Software Defined Networking (SDN): Etat de L'art

Ihssane Choukri, Mohammed Ouzzif, Khalid Bouragba

Software Defined Networking (SDN): Etat de L'art

Ihssane Choukri, Mohammed Ouzzif, Khalid Bouragba

Laboratoire RITM, ESTC, Université Hassan II, Casablanca, Maroc

Email : choukriihssan@gmail.com, ouzzif@est-uh2c.ac.ma, bouragba2008@gmail.com

RESUME

Internet a connu un énorme succès, Il est devenu un outil universel indispensable pour les entreprises et la plupart d'individus. Cependant, malgré leur adoption, les réseaux classiques sont complexes et difficiles à gérer. Une des raisons de cette difficulté réside dans l'architecture des réseaux actuels où le plan de contrôle et le plan de données sont intégrés verticalement dans chaque équipement réseau.

SDN est un nouveau paradigme reseau, qui permet de simplifier la gestion et l'innovation dans le réseau, en séparant la logique de contrôle du réseau des équipements d'interconnexions ,en promouvant la centralisation du contrôle et la capacité de programmer le réseau. Dans cet article, nous présentons une vue générale sur SDN. Nous commençons par présenter SDN, son architecture, et ses interfaces de communications. Nous décrivons par la suite le protocole Openflow, son fonctionnement, et les principaux contrôleurs SDN. Nous examinons également les problèmes confrontées par SDN, en nous concentrant sur les principaux défis de plan de contrôle tels que la performance, la scalabilité, la sécurité, et la fiabilité, nous discutons ainsi, les solutions existantes afin de surmonter ces défis.

Mots clés : Software-Defined Networking, Openflow, Réseaux programmables, plan de contrôle, plan de données.

1 INTRODUCTION

Avec l'avènement de l'internet et les nouvelles technologies de l'information comme le Big Data qui nécessite un traitement distribué, le Cloud Computing, ou encore l'internet des objets (IoT), les architectures réseaux classiques constituent un grand défi, tant pour les opérateurs que pour les administrateurs réseaux.

En effet, depuis plusieurs années, il est très difficile, voire impossible d'innover ou apporter des changements au réseau. Selon [1] , la conception ou le déploiement d'un nouveau protocole de routage peuvent prendre de 5 à 10 années. Aussi les taches de configuration et de gestion des réseaux sont plus complexes. Une des raisons de cette difficulté d'évoluer, ou d'administrer simplement les réseaux, est le fort couplage qui existe entre le plan de contrôle et le plan de données des équipements d'interconnexions dans les architectures des réseaux actuels. C'est dans ce contexte qu'a apparu le concept des réseaux définis par les logiciels (Software Defined Networking ou SDN), afin de répondre à la rigidité architecturale des réseaux actuels, notamment en les rendant plus programmables.

L'idée principale de ce nouveau paradigme, est de sortir la partie intelligente des équipements d'interconnexions, et la placer vers un seul point de contrôle appelé contrôleur, ce dernier fournit une vue centrale de reseau, ce qui simplifie d'une part, la gestion et la configuration de réseau.

Le SDN présente donc plusieurs avantages, il peut servir plusieurs domaines, et être intégré avec les nouvelles technologies, tels que Big Data, Machine Learning, 5G, IoT, et les Smart Cities , en offrant ainsi une programmabilité et une vue globale, centralisé du reseau [2] .

Par exemple la capacité de programmation de SDN est particulièrement utile pour les applications Big Data nécessitant de nombreuses reconfigurations [3]. Le SDN améliore aussi la résilience et la scalabilité du réseau, qui sont essentielles pour le déploiement de l'IoT à grande échelle, tel que les Smart Cities. Selon [4] SDN peut jouer également un rôle crucial dans la conception des réseaux sans fil 5G.

Il est en outre envisageable d'utiliser le SDN avec différentes approches, afin d'améliorer les performances des réseaux, par exemple l'utilisation de Machine Learning avec SDN permet de fournir plus d'intelligence aux réseaux, et cela grâce aux capacités du SDN.

Malgré ses avantages, et sa capacité de simplifier les réseaux, SDN rencontre des défis qui peuvent limiter ses fonctionnalités et ses performances dans les réseaux à grande échelle. Cet article aborde les principaux défis du SDN de manière complète et détaillée. Il traite les problèmes de performance, de scalabilité, de sécurité, et de fiabilité, au niveau des contrôleurs SDN.

De nombreux travaux ont porté sur le sujet de SDN, vu son importance dans le domaine des réseaux. Plusieurs articles récents [1] , [5] ,[6], fournissent des études intéressantes sur SDN. Notre objectif dans cet article est un peu différent, nous visons à donner au lecteur particulièrement intéressé par les réseaux SDN, un aperçu sur le SDN, ainsi la possibilité d'intégrer ce nouveau paradigme avec les nouvelles technologies, pour objectif d'une part, d'améliorer, et simplifier le déploiement de ces technologies en utilisant SDN, et

d'autre part profiter de ces technologies afin de surmonter les défis de SDN.
Le reste de l'article est organisé comme suit. Dans la section 2, nous présentons le SDN, son architecture, et ses interfaces de communications. Ensuite, nous décrivons dans la section 3, le protocole Openflow, son fonctionnement, et les principaux contrôleurs SDN. Dans la section 4, nous présentons quelques défis de SDN, et les solutions récentes proposées pour surmonter ces défis. Dans la section 6, nous concluons notre papier avec quelques perspectives.

2 SOFTWARE-DEFINED NETWORKING

2-1. C'est quoi SDN ?

Le SDN est un nouveau paradigme qui décrit une architecture réseau dont le plan de contrôle est totalement découplé de plan de données. Selon l'ONF (Open Network Fondation) [7] SDN est une architecture qui sépare le plan de contrôle du plan de données, et centralise toue l'intelligence de reseau dans une entité programmable appelé «Contrôleur», afin de gérer plusieurs éléments du plan de données(Ex switches ou routeurs, etc.) via des APIs (Application Programming Interface).
=>Plus concrètement, on peut dire qu'une architecture réseau suit le paradigme SDN si, et seulement si, elle vérifie les points suivants :

-**Le plan de contrôle est complètement découplé du plan de données**, cette séparation est matérialisée à travers la définition d'une interface de programmation (Southbound API)

-**Toute l'intelligence du réseau est externalisée dans un point logiquement centralisé appelé contrôleur SDN**, ce dernier offre une vue globale sur toute l'infrastructure physique.

-**Le contrôleur SDN est un composant programmable** qui expose une API (NorthboundAPI) pour spécifier des applications de contrôle.

2-2. Architecture de SDN

Un réseau traditionnel est composé généralement des équipements d'interconnexions tels que des switchs et des routeurs. Ces équipements incorporent à la fois la partie transmission et la partie de contrôle de reseau. Dans ce modèle d'architecture, il est difficile de développer de nouveaux services, en raison du fort couplage qui existe entre le plan de contrôle et le plan de transmission.
Afin d'ouvrir les équipements réseaux aux innovations, l'architecture SDN, a vu le jour. Elle permet de découpler la partie de contrôle de la partie transmission des équipements d'interconnexions. Le SDN est composée principalement de trois couches et d'interfaces de communication (Figure 1), nous décrivons dans ce qui suit ces couches, ainsi que les interfaces de communications :

-La couche de transmission : appelée aussi «plan de données», elle est composée des équipements d'acheminement tels que les switches ou les routeurs, son rôle principal est de transmettre les données, et collecter les statistiques.

-La couche de contrôle : appelée aussi «plan de contrôle», elle est constituée principalement d'un ou plusieurs contrôleurs SDN, son rôle est de contrôler et de gérer les équipements de l'infrastructure à travers une interface appelée 'south-bound API'.

-La couche application : représente les applications qui permettent de déployer de nouvelles fonctionnalités réseau, comme l'ingénierie de trafic, QoS, la sécurité, etc. Ces applications sont construits moyennant une interface de programmation appelée 'north-bound API

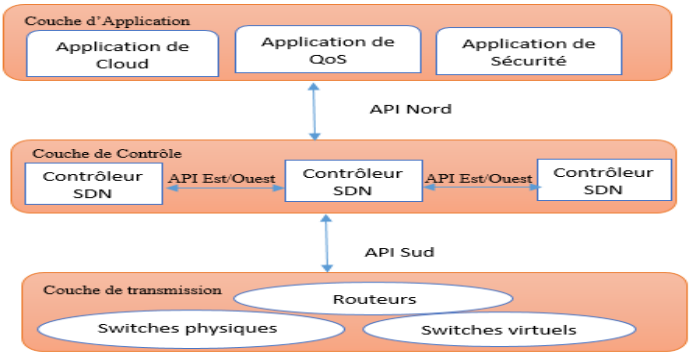

Figure 1 : Architecture SDN

2-3. Interfaces de communications

Il existe principalement trois types d'interfaces permettent aux contrôleurs de communiquer avec leur environnement : interface Sud, Nord et Est/Ouest

✓ **Interfaces Sud**

Les interfaces Sud ou (Southbound APIs) représentent les interfaces de communication, qui permettent au contrôleur SDN d'interagir avec les équipements de la couche d'infrastructure, tel que les switches, et les routeurs.
Le protocole le plus utilisé, et le plus déployé comme interface Sud est le protocole OpenFlow, qui a été standardisé par l'ONF, sa dernière version est 1.5 [8], plus de détails sur ce protocole sera donnée dans la prochaine section . Il existe dorénavant d'autres alternatives d'interface Sud, tels que ForCes [9], ou Open vSwitch Database (OVSDB) [10] , mais le protocole openflow est actuellement le standard de facto, qui est largement accepté et répandu dans les réseaux SDN.

✓ **Interfaces Nord**

Les interfaces Nord servent à programmer les équipements de transmission, en exploitant l'abstraction du réseau fourni par le plan de contrôle. Il est noté que

contrairement à la Southbound API qui a été standardisé, l'interface nord reste encore une question ouverte.
Bien que la nécessité d'une telle interface standardisée constitue un débat considérable au sein de l'industrie, l'avantage d'une API nord ouverte est aussi important, une API nord ouverte permette plus d'innovation et d'expérimentation. Plusieurs implémentations de cette interface existent, chaque 'une de ces implémentations offre des fonctionnalités bien différents. Le RESTful [11] considéré comme l'API nord le plus répandue dans les réseaux SDN.

✓ **Interfaces Est/Ouest**

Les interfaces Est/Ouest sont des interfaces de communication qui permettent la communication entre les contrôleurs dans une architecture multi-contrôleurs pour synchroniser l'état du réseau [12]. Ces architectures sont très récentes et aucun standard de communication inter-contrôleur n'est actuellement disponible.

3 OPENFLOW

3-1. Définition d'Openflow

Openflow est le protocole utilisé pour la communication entre la couche transmission et la couche de contrôle, il a été initialement proposé et implémenté par l'université de Stanford, et standardisé par la suite par l'ONF, sa dernière version est 1.5 [8]. Nous détaillons par la suite la structure d'openflow, son fonctionnement, ses différentes spécifications, ainsi que quelques contrôleurs openflow.

3-2. Architecture Openflow

L'architecture openflow est l'implémentation réelle des réseaux SDN, Cette architecture est basée principalement sur trois composantes : le plan de données, qui est composée des switches openflow ; le plan de contrôle, constitué par des contrôleurs OpenFlow ; une chaîne sécurisée qui permettent aux commutateurs de se connecter au plan de contrôle.
La spécification d'un commutateur openflow est standardisée par l'ONF. Selon la spécification d'ONF [13], un commutateur openflow doit contenir un ou plusieurs tables de flux , ces tables de flux contiennent plusieurs d'entrées qui correspondent à des règles, où chacune est constituée principalement des trois champs suivants (Tableau 1):
-L'En-tête de paquet : il définit le flux de données, il contient les informations nécessaires pour déterminer le paquet auquel cette règle sera appliquée. L'en-tête de paquet peut identifier différents protocoles tel qu'Ethernet, IPv4, IPv6 ou MPLS, cela dépend de la spécification d'openflow déployée.

-L'Action : spécifie comment les paquets d'un flux seront traités. Une action peut être l'une des suivantes : transférer le paquet vers un ou plusieurs ports, supprimer le paquet , transférer le paquet vers le contrôleur, ou modifier le champ d'entête de paquet [6].

-Les Compteurs : sont réservés à la collecte des statistiques de flux. Ils enregistrent le nombre de paquets et d'octets reçus de chaque flux, et le temps écoulé depuis le dernier transfert de flux.

Champs d'en-tête	Compteurs	Actions

Tableau 1 : *Structure d'une entrée de table de flux d'un commutateur openflow 1.0*

3-3. Fonctionnement Openflow

Lorsqu'un paquet arrive à un commutateur, le commutateur vérifie s'il y a une entrée dans la table de flux qui correspond à l'en-tête de paquet. Si c'est le cas, le commutateur exécute l'action correspondante dans la table de flux. Dans le cas contraire, c'est-à-dire il y a pas une entrée correspondante (1) , le commutateur génère un message asynchrone vers le contrôleur (2) sous la forme d'un 'Packet_in', puis le contrôleur décide selon sa configuration une action pour ce paquet, et envoie une nouvelle règle de transmission sous la forme d'un 'Packet_out' et 'Flow-mod' au commutateur (3), et enfin, la table de flux du commutateur est actualisée, pour prendre en compte la nouvelle règle installé par le contrôleur (4). La Figure 2 [14] décrit le processus de transmission d'un paquet avec openflow.

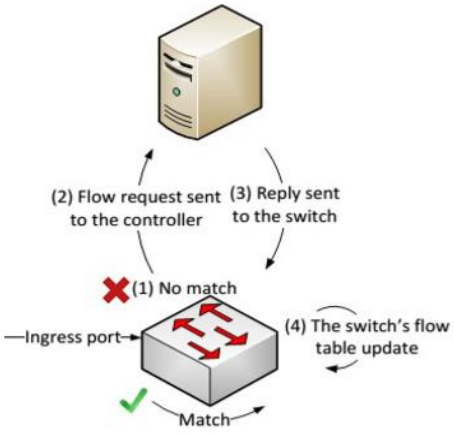

Figure 2 : *processus de transmission d'un paquet avec openflow [14]*

L'échange d'informations entre le commutateur et le contrôleur s'effectue par l'envoi de messages via un canal de contrôle sécurisé en utilisant TLS (Transport Layer Security).

3-4. Les Contrôleurs communes d'OpenFlow

Plusieurs contrôleurs ont été développés, dont la majorité sont open source et supportent le protocole

openflow. Le tableau 2 présente les contrôleurs SDN les plus connus.

Contrôleur	Organisation	Langage	Fonctionnalités
NOX [31]	Nicira	C++	le premier contrôleur openflow
POX [15]	Nicira	Python	améliorer les performances de NOX
Ryu [16]	NTT, OSRG group	Python	supporte l'OpenStack
Beacon[17]	Stanford	Java	basé sur le Multithreading
Floodlight [35]	Big Switch	Java	testé avec des commutateurs OpenFlow physiques et virtuels.
Opendaylight[36]	Linux Foundation	Java	supporte le Framework OSGi et le REST API

Tableau 2 : *Quelques contrôleurs SDN les plus connus.*

4. LES DÉFIS SDN

Le SDN a connue plusieurs défis que ce soit sur le plan de données que sur le plan de contrôle , mais dans cette section on va présenter les principaux défis de SDN , au niveau de son plan de contrôle, qui comprend la performance, la scalabilité , la sécurité, et la fiabilité [6]. La section 5 décrit ces défis et présente les solutions récentes proposées dans la littérature.

4-1. Performance

Les performances des contrôleurs SDN constituent un domaine important que les chercheurs essayent toujours de l'améliorer, et puisque le SDN est une technique basée sur les flux, ses performances sont mesurées en fonction de deux métriques : le temps nécessaire pour instaurer un nouveau flux dans les commutateurs (latence) et le nombre de flux que le contrôleur peut traiter par seconde (débit).

Des chercheurs ont utilisé la technique de multithreading, afin d'améliorer les performances des contrôleurs SDN. En effet le premier contrôleur développé pour gérer les réseaux SDN est le contrôleur NOX [22], ce contrôleur a un débit de 30000 flux par seconde et une latence de 10 ms. Un nouveau contrôleur multithreads, appelé NOXMT (version améliorée du contrôleur NOX) [23] a été introduit par les auteurs, afin d'améliorer les performances des contrôleurs. NOX-MT permet d'améliore le débit de contrôleur de plus de 30 fois, en le comparant avec le contrôleur NOX.

L'utilisation des contrôleurs multiples est une solution plus efficace pour améliorer les performances des contrôleurs SDN. Différentes architectures de plusieurs contrôleurs SDN ont été proposées dans la littérature. Les solutions tels que HyperFlow [24], Onix [25] Opendaylight (ODL)[26] déploient un plan de contrôle logiquement centralisé, ou plusieurs contrôleurs sont utilisés, en partageant entre eux les charges et synchronisant les données, ce qui permet d'améliorer les performances et assurer la cohérence de réseau.

D'autres solutions ont été proposées afin de permettre la communication entre plusieurs domaines de reseau SDN. CIDC [28] est une nouvelle interface proposée, afin de permettre la communication inter-contrôleurs pour les plans de contrôle logiquement distribués. Cette interface est implémentée dans chaque contrôleur et fonctionne selon des modes de communication ce qui améliore les performances des contrôleurs SDN.

4-2. Scalabilité

Un autre défi du SDN est la scalabilité ou l'évolutivité du réseau, plus la taille de réseau augmente, plus des demandes sont envoyés au contrôleur et à un moment donné, le contrôleur devient incapable de traiter toutes ces demandes.

Des solutions ont été proposés par les auteurs, afin d'améliorer la scalabilité des contrôleurs SDN, L'une des méthodes utilisé afin de surmonter le problème de scalabilité, consiste à un niveau de parallélisme supérieur dans les systèmes multi-cœurs. Tootoonchian et al.[23] ont montré que des simples modifications apportées au contrôleur NOX, augmentent ses performances de plus de 10 fois.

Une solution viable pour surmonter les problèmes de scalabilité est proposée dans «DIFANE» [29]. Il s'agit d'une solution qui permet de conserver de manière proactive, tout le trafic dans le plan de données, en dirigeant les paquets via des commutateurs intermédiaires stockant les règles nécessaires. Une autre solution pour améliorer la scalabilité des contrôleurs SDN, est l'utilisation des contrôleurs multiples, des solutions permettant de distribuer physiquement les contrôleurs SDN, tout en maintenant la vue globale de réseau. Onix [25] par exemple, est une plate-forme de contrôle distribuée, qui fournit aux applications de contrôle un ensemble d'API, facilitant l'accès à l'état du réseau (NIB) qui est distribué sur des instances Onix.

4-3. Sécurité

Le SDN peut poser également des problèmes de sécurité. Le fait de centraliser toute l'intelligence du reseau dans un seul contrôleur peut accroître la vulnérabilité du contrôleur. Un contrôleur SDN représente le point critique de reseau, s'il est compromis ou devient indisponible, tous les aspects du réseau seront endommagés.

Les réseaux SDN sont soumis à divers problèmes de sécurité tels que le déni de service [31], l'usurpation d'identité, l'élévation des privilèges, la falsification, et la répudiation . Des solutions ont été proposées par les auteurs pour améliorer la sécurité de SDN. Parmi les solutions proposées dans le contrôle d'accès : AuthFlow [32] est un mécanisme d'authentification et de contrôle d'accès basé sur les informations d'identification de l'hôte, permet de refuser l'accès aux hôtes non autorisés.

Un certain nombre de solutions ont été proposées pour surmonter l'attaque par déni de service sur les contrôleurs SDN ou sur les tables de flux des commutateurs. Les auteurs proposent la solution AVANT-GUARD [35] qui limite les demandes de flux envoyées au plan de contrôle à l'aide d'un outil de migration de connexion.

4-4. Fiabilité

Dans les premiers déploiements de réseaux SDN qui utilisaient un seul contrôleur centralisé, responsable de tout le réseau, cela pose des problèmes sur les contrôleurs SDN, qui deviennent des points uniques de défaillance (SPOF, Single Point Of Failure) ; Si par exemple le contrôleur tombe en panne ou devient défaillant, tout le réseau devient indisponible. Plusieurs solutions ont été proposés afin d'améliorer la fiabilité des contrôleurs SDN.

Obadia et al. proposent deux mécanismes pour détecter les défaillances dans les contrôleurs: un algorithme connue sous le nom de mécanisme de découverte des commutateurs(Greedy Algorithm), et une méthode de pré-partitionnement entre contrôleurs (Pre-Partitioning Failover ou PPF) [37].

Dans la première méthode, lorsqu'un switch détecte la défaillance du contrôleur, il envoie un paquet LLDP indiquant qu'il n'a pas de contrôleur maître, puis le contrôleur qui reçoit ce paquet deviendra le maître et ajoutera ensuite le switch orphelin à son domaine. Dans la deuxième méthode, chaque contrôleur calcule une liste des contrôleurs qui peuvent prendre le contrôle des commutateurs de son domaine en cas de défaillance, puis chaque 'un envoyera sa liste aux contrôleurs voisins.

Chen et al. proposent FCF-M (Fast Controller Failover for Multi-domain SDNs) [38] . Dans cette approche chaque domaine est géré par un contrôleur principal et un contrôleur de backup. La défaillance d'un contrôleur est détectée à l'aide d'un mécanisme heartbeat circulaire ; chaque contrôleur successeur vérifie la disponibilité de son prédécesseur, en envoyant des messages heartbeats d'une façon circulaire. Lorsque le contrôleur successeur détecte une panne de son prédécesseur, il sélectionne un contrôleur en fonction de la distance et de la charge, puis lui affecte localement les commutateurs orphelins.

Moazenni et al. proposent une méthode RDSDN (Reliable Distributed SDN) [39] pour améliorer la tolérance aux pannes des contrôleurs SDN. Une architecture de plusieurs contrôleurs distribués est utilisée, ou chaque contrôleur est responsable d'un sous-réseau en tant que master, et définie comme esclave pour les autres sous-réseaux. Une nouvelle formule est proposée pour calculer le taux de fiabilité de chaque sous-réseau. Ainsi les taux de fiabilité calculés sont partagés entre les contrôleurs, afin de sélectionner le contrôleur ayant la valeur de fiabilité la plus grande, ce dernier est choisie comme coordinateur pour le réseau. Dans RDSDN, la défaillance des contrôleurs est détectée par le coordinateur, qui décide quel autre contrôleur est le plus approprié pour prendre en charge le sous-réseau dont son contrôleur a tombé en panne.

5. CONCLUSION ET PERSPECTIVES

Dans cet article, nous avons fourni une vue générale sur SDN (Software Defined Networking), notamment les avantages et l'architecture de SDN. Ensuite nous avons présenté le protocole OpenFlow, son fonctionnement, et quelques contrôleurs SDN. Ensuite nous avons présenté quelques défis de SDN, et les solutions récentes proposées pour surmonter ces défis. Enfin, vu l'importance de machine learning qui ont suscité beaucoup d'intérêt dans de nombreux efforts de recherche. Déployer plus d'intelligence peut être une solution efficace afin de résoudre les problèmes de SDN. Dans ce contexte peut de travaux qui se concentrent sur l'application de machine learning dans le domaine SDN, pour combler cette lacune, nous prévoyant dans les travaux futurs de faire une discussion sur la possibilité d'appliquer les techniques de machine learning, afin de surmonter les principaux défis de SDN notamment la performance, la scalabilité, la sécurité, et la fiabilité. Nous espérons que notre discussion ouvrira une nouvelle voie aux lecteurs pour la mise en place d'un réseau intelligent, et d'encourager davantage les études ultérieures sur ce sujet.

Bibliographie

[1] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, et S. Uhlig, « Software-Defined Networking: A Comprehensive Survey », *Proc. IEEE*, vol. 103, nº 1, p. 14-76, janv. 2015.

[2] P. C. da R. Fonseca et E. S. Mota, « A Survey on Fault Management in Software-Defined Networks », *IEEE Commun. Surv. Tutor.*, vol. 19, nº 4, p. 2284-2321, 2017.

[3] L. Cui, F. R. Yu, et Q. Yan, « When big data meets software-defined networking: SDN for big data and big data for SDN », *IEEE Netw.*, vol. 30, nº 1, p. 58-65, janv. 2016.

[4] A. Hakiri et P. Berthou, « Leveraging SDN for The 5G Networks: Trends, Prospects and Challenges », p. 23.

[5] W. Xia, Y. Wen, C. H. Foh, D. Niyato, et H. Xie, « A Survey on Software-Defined Networking », *IEEE Commun. Surv. Tutor.*, vol. 17, nº 1, p. 27-51, 2015.

[6] M. Jammal, T. Singh, A. Shami, R. Asal, et Y. Li, « Software defined networking: State of the art and research challenges », *Comput. Netw.*, vol. 72, p. 74-98, oct. 2014.

[7] « Software-Defined Networking (SDN) Definition », *Open Networking Foundation*. [En ligne]. Disponible sur: https://www.opennetworking.org/sdn-definition/. [Consulté le: 19-nov-2018].

[8] « openflow-switch-v1.5.1.pdf ». [En ligne]. Disponible sur: https://www.opennetworking.org/wp-

content/uploads/2014/10/openflow-switch-v1.5.1.pdf. [Consulté le: 29-nov-2018].
[9] J. Halpern et J. Hadi, « Forwarding and Control Element Separation (ForCES) Forwarding Element Model », RFC Editor, RFC5812, mars 2010.
[10] B. Pfaff et B. Davie, « The Open vSwitch Database Management Protocol », RFC Editor, RFC7047, déc. 2013.
[11] R. T. Fielding, « in Information and Computer Science », p. 180, 2000.
[12] [En ligne]. Disponible sur: https://www.ietf.org/archive/id/draft-yin-sdn-sdni-00.txt. [Consulté le: 30-nov-2018].
[13] « SDN Technical Specifications », *Open Networking Foundation*. [En ligne]. Disponible sur: https://www.opennetworking.org/software-defined-standards/specifications/. [Consulté le: 29-nov-2018].
[14] F. Benamrane, M. Ben mamoun, et R. Benaini, « Performances of OpenFlow-Based Software-Defined Networks: An overview », *J. Netw.*, vol. 10, nº 6, juin 2015.
[15] N. O. X. Repo, *The POX network software platform. Contribute to noxrepo/pox development by creating an account on GitHub*. 2018.
[16] « What is Ryu Controller? - SDxCentral ». [En ligne]. Disponible sur: https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/what-is-ryu-controller/. [Consulté le: 24-déc-2018].
[17] D. Erickson, « The beacon openflow controller », in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, Hong Kong, China, 2013, p. 13.
[18] « Les réseaux SDN transforment le Big Data en capital informationnel ». [En ligne]. Disponible sur: https://www.decideo.fr/Les-reseaux-SDN-transforment-le-Big-Data-en-capital-informationnel_a6699.html. [Consulté le: 31-déc-2018].
[19] S. K. Routray et K. P. Sharmila, « Software defined networking for 5G », in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, 2017, p. 1-5.
[20] I. F. Akyildiz, P. Wang, et S.-C. Lin, « SoftAir: A software defined networking architecture for 5G wireless systems », *Comput. Netw.*, vol. 85, p. 1-18, juill. 2015.
[21] J. S. B. Martins, « Towards Smart City Innovation Under the Perspective of Software-Defined Networking, Artificial Intelligence and Big Data », *ArXiv181011665 Cs*, oct. 2018.
[22] N. Gude *et al.*, « NOX: towards an operating system for networks », *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, nº 3, p. 105, juill. 2008.
[23] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, et R. Sherwood, « On Controller Performance in Software-Defined Networks », p. 6.
[24] A. Tootoonchian et Y. Ganjali, « HyperFlow: A Distributed Control Plane for OpenFlow », p. 6, 2010.
[25] T. Koponen *et al.*, « Onix: A Distributed Control Platform for Large-scale Production Networks », p. 14, 2010.
[26] « OpenDaylight | A Linux Foundation Collaborative Project ». [En ligne]. Disponible sur: http://archive15.opendaylight.org/. [Consulté le: 31-mars-2019].
[27] T. Tsou, P. Aranda, H. Xie, R. Sidi, H. Yin, et D. Lopez, « SDNi: A Message Exchange Protocol for Software Defined Networks (SDNS) across Multiple Domains ». [En ligne]. Disponible sur: https://tools.ietf.org/html/draft-yin-sdn-sdni-00. [Consulté le: 31-mars-2019].
[28] « Benamrane et al. - 2017 - Etude des Performances des Architectures du Plan d.pdf ». .
[29] M. Yu, J. Rexford, M. J. Freedman, et J. Wang, « Scalable Flow-Based Networking with DIFANE », p. 12.
[30] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, et S. Banerjee, « DevoFlow: Scaling Flow Management for High-Performance Networks », p. 12.
[31] S. Shin et G. Gu, « Attacking Software-Defined Networks: A First Feasibility Study », p. 2.
[32] D. M. Ferrazani Mattos et O. C. M. B. Duarte, « AuthFlow: authentication and access control mechanism for software defined networking », *Ann. Telecommun.*, vol. 71, nº 11-12, p. 607-615, déc. 2016.
[33] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, et G. Gu, « A security enforcement kernel for OpenFlow networks », in *Proceedings of the first workshop on Hot topics in software defined networks - HotSDN '12*, Helsinki, Finland, 2012, p. 121.
[34] S. Shin *et al.*, « Rosemary: A Robust, Secure, and High-performance Network Operating System », in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14*, Scottsdale, Arizona, USA, 2014, p. 78-89.
[35] S. Shin, V. Yegneswaran, P. Porras, et G. Gu, « AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks », in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13*, Berlin, Germany, 2013, p. 413-424.
[36] G. Yao, J. Bi, et P. Xiao, « Source address validation solution with OpenFlow/NOX architecture », in *2011 19th IEEE International Conference on Network Protocols*, Vancouver, AB, Canada, 2011, p. 7-12.
[37] M. Obadia, M. Bouet, J. Leguay, K. Phemius, et L. Iannone, « Failover mechanisms for distributed SDN controllers », in *2014 International Conference and Workshop on the Network of the Future (NOF)*, Paris, France, 2014, p. 1-6.
[38] Yi-Chen Chan, Kuochen Wang, et Yi-Huai Hsu, « Fast Controller Failover for Multi-domain Software-Defined Networks », in *2015 European Conference on Networks and Communications (EuCNC)*, Paris, France, 2015, p. 370-374.
[39] S. Moazzeni, M. R. Khayyambashi, N. Movahhedinia, et F. Callegati, « On reliability improvement of Software-Defined Networks », *Comput. Netw.*, vol. 133, p. 195-211, mars 2018.