[9] « NG-SDN Definition », *Open Networking Foundation*. https://www.opennetworking.org/ng-sdn/ (consulté le mai 07, 2020).
[10] « Software-Defined Networking (SDN) Definition », *Open Networking Foundation*. https://www.opennetworking.org/sdn-definition/ (consulté le mai 11, 2020).
[11] « Qu'est-ce que la virtualisation des fonctions réseau ? » https://www.redhat.com/fr/topics/virtualization/what-is-nfv (consulté le juin 07, 2020).
[12] « Forwarding information base », *Wikipedia*. juin 02, 2020, Consulté le: juin 03, 2020. [En ligne]. Disponible sur: https://en.wikipedia.org/w/index.php?title=Forwarding_information_base&oldid=960402707.
[13] « Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions », *Open Networking Foundation*. https://www.opennetworking.org/onos/ (consulté le juin 07, 2020).
[14] L. K. T. O. at Plv. diverse technical background formed at the intersection of legacy networking approach, SDN, L. is leading Plv. member contribution to the O. community, et I. C. T. D. N. Innovation, « Solving SDN Challenges with a Single Solution: Stratum », *PLVision*, juin 13, 2019. https://plvision.eu/rd-lab/blog/sdn/sdn-solution-stratum (consulté le juin 07, 2020).
[15] « In-band Network Telemetry (INT) Specifications ». https://p4.org/assets/INT-current-spec.pdf (consulté le juin 08, 2020).
[16] « NG-SDN Tutorial @ ONF Connect 2019 », Consulté le: mai 08, 2020. [En ligne]. Disponible sur: https://docs.google.com/presentation/d/1zFazmS2bXwbpLf8dOPQCoI7c8n-ffgz3zL8omBH_VoE/edit?usp=embed_facebook.
[17] « OpenConfig - Home ». https://www.openconfig.net/ (consulté le juin 08, 2020).
[18] « gRPC », *Wikipédia*. janv. 10, 2019, Consulté le: juin 08, 2020. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=GRPC&oldid=155674844.
[19] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, et S. Uhlig, « Software-Defined Networking: A Comprehensive Survey », *Proc. IEEE*, vol. 103, nº 1, p. 14-76, janv. 2015, doi: 10.1109/JPROC.2014.2371999.
[20] N. McKeown *et al.*, « OpenFlow: enabling innovation in campus networks », *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, nº 2, p. 69–74, mars 2008, doi:

10.1145/1355734.1355746.
[21] N. McKeown, « Software-defined Networking », avr. 2009, Consulté le: mai 11, 2020. [En ligne]. Disponible sur: https://www.cs.odu.edu/~cs752/papers/sdr-infocom_brazil_2009_v1-1.pdf.
[22] « ONF SDN Projects », *Open Networking Foundation*. https://www.opennetworking.org/onf-sdn-projects/ (consulté le mai 25, 2020).
[23] I. Choukri, M. OUZZIF, et K. Bouragba, « Software Defined Networking (SDN): Etat de L'art », in *Colloque sur les Objets et systèmes Connectés*, CASABLANCA, Morocco, juin 2019, Consulté le: mai 11, 2020. [En ligne]. Disponible sur: https://hal.archives-ouvertes.fr/hal-02298874.
[24] M. BOUCADAIR et C. JACQUENET, « SDN : promesses et enjeux - Analyse de l'ingénierie d'exploitation de réseaux », *Ref : TIP382WEB - « Réseaux Télécommunications »*, nov. 10, 2013. http://www.techniques.ingenieur.fr/base-documentaire/technologies-de-l-information-th9/administration-de-reseaux-applications-et-mise-en-oeuvre-42481210/sdn-promesses-et-enjeux-te7608/ (consulté le mai 25, 2020).
[25] « SDN Technical Specifications », *Open Networking Foundation*. https://www.opennetworking.org/software-defined-standards/specifications/ (consulté le juin 04, 2020).
[26] P. Bosshart *et al.*, « P4: programming protocol-independent packet processors », *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, nº 3, p. 87–95, juill. 2014, doi: 10.1145/2656877.2656890.
[27] C. Cascone, « P4 and P4Runtime Technical Introduction and Use Cases for the Next-Gen SDN Stack », avr. 2019, Consulté le: juin 07, 2020. [En ligne]. Disponible sur: https://static.sched.com/hosted_files/onsna19/b3/P4_ONOS_talk_ONS-NA_19.pdf.
[28] « P4Runtime Specification ». https://p4.org/p4runtime/spec/v1.0.0/P4Runtime-Spec.html (consulté le juin 08, 2020).
[29] Sloane et al. et ONF, « Next-Gen SDN », Consulté le: juin 07, 2020. [En ligne]. Disponible sur: https://www.opennetworking.org/wp-content/uploads/2019/09/Connect-2019-NG-SDN.pdf.
[30] « Stratum - ONF Launches Major New Open Source SDN Switching Platform with Support from Google », *Open Networking Foundation*, mars 12, 2018. https://www.opennetworking.org/news-and-events/press-releases/stratum-onf_google-launches-major-new-open-source-sdn-switching-platform-with-support-from-google/ (consulté le juin 07, 2020).
[31] « Software-Defined Networks: A Systems Approach — Software-Defined Networks: A Systems Approach Version 0.3-dev documentation ». https://sdn.systemsapproach.org/index.html (consulté le juin 08, 2020).

5.3 Notes bibliographiques sur les recherches actuelles

Ces bibliographies et notes sont sur des thèses actuelles qui me semble intéressantes. Il existe, me semble-t-il peu de recherche sur le machine Learning, j'ai relevé cette thèse en cours :

- Sehaki (thèse en cours) sur « SD-SON: réseau de recouvrement spécifique au service basé sur SDN » portant sur les IDN (Intelligence Defined Réseaux)

Sinon, la tendance est à travailler sur les couches hautes du SDN :

- Jean-Michel SANNER (2020) dans « Architecture du plan de contrôle SDN et placement de services réseaux dans les infrastructures des opérateurs » fait un état des lieux intéressant des différentes architectures SDN et propose une architecture plus adaptée aux opérateurs avec un plan de contrôle qui dépasse les limites d'Openflow….

- Abhashkumar et Al. (2017) propose d'augmenter de 50% l'efficacité des pipelines des switch en utilisant le P5, un système qui exploite la connaissance des déploiements d'applications intégrés dans une abstraction de politique de haut niveau. « P5: Policy-driven optimization of P4 pipeline »
- Antichini et Rérvári (2020) « Full-stack SDN : The next Big Challenge? » propose d'envisager un contrôler central au niveau de la couche application, remettant en cause toute l'attention portée sur les couches d'infrastructure et de système d'exploitation. Ils encouragent une nouvelle infrastructure logicielle (reposant sur des services mesh) et une continuité sur cet axe de recherche…
- Stoenescu et al (2018) propose un outil pour débugger P4 « Debugging P4 programs with Vera »
- Cihat Baktir et al. (2017) propose d'envisager une intéraction entre le Edge Computing et le SDN à travers des discussions sur leurs propriétés et sur un model d'architectute mise en place. Un Aspect peut-être intéressant pour le monde de l'IoT ?...
- Soni (2018) « Towards network softwarization : a modular approach for network control delegation » propose une approche du SDN et du NFV dans le but de répondre aux problématiques du multicast.
- Aouadj (2020) introduit un nouveau langage de programmation Airnet dans l'API Nourthbound « AirNet : le modèle de virtualisation « Edge-Fabric » comme plan de contrôle pour les réseaux programmables »
- Nazih Salhab (thèse en cours) « Provisionnement des ressources et optimisation dynamique des slices réseaux dans un environnement SDN/NFV »

6. Résumé et mots-clés

6.1 Résumé

Le Software-Defined Network a pour ambition de devenir une administration du réseau centralisée avec une gestion globale, intelligente, sécurisé, optimisé voir automatisé du réseau où l'intervention humaine ne se fera que pour son minimum.
C'est l'évolution vers cet SDN que nous traitons ici à travers deux temps forts :

- La première version du SDN qui sépare du plan de contrôle du plan de commutation et l'innovation OpenFlow, une infrastructure qui a permis d'abstraire partiellement l'équipement.
- La Prochaine génération SDN découpée en deux phases.
    - La phase 1 qui complète totalement l'abstraction de l'équipement grâce aux infrastructures P4 et P4Runtime encourageant ainsi la propagation de la programmation à toutes les strates du domaine du réseaux.
    - La phase 2, prévue pour 2030 et qui devrait compléter l'objectif ci-dessus, introduit les différentes infrastructures qui permettront l'indépendance des réseaux programmables avec la télémétrie, le CI/CD, la conteneurisation et l'orchestration et bine plus…

6.2 mots-clés

Software-defined Networking (SDN), plan de contrôle, plan de commutation, OpenFlow, réseau programmable, Language programmable, P4, P4Runtime, network virtualization, Network Function Virtualization (NFV)

7. Pitch and Keywords

7.1 Pitch

The Software-Defined Network aims to become a centralized network administration with a global, intelligent, secure, optimized and even including an automated management of the network where human intervention will be done to its minimum.
This is the evolution towards which we are dealing with here through two highlights:

- The first version of the SDN which separates the control plane from the data plane and innovate OpenFlow, an infrastructure which made it possible to partially abstract the equipment.
- The Next Generation SDN split into two phases:
    - Phase 1, which completely completes the abstraction of the equipment thanks to the P4 and P4Runtime infrastructures, thus encouraging the spread of programming to all the layers of the network.
    - Phase 2, planned for 2030 and which should complete the above objective, introduces the various infrastructures which will allow the independence of programmable networks with telemetry, CI/CD, containerization, orchestration and much more...

7.2 Keywords

Software-defined Networking (SDN), Control plane, Data plane, OpenFlow, Programmable networks, Programmable languages, P4, P4Runtime, network virtualization, Network Function Virtualization (NFV)