Université Saad Dahleb de Blida 1

Faculté des Sciences

Département d'Informatique

Mémoire présenté par :

YOUCEF OUALI Sarah

AISSIOU Yasmine

En vue d'obtenir le diplôme de Master

Domaine : mathématique et informatique (MI)

Filière : Informatique

Spécialité : Informatique

Option : Génie des logiciels

Sujet : ***Conception et implémentation d'une approche d'équilibrage de charge flexible pour les environnements SDN-NFV***

Sujet Proposé par : Mr : *K.ZERAOULIA*

Soutenu le 25/06/2016 devant le jury composé de :

Mr. **A.CHEMCHEM** .....Président

Mme.**F.Z.ZAHRA** .....Membre

Promotion :
2015/2016

Résumé

Avec l'accroissement constant des infrastructures réseaux actuelles, un nouveau modèle de fonctionnement et de déploiement devient nécessaire. SDN (Software Defined Networking) et la NFV (virtualisation des fonctions réseaux) sont des technologies conçues pour répondre à ces nouvelles problématiques : elles permettent de rationaliser l'utilisation des ressources disponibles, tout en offrant une flexibilité que des équipements physiques dédiés ne peuvent fournir.

SDN basée sur le protocole de communication OpenFlow peut accélérer le déploiement de NFV en offrant une nouvelle architecture réseau évolutive, élastique, facilite la gestion de réseau adaptée aux exigences dynamiques des communications NFV pour les infrastructures de réseaux virtuelles et physiques.

Tout au long de ce travail, on présentera les deux technologies (SDN et NFV) et les avantages de les combiner, ainsi que l'intégration du Load Balancing (équilibrage de charge) dans un environnement SDN/NFV, dans le but de proposer et de développer de nouvelles approches d'équilibrage de charge dans cet environnement pour augmenter l'efficacité de traitement du réseau.

Mot clés : cloud computing, SDN, NFV, équilibrage de charge, openFlow.

Abstract

With constant increase of infrastructure current network, a new model of operation and deployment become necessary SDN (Software Defined Networking) and the NFV(Virtualization of Networking Functions) are designed technology to answer these new problematic : they allow the rationalize of the use of available resources, all by offering a flexibility that physical equipment dedicated cannot provide.

SDN based on protocol of communication OpenFlow can accelerate the deployment of NFV by offering new architecture scalable network, elastic, facilitates management of adopted network according to the requirement dynamic communication of NFV for the infrastructures of virtual and physical network.

Throughout this work, we will present the two technologies (SDN, NFV) and advantages of load balancing in SDN/NFV environment in order to propose and develop new approaches of load balancing in this environment to in increase the effectiveness of treatment networks.

key words : cloud computing, SDN, NFV, load balancing, openFlow.

ملخص

مع الزيادة المستمرة في البنية التحتية للشبكة الحالية، يصبح من الضروري استعمال نموذج جديد. الشبكات (SDN) و خصائص الشبكة الافتراضية (NFV) من بين التقنيات التي صممت لحل هذه الإشكاليات، فهي تسمح بتبسيط استخدام الموارد المتاحة، مع توفير المرونة التي لا تستطيع تقديمها المعدات المادية.

SDN القائمة على بروتوكول الاتصالات (OPENFLOW) تسمح بتسريع نشر تقنية ال NFV عن طريق شبكة هندسية جديدة قابلة للتطوير, مرنة، و تسهل ادارة الشبكة المكيفة للمتطلبات الديناميكية للاتصالات NFV للبنية التحتية الافتراضية و المادية.

طيلة هذا العمل، سنقدم التكنولوجيات (SDN و NFV) و فوائد الجمع بينهما و ادماج تقنية موازنة الحمولة (LoadBalancing) في بيئة NFV/SDN، من اجل اقتراح و تطوير مناهج جديدة لهذه التقنية لزيادة كفاءة معالجة الشبكة.

المفاتيح: الحوسبة السحابية، SDN، NFV، موازنة الحمولة, OPENFLOW، POX

Remerciements

Nos remerciements les plus forts vont d'abord à notre dieu qui nous a donné la force, la patience et le courage pour accomplir ce modeste travail.

Nous tenons à exprimer nos remerciements les plus sincères à notre promoteur Monsieur ZERAOULIA Khaled, qui nous a proposé ce sujet de Master intéressant, nous a soutenu tout au long de notre travail. Nous le remercions sincèrement pour l'encadrement apporté ainsi que pour son dévouement, pour les corrections et les conseils qu'il nous a donné.

Nos remerciements et notre profonde gratitude à nos familles, et surtout à nos parents pour nous avoir aidés et soutenus pendant tout notre cursus, et pendant le travail sur ce mémoire particulièrement. Nous remercions aussi les personnes qui ont partagés aves nous les années d'étude et qu'on leur souhaite bon courage. Nos remerciements vont enfin à ceux qui ont contribués de près ou de loin à la réalisation de ce travail.

Dédicace

A la lumière de ma vie, mes parents chéris, les deux prunelles de mes yeux, en témoignage de ma reconnaissance envers le soutien, les sacrifies et tous les efforts qu'ils ont fait pour mon éducation

A mon frère et sa femme

A mon adorable sœur et son mari

A mon neveu et ma nièce

A ma famille

A mes copines et mes camarades

A tous mes enseignants que je respecte tant

Je dédie ce modeste travail, en signe de reconnaissance

et de profonde affection

YOUCEF OUALI
SARAH

Dédicace

A la lumière de ma vie, mes parents chéris, les deux prunelles de mes yeux, en témoignage de ma reconnaissance envers le soutien, les sacrifies et tous les efforts qu'ils ont fait pour mon éducation

A mon père pour l'effort qu'il a consenti à ma formation

A ma défunte mère qui a voulu assister à ce grand jour

A mon adorable petite sœur

A ma famille

A mes copines et mes camarades

A tous mes enseignants que je respecte tant

Je dédie ce modeste travail, en signe de reconnaissance

et de profonde affection

AISSIOU YASMINE

Table des matières

Liste des tableaux

Table des figures

Introduction Générale

Les technologies de l'information et de la communication évoluent et révolutionnent nos modes de vie et de travail. Le cloud computing ou informatique virtuelle, est apparu ces dernières années comme un nouveau modèle de gestion et d'utilisation des systèmes informatiques. Le concept consiste à déporter sur des serveurs distants les traitements et les stockages habituellement effectués en local afin d'y accéder sous forme de service.

Le Cloud Computing (qu'on appelle aussi le nuage informatique) fournit donc des services ou des applications informatiques en ligne, accessibles partout, à tout moment, et de n'importe quel terminal (smartphone, PC de bureau, ordinateur portable et tablette). Pour être plus précis, le Cloud Computing permet de partager, chez un fournisseur d'offres Cloud, une infrastructure, une solution applicative ou encore une plateforme à tout utilisateur qui en fait la demande via un simple site internet (aussi appelé portail) en libre-service.

Le Cloud Computing est aujourd'hui le sujet phare dans le domaine des systèmes d'information et de communication. Après la virtualisation, le Cloud parait être la révélation qui va permettre aux entreprises d'être plus performantes et de gérer le coût des systèmes d'information plus sereinement. Mais, en dépit de l'usage croissant de la technologie Cloud, de nombreux problèmes cruciaux doivent encore être résolus. La surcharge du réseau et l'un de ces problèmes.

L'objectif de ce travail et d'approfondir et d'expérimenter nos connaissances sur le thème du cloud computing et faire son état de l'art, puis choisir la meilleur solution disponible, l'analyser et de l'évaluer. Pour cela, nous avons étudié l'approche «Software Defined Networking» qui représente l'architecture prédominante dans le Cloud Computing. Par la suite, nous avons présenté l'approche «Network Functions Virtualization» qui peut être combinée avec SDN pour virtualiser une partie ou la totalité du réseau.

Enfin, nous avons effectué une étude sur les techniques d'équilibrage de charge «Load Balancing» existantes dans l'approche SDN qui nous a permis de proposer de nouvelles approches dans le but de gérer la distribution du trafic et améliorer le temps de réponse.

Le présent manuscrit s'articule autour de cinq chapitres :

- Le premier chapitre nous donnons quelques définitions et généralités sur le Cloud Computing.

- Le deuxième chapitre est consacré à la description des deux approche SDN et NFV.

- Le troisième chapitre s'intéresse aux techniques d'équilibrage de charge.

- Le quatrième détaille le principe de l'utilisation d'OpenFlow qui est le protocole qui permet de mettre en œuvre l'approche SDN, ainsi que les techniques existantes et les algorithmes d'équilibrage de la charge utilisés actuellement dans les réseaux traditionnels et modernes.

- Enfin dans le dernier chapitre, nous présenterons l'environnement de travail où nous avons établi une simulation des techniques d'équilibrage de charge que nous avons proposé afin d'évaluer leurs performances dans une architecture SDN intégrant NFV.

1

Le Cloud Computing

1.1 introduction :

Dans l'environnement extrêmement concurrentiel qui les entoure, les directions informatiques doivent répondre rapidement aux besoins de leurs utilisateurs en ressources pour leurs applications professionnelles. De ce fait, on constate l'adoption rapide du modèle cloud computing, qui permet le déploiement à la demande des ressources en libre-service qui repose sur le partage d'un ensemble de ressources de calcule, de communication, et de stockage.

Cette technologie a beaucoup de bénéfice comme, réduction des couts, meilleur passage à l'échelle, partage des ressources, accessibilité des donnés, flexibilité qui s'appuie sur la virtualisation : les ressources et les services sont séparés de l'infrastructure sous forme de machines virtuelles (VM).

Dans ce chapitre nous allons présenter les notions fondamentales du Cloud Computing, son utilité, les modèles de service, les modèles de déploiement et les avantages du cloud computing. On citera aussi le principe de la virtualisation et son intérêt sur le cloud.

1.2 définition pratique du cloud computing :

Le cloud computing se traduit littéralement par "informatique dans les nuages", faisant référence aux technologies d'internet qui est souvent représenté schématiquement par un nuage. C'est un concept abstrait qui regroupe plusieurs technologies servant à délivrer des services. Son but est de pousser les entreprises à externaliser les ressources numériques qu'elles stockent. Ces ressources offrant des capacités de stockage et de calcul, des logiciels de gestion de messagerie, et d'autres services sont mises à disposition par des sociétés tierces et accessibles, grâce à un système d'identification, via un PC et une connexion à Internet [1].

Le National Institue of Standards and Technology (NIST) a donné une définition qui est souvent citée comme référence : «Le cloud computing est un modèle Informatique qui permet un accès facile et à la demande par le réseau à un ensemble partagé de ressources informatiques configurables (serveurs, stockage, applications et services) qui peuvent être rapidement provisionnées et libérées par un minimum d'efforts de gestion ou d'interaction avec le fournisseur du service » [2].

Pour le groupe de travail CIGREF [1] le Cloud Computing est défini par les quatre

1. CIGREF :(Réseau de Grandes Entreprises) :c'est une association créée en 1970, il regroupe près de 140 grandes entreprises et organismes français dans tous les secteurs d'activité. elle a pour mission de «développer la capacité des grandes entreprises à intégrer et maîtriser le numérique».

points suivant :

- Un Cloud est toujours un espace virtuel.
- Contenants des informations qui sont fragmentées.
- Dont les fragments sont toujours dupliqués et répartis dans cet espace virtuel, lequel peut être sur un ou plusieurs supports physiques.
- Qui possède « une console (programme) de restitution » permettant de reconstituer l'information [3].

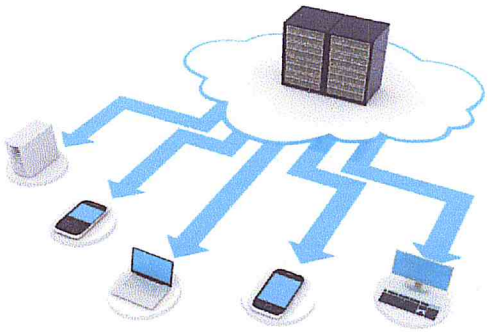


FIGURE 1.1 – cloud computing.

synonyme :

"informatique virtuelle", "informatique dans le nuage", "informatique en nuage", "informatique dématérialisée".

1.3 Historique :

Techniquement, le concept de Cloud Computing est loin d'être nouveau, il est même présent depuis des décennies. On en trouve les premières traces dans les années 1960, quand John McCarty [2] affirmait que cette puissance de traitement informatique serait accessible au public dans le futur. Le terme en lui-même est apparu plus couramment aux alentours de la fin du XXe siècle et il semblerait qu'Amazon.com [3] soit l'un des premiers à avoir assemblé des data-center [4] et fournit des accès à des clients. Les entreprises comme

2. John McCarthy : est le principal pionnier de l'intelligence artificielle. Il est également l'inventeur en 1958 du langage Lisp. A la fin des années 1950, il a créé avec Fernando Cobarto la technique du temps partagé, qui permet à plusieurs utilisateurs d'employer simultanément un même ordinateur.
3. AMAZON.com : une entreprise de commerce électronique américaine. Sa spécialité la plus connue est la vente de livres, mais elle est diversifiée dans d'autres produits, notamment dans la vente de tous types de produits culturels : disques CD, DVD, appareils photos numériques, et informatique, etc.
4. Data-Center : centre de traitement de données.

IBM [5] et Google ainsi que plusieurs universités ont seulement commencé à s'y intéresser sérieusement aux alentours de 2008, quand le Cloud Computing est devenu un concept à la mode [3].

1.4 Caractéristiques du cloud computing :

Les cinq caractéristiques suivantes, telles que définies par le NIST, sont considérés comme inhérents à des services de cloud computing :

- **Accès «On-Demand » « Self-Service » :** A la demande signifie que Le logiciel dans le Cloud est accessible lorsque l'utilisateur le souhaite. Self-service signifie que il n'y a pas de recours à un administrateur informatique interne, ni d'action du côté du fournisseur du service [10].
- **Accès au réseau étendu :** La nécessité d'un accés au réseau internet est évidente. Si l'application est utilisée depuis un lieu où la couverture réseau n'est pas suffisante cela peut poser probléme, dans ce cas de figure une coupure internet peut avoir autant d'impact que le crash d'un ordinateur. [10].
- **Les ressources sont communs :** Des ressources telles que la bande passante réseau, machines virtuelles, mémoire, puissance de traitement, capacité de stockage, etc sont disponible et peuvent être utilisées par plusieurs clients. Autrement dit, les ressources virtuelles et physiques sont affectées dynamiquement et réaffectés, c'est-à-dire une fois qu'un client à terminer avec une ressource il la libère pour que d'autres clients utilisent cette même ressource [4].
- **Élasticité rapide :** L'utilisateur accède aux ressources informatiques autant que nécessaire. Les ressources informatiques sont mobilisables rapidement, par exemple les utilisateurs peuvent ajouter de l'espace de stockage en moins de temps qu'il n'en faudrait pour commander et installer un ordinateur. De même, ces ressources sont démobilisables, l'utilisateur peut décider de diminuer l'espace de stockage ou le nombre d'utilisateurs à sa guise [10].
- **Service mesurée :** L'utilisation du service est mesurée par des paramètres liés directement à cette dernière : heures d'utilisation, nombre d'utilisateurs, espace disque, fonctions utilisées, etc. Le prix du service est lié à la mesure de ces grandeurs, et donc directement lié a l'utilisation (c'est l'utilisateur qui réalise les opérations) [10].

5. IBM : International Business Machines Corporation société multinationale américaine présente dans les domaines du matériel informatique, du logiciel et des services informatiques.

1.5 Les modèles de service :

Les 3 Niveaux du cloud impliqués dans l'établissement d'un service entre un client et un offreur de service sont illustrés dans la figure 1.2 :

FIGURE 1.2 – les modèles de service du cloud computing.

Ces trois modèles de services doivent être déployés sur des infrastructures qui possèdent les cinq caractéristiques essentielles citées plus haut pour être considérées comme du Cloud Computing.

1.5.1 Le SAAS (Software As A Service) :

Concept consistant à proposer un abonnement à un logiciel plutôt que l'achat d'une licence. On oublie donc le modèle client-serveur et aucune application n'est installée sur l'ordinateur, elles sont directement utilisables via le navigateur Web. L'utilisation reste transparente pour les utilisateurs, qui ne se soucient ni de la plateforme, ni du matériel, qui sont mutualisés avec d'autres entreprises. Le Saas remplace l'ASP (application service provider), qui est une entreprise qui fournit des logiciels ou des services informatiques à ses clients au travers d'un réseau [1].

• Avantage :
plus d'installation, plus de mise à jour, une migration rapide de données, le paiement à l'usage, et test de nouveaux logiciels avec facilité.
• Inconvénient :
limitation par définition au logiciel proposé. Pas de contrôle sur le stockage et la sécurisation des données associées au logiciel [1].

Exemple : les logiciels de messagerie au travers d'un navigateur comme Gmail ou Yahoo mail.

1.5.2 Le PAAS (Platform As A Service) :

Il s'agit des plateformes du nuage, regroupant principalement les serveurs mutualisés et leurs systèmes d'exploitation. En plus de pouvoir délivrer des logiciels en mode SaaS, le PaaS dispose d'environnements spécialisés au développement comprenant les langages, les outils et les modules nécessaires.
L'avantage est que ces environnements sont hébergés par un prestataire basé à l'extérieur de l'entreprise ce qui permet de ne disposer d'aucune infrastructure et de personnel de maintenance et donc de pouvoir se consacrer au développement [1].

• Avantage :
Le déploiement est automatisé, pas de logiciel supplémentaire à acheter ou à installer.
• Inconvénient :
Limitation à une ou deux technologies (ex : Python ou Java pour Google AppEngine, .NET pour Microsoft Azure, propriétaire pour force.com). Pas de contrôle des machines virtuelles sous-jacentes. Convient uniquement aux applications Web [1].

Exemple : Google App Engine (GAE) est le principal acteur proposant ce genre d'infrastructures ou l'utilisateur de ces services n'a pas à gérer des serveurs ou des systèmes pour déployer ses applications en ligne et dimensionner des ressources adaptées au trafic.

1.5.3 Le IAAS (infrastructure as a service) :

Il s'agit de la mise à disposition, à la demande, de ressources d'infrastructures dont la plus grande partie est localisée à distance dans des Datacenter.
L'IaaS permet l'accès aux serveurs et à leurs configurations pour les administrateurs de l'entreprise. Le client a la possibilité de louer des clusters [6], de la mémoire ou du stockage de données. Le coût est directement lié au taux d'occupation. Une analogie peut être faîte avec le mode d'utilisation des industries des commodités (électricité, eau, gaz) ou des télécommunications [1].

• Avantage :
Grande flexibilité, contrôle total des systèmes, qui permet d'installer tout type de logiciel métier.
• Inconvénient : Besoin d'administrateurs système comme pour les solutions de serveurs classiques sur site.

6. clusters :est une grappe de serveurs sur un réseau, appelé ferme ou grille de calcul(des techniques consistant à regrouper plusieurs ordinateurs indépendants appelés nœuds).

Exemple :Amazon EC2 [7] est le principal qui propose ce genre d'infrastructures. Eucalyptus un exemple d'infrastructure.

La figure 1.3 montre un compromis flexibilité/simplicité des trois couches du cloud computing, la flexibilité est obtenue grâce à la virtualisation des systèmes d'exploitation. La plateforme est exécutée via des machines virtuelles et les ressources peuvent être allouées et délibérées à la demande. Ainsi, l'IaaS est considéré comme le service le plus flexible.

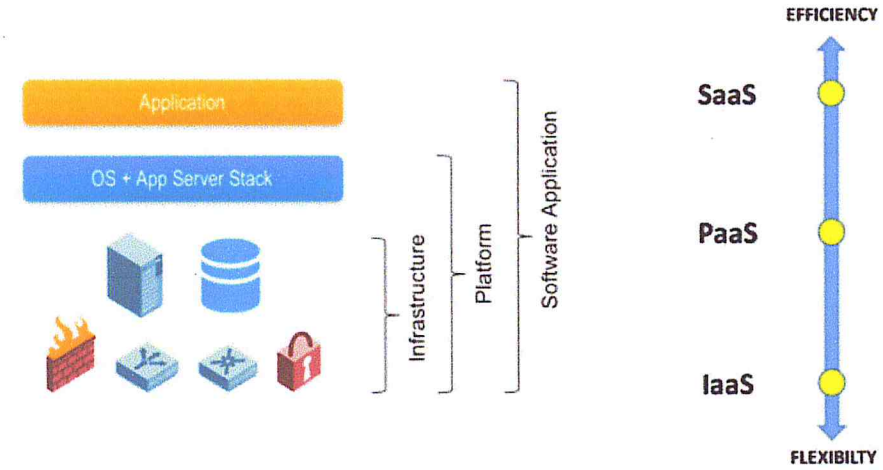


FIGURE 1.3 – les niveaux du cloud computing.

Des exemples de prestataires de Cloud public dans le monde :

IaaS	PaaS	SaaS
• Amazon – offres EC2 et AWS • Microsoft – offre Azur	• Microsoft – offre Azur • Google – offre Google App Engine	• Google – offre Google Apps (messagerie et bureautique) • SalesForce – CRM (Customer Relationship Management) • Microsoft – offre Office 365 (outils collaboratifs)

7. EC2 : ou Amazon Elastic Compute Cloud est un service proposé par Amazon permettant à des tiers de louer des serveurs sur lesquels exécuter leurs propres applications web.

1.6 les modeles de déploiements :

Nous distinguons quatre formes de Cloud Computing : Le Cloud publique, également le premier apparu, le Cloud privé, le Cloud communautaire, et le Cloud hybride.

1.6.1 Le cloud privé :

Dans cette infrastructure les ressources sont mis à disposition d'une seule organisation qui les gèrent elle-même (Cloud Privé interne) ou par un tiers (Cloud Privé externe). Un exemple de ceci est Redmine [8], qui utilise son propre VMware vCloud [9] installation pour déployer son système [5][44].

1.6.2 Le cloud public :

Cette infrastructure cloud est disponible pour un groupe de la grande industrie ou le grand public et les utilisateurs ont accès à des services cloud via l'Internet public sans savoir précisément où sont hébergées leurs données ni où sont exécutés leurs traitements. Un exemple qui montre ceci : Amazon, Google et Microsoft dans lesquels n'importe quel particulier ou entreprise peut y héberger ses applications, services et données [5][44].

1.6.3 Le cloud communautaire :

L'infrastructure Cloud est partagée par plusieurs organisations pour les besoins d'une communauté qui souhaite mettre en commun des moyens (par exemple les exigences de sécurité et de conformité). Elle peut être gérée par des organisations ou par une tierce partie et peut être placée dans des locaux ou à l'extérieur [5][44].

1.6.4 Le cloud hybride :

Cette infrastructure cloud représente la conjonction de deux ou plusieurs Cloud (public, privé et communautaire) mais qui restent des entités uniques connectés via la technologie, qui permet la portabilité des données ou des applications [5][44].

8. redmine : est une application web libre de gestion de projets presque complète en mode web développée en Ruby(est un langage de programmation libre. Il est interprété, orienté objet et multi-paradigme.) Sur la base du framework Ruby on Rails.

9. VCloud : est un projet de cloud computing mené par VMware. Il a pour but de permettre à ses clients de migrer leur travail, à leur demande, à partir de leur stockage interne des hyperviseurs VMware vers un stockage à distance.

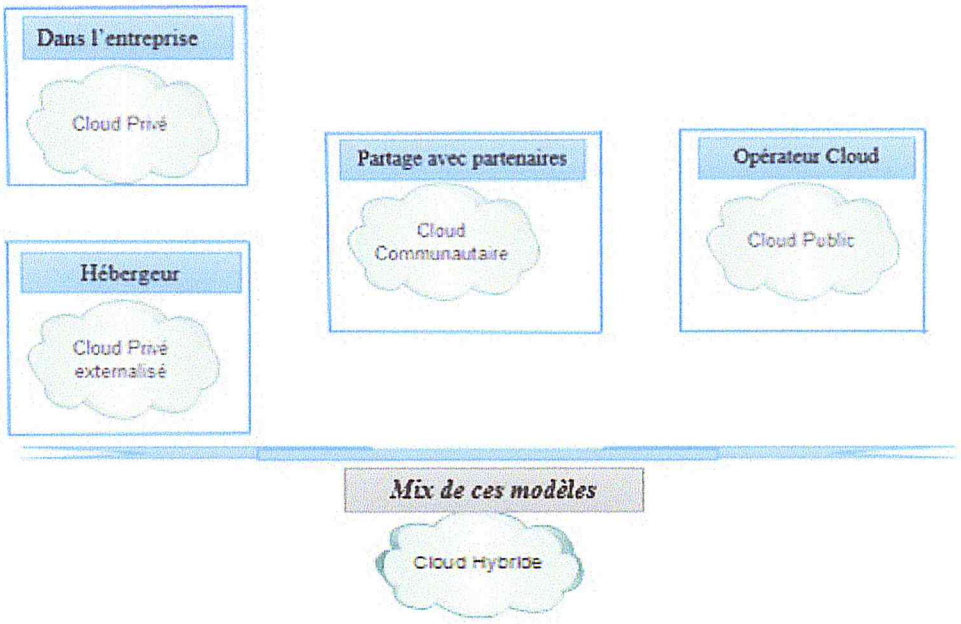


FIGURE 1.4 – les modeles de deploiment du cloud computing.

1.7 Les caractéristiques communes des différents modèles de Cloud :

Le Cloud Computing tire parti d'un certain nombre de caractéristiques pour fournir des services dans des conditions techniques et économiques très avantageuses. La plupart des entreprises et des particuliers ont intérêt à utiliser des fournisseurs fiables pour garantir la fiabilité et les meilleures conditions économiques. Parmi ces caractéristiques communes, on trouve généralement [6] :

- **Des infrastructures gigantesques :** augmentation de la taille du système de stockage.
- **Une grande homogénéité des moyens :** les systèmes regroupent des milliers de composants identiques ce qui simplifie la gestion, la fiabilité, l'audit et la sécurité.
- **Virtualisation :** la virtualisation est une caractéristique indispensable qui présente de très nombreux avantages. Le matériel est remplacé par du logiciel avec tous les avantages du logiciel. La machine virtuelle ne tombe pratiquement jamais en panne ce qui accroit sérieusement la fiabilité des systèmes.
- **Elasticité :** l'ensemble des caractéristiques précédentes (taille, homogénéité et virtualisation) permet d'adapter automatiquement la capacité de traitement d'une application à la demande constatée. La mise en ligne d'un nouveau serveur peut s'effectuer en moins d'une minute. Il n'est plus nécessaire de s'équiper pour absorber des pointes de trafic.
- **Coûts de logiciels très réduits :** la plupart des plates-formes publiques utilisent des

logiciels open source gratuits. Les couts des logiciels propriétaires sont souvent facturés à l'usage sans nécessiter l'achat de licences. La plupart de ces logiciels sont déjà préinstallés et préconfigurés ce qui fait gagner beaucoup de temps.

- **Distribution géographique :** les grandes plates-formes publiques disposent de centres répartis sur la planète pour réduire les risques et placer les données au plus près des utilisateurs.

- **Orientation Service :** les fonctions fournies aux utilisateurs se présentent sous la forme de Web services (REST [10]) faciles à utiliser dans un navigateur ou par des scripts automatisés. Des groupes de standardisation ont été créés pour définir des interfaces communes et simplifier ainsi le passage d'une infrastructure à une autre.

- **Fonctions de sécurité avancées :** la sécurité est une préoccupation majeure des organisations qui utilisent les services du Cloud. Ces plates-formes disposent généralement de nombreux systèmes de protection, hors de portée des moyens de la plupart des organisations.

1.8 La virtualisation :

1.8.1 introduction

La flexibilité de traitement et de partage des ressources informatiques au niveau du cloud s'appuie sur la virtualisation : les ressources et les services sont séparés de l'infrastructure sous forme de machines virtuelles (VM).

1.8.2 Principe de la virtualisation :

La virtualisation est une technique informatique consistant à faire fonctionner plusieurs environnements logiques indépendants séparément sur une même machine. Il s'agit d'une extension du principe d'émulation, l'émulation consiste à substituer un ou plusieurs éléments informatiques par une application. Appliquée à la virtualisation, un système prétend être plusieurs systèmes différents, alors c'est une technique qui consiste à dissocier les ressources matérielles (serveurs, ordinateurs) des ressources logicielles (systèmes d'exploitation, applications), afin de délivrer une meilleure utilisation et flexibilité des ressources de traitement [7].

10. REST :(representational state transfer) est un style d'architecture pour les systèmes hypermédia (Un hypermédia est une extension de l'hypertexte à des données multimédias, qui ajoute aux informations de type texte, d'autres médias comme des images, sons, vidéos ou encore des données multimédia.) distribués.

1.8.3 La définition virtualisation de réseaux (Network Virtualization) :

Le terme de NV se réfère à la création de partitions logiques du réseau isolées et superposées au-dessus d'une infrastructure physique commune (débit des liens, ressources CPU, des routeurs,...). Chaque partition est logiquement isolée des autres, et doit se comporter comme un réseau entièrement dédié pour assurer la confidentialité, la sécurité, et un ensemble indépendant de politiques, de niveaux de service, et même des décisions de routage [44].

Avantage :

- L'exécution concurrente de différents services (par exemple, systèmes d'exploitation) sur une même plate-forme matérielle (par exemple, machine physique) ;
- L'exécution d'un même service sur des plates-formes matérielles multiples ;
- La plate-forme logique ainsi constituée fournit les mêmes entrées, sorties, et comportements que l'on pourrait attendre d'une plate-forme physique.
- Une meilleure flexibilité puisqu'une modification de la structure des réseaux peut être réalisée en modifiant la configuration du commutateur [44].

1.8.4 Les avantages de la virtualisation :

- Assurez une disponibilité des applications.
- Réduire les interruptions de service.
- Accroître la flexibilité et l'évolutivité de l'infrastructure informatique.
- Accélérer le déploiement des charges de travail.
- Moins coûteuse à acquérir et à exploiter.
- Optimisé la continuité et la reprise d'activité [8].

1.9 Avantages et inconvénients du cloud computing :

1.9.1 avantages :

- **mises à jour et évolutivité** : il suffit de mettre à jour l'application réseau et tous les utilisateurs bénéficient des nouveautés et des corrections.
- **mise en commun des ressources** : les ressources sont partager et permettent ainsi un travail à plusieurs (Partage et travail collaboratif).
- **mobilité et accessibilité** : Les données sont sur un serveur, l'utilisateur peut à tout moment et à partir de n'importe quel appareil se connecter à ses applications. Il peut y accéder à partir de n'importe quel type d'appareil à condition que celui-ci soit doté d'un

navigateur.
- **Fiabilité** : Les services basés sur des infrastructures performantes possédant des politiques efficaces de tolérance aux pannes [9].

1.9.2 Inconvénients :

- **sécurité** : la plateforme cloud doit être suffisamment sécurisée pour éviter le risque d'intrusion, de vol des données par piratage. L'autre risque est qu'un utilisateur oublie de se déconnecter sur un appareil accessible par des éléments externes à l'organisation.

- **Connexion à internet obligatoire** : Si l'utilisateur n'a pas de connexion internet, ou une connexion insuffisante, il ne pourra pas accéder à sa plateforme de travail. L'idée dans ce cas est de permettre le travail sur une application locale qui synchronise ensuite les données avec le serveur dès que l'utilisateur a à nouveau accès au réseau. Le problème de la sécurité des données en local se pose donc à nouveau [9].

1.10 Cloud Computing et sécurité :

La sécurité permet de garantir la confidentialité, l'intégrité, l'authenticité et la disponibilité des informations.

lorsqu'il est question de Cloud Computing, des préoccupations encore plus accentuées lorsqu'il s'agit de Cloud public. Certaines questions légitimes reviennent sans cesse :

- Mes données sont-elles sures dans le Cloud ?
- Où sont stockées mes données ?
- Qui va avoir accès à mes données ?
- Aurais-je accès à mes données à n'importe quel moment ?
- Que deviendrons mes données s'il y a interruption du service ?

La mise sur pied d'une solution de Cloud Computing comporte des problèmes de sécurité inhérents à la solution elle-même. Le fait de centraliser toutes les informations sur un site pose un grand nombre de problèmes. On peut citer comme problème potentiel :
- Une possible interruption massive du service.
- Une cible de choix pour les hackers.
- Interface et API [11] non sécurisé.

11. API : Application Programmable Interface, est un ensemble de fonctions permettant d'accéder aux services d'une application, par l'intermédiaire d'un langage de programmation.

Ce point de vulnérabilité du Cloud Computing fait depuis quelques années l'objet de recherches avancées. Il a été créé un organisme chargé de mettre sur pied des normes en matière de sécurité dans le Cloud Computing. Cet organisme s'appelle CSA (Cloud Security Alliance). Du travail de cet organisme, il en est ressorti certaines techniques utilisées de nos jours pour améliorer la sécurité du Cloud Computing. Parmi ces techniques on peut citer :
- La multi-location : cette technique permet de créer des instances d'une même donnée sur plusieurs sites différents. Elle permet une récupération facile en cas de désastre.
- Le chiffrement : le chiffrement de l'accès à l'interface de contrôle, le chiffrement des données dans le Cloud.
- L'isolation des machines virtuelles.

1.11 La révolution du cloud computing :

À l'heure où les entreprises migrent vers le cloud et vers de nouveaux environnements informatiques à la demande, tout doit permettre aux applications de délivrer la qualité de service attendue par leurs utilisateurs. Critère essentiel : le temps de réponse. Et pour ceci, il faut que la capacité du réseau puisse suivre la sollicitation faite aux applications. En deux mots, être capable de modifier la bande passante en fonction des besoins ! Mais également fixer la priorité de tel ou tel flux.

Avec les réseaux traditionnels, il est souvent très difficile de re-paramétrer à la volée des cascades de switches, sous prétexte qu'une application est en train de subir un pic de charge, et qu'elle doit donc disposer de 3 fois plus de bande passante pour garantir un temps de réponse acceptable. La virtualisation, le cloud sont typiquement des activités qui transforment radicalement les typologies et les caractéristiques des flux réseaux dans le Datacenter.

SDN est la première réponse pour apporter plus d'intelligence réseau aux applications. Les technologies SDN viennent bousculer cet état de fait et offrent au réseau ce qui lui manquait pour achever la révolution du Cloud : l'élasticité et la gestion centralisée [12].

1.12 Solutions du Cloud existante :

Le Cloud Computing représente un nouveau défi dans le monde informatique. Plusieurs solutions sont proposées : des solutions propriétaires et des solutions open sources. Nous allons présenter dans cette partie quelque solutions Cloud existante [3].

1.12.1 Solutions propriétaires :

1. **VMwareCloud** :

   Les solutions de Cloud Computing VMware favorisent l'innovation et rendent l'environnement informatique plus efficace, plus flexible et plus fiable. VMware fournit à la direction informatique tout ce qui lui est nécessaire pour concevoir, faire fonctionner et gérer le Cloud.
   Les solutions de Cloud Computing VMware optimisent les capacités du Cloud, leurs Efficacité, flexibilité et fiabilité garantie [3].

2. **Office 365** :

   C'est la version Cloud Computing de Microsoft avec des niveaux d'utilisation au choix : messagerie, office, partage et accès aux de données. Avec Office 365, Microsoft optimise le Virtual Office, et offre une solution Cloud qui permet via un simple abonnement d'accéder à l'ensemble des données depuis n'importe quelle plateforme (PC, Smartphone, Tablette) [11].
   Le but recherché derrière cette démarche est d'externaliser la messagerie électronique, de permettre aux utilisateurs d'accéder à des documents partagés sur l'espace SharePoint [12] online et de pouvoir communiquer à l'aide de la messagerie instantanée de la vidéo conférence et cela de façon intégrée et cohérente selon des règles d'accès précises à travers des rôles utilisateurs.

1.12.2 Solutions Open source :

1. **OpenNubela** :

   Il s'agit d'une plateforme purement open-source permettant de déployer des Clouds privés, hybrides et publiques. Elle est écrite en C++, Ruby et Shell. Sa puissance consiste dans ses connecteurs vers des fournisseurs d'IaaS sur les Clouds publiques tels que : Amazon EC2 Web Service, ElasticHosts REST, etc [3].

12. SharePoint :une série de logiciels pour applications Web et portails développée par Microsoft. Les fonctionnalités des produits SharePoint sont la gestion de contenu, les moteurs de recherche, la gestion électronique de documents, les forums, la possibilité de créer des formulaires et des statistiques décisionnelles.

2. **OpenStack :**

   OpenStack est une offre d'IaaS 100 % open-source encore en développement qui a livré son code source récemment et qui permet aux sociétés de développer leurs propres solutions d'infrastructure du Cloud Computing.
   Plus que trente fournisseurs soutiennent ce projet tels que : AMD, Intel, Dell et Citrix. Il comprend le logiciel OpenStackCompute pour la création automatique et la gestion de grands groupes de serveurs privés virtuels et le logiciel OpenStack Stockage pour optimiser la gestion de stockage, répliquer le contenu sur différents serveurs et le mettre à disposition pour une utilisation massive de données [3].

1.13 Conclusion :

Dans ce chapitre, on a fourni le principe du cloud computing ainsi que ses concepts, en présentant ses caractéristiques, ses modèles de service et ses modèles de déploiement. On ajoutant le concept de la virtualisation et la virtualisation réseau(NV) ainsi que ses avantages et les solutions existantes de cette technologie.

On peut considérer que le cloud computing est une évolution technique, C'est un modèle qui permet l'accès au réseau à la demande et les ressources sont partagées, il offre aussi un déploiement rapide, la réduction des coûts, l'élasticité rapide, et l'accès au réseau omniprésent. Le client peut bénéficier d'une flexibilité importante avec un effort minimal de gestion.

Dans le chapitre 2, nous allons présenter l'approche SDN que nous allons adopter dans la suite de notre travail.

2

SDN - NFV

2.1 introduction :

Aujourd'hui avec l'avènement du Cloud, nous assistons à une nouvelle révolution dans l'automatisation avec le SDN (Software Defined Network). Ce concept développé dès 2008 dans les universités américaines de Berkeley et de Sandford, introduit un nouveau paradigme dans l'architecture réseau, qui permet de l'automatisée et de le rendre programmable et qui peut être mise en place via divers protocoles que nous allons présenter par la suite. Ainsi, on présentera aussi le NFV (Network Functions Virtualization) qui garantit une flexibilité de déploiement et une réduction des pannes et des couts. SDN et NFV sont des technologies complémentaires qui peuvent être utilisées en combinaison pour apporter au réseau le même niveau d'agilité que celui déjà atteint par d'autres secteurs de l'informatique.

L'objectif de ce chapitre est de présenter l'approche SDN et le protocole utiliser pour mettre en place cette technologie, ainsi que la virtualisation des fonctions et la relation entre eux, sans oublié les avantages qu'apporte au cloud computing.

2.2 Pourquoi le NFV et le SDN

- La virtualisation : l'utilisation des ressources du réseau sans préoccuper de savoir où il se trouve physiquement, combien, comment elles sont organisé, etc ;
- Orchestration : permet de contrôler et gérer des milliers de périphériques ;
- programmable : être capable de changer de comportement à la volée ;
- Mise à l'échelle dynamique : la possibilité de changer la taille, la quantité ;
- Automatisation : dépannage ; Réduire les temps d'arrêt ; Approvisionnement / Réapprovisionnement / Segmentation des ressources ; Ajouter de nouvelles charges de travail/des sites/ des dispositifs et des ressources ;
- Visibilité : surveiller les ressources du moniteur, de la connectivité ;
- Performance : Optimiser l'utilisation de dispositif de réseau ;
- Multi-location : les locataires ont besoin de compléter le contrôle sur leurs adresses, topologie, sécurité, et routage ;
- Intégration de services : équilibreurs de charge, pare-feu, systèmes de détection d'intrusion (IDS), provisionnés à la demande[28].

2.3 Software-Defined Networking (SDN) :

Dans cette section, nous allons donner un aperçu sur SDN et examiner ses interfaces. Nous allons aussi montrer le potentiel des protocoles de contrôle pour SDN ainsi que ses avantages sur le réseau.

2.3.1 définition :

Le concept de SDN (Software Defined Networking) repose sur un découplage entre le plan de commutation (data plane) local aux équipements réseaux et le plan de contrôle (contrôle plane) qui devient déporté, autorisant une gestion centralisée du réseau. La conséquence majeure est que le réseau devient programmable [22].

Les principaux aspects de SDN peuvent être résumés dans ces trois points [23] :

- La séparation des plans de données et de contrôle par une interface uniforme appelée OpenFlow ;
- La centralisation logique du plan de contrôle, en utilisant un système d'exploitation réseau, qui construit et présente une carte logique (map) de l'ensemble du réseau pour les services ou les applications de contrôle ;
- Le découpage et la virtualisation du réseau sous-jacent.

2.3.2 Architecture SDN :

Software-Defined Networking (SDN) est une architecture émergente qui est dynamique, gérable, rentable et adaptable, ce qui la rend idéal pour les applications dynamiques gourmandes en bande passante. Cette architecture découple les fonctions de transmission du réseau et le contrôle permettant à ce dernier de devenir directement programmable.

L'architecture SDN est :

- **Directement programmable :** le contrôle du réseau est directement programmable car il est dissocié des fonctionnalités de redirection.
- **Agile :** La séparation du contrôle et des fonctionnalités de redirection permet aux administrateurs d'ajuster le trafic de façon dynamique à l'échelle du réseau afin de répondre à l'évolution des besoins.
- **Gérée centralement :** L'intelligence du réseau est centralisée (logiquement) au sein des contrôleurs logiciels SDN qui maintiennent une vue globale du réseau, qui est perçu par las applications et les moteurs de politique (Policy Engines) [1] comme un commutateur logique unique.
- **Configurée par programmation :** Le SDN permet aux administrateurs de réseaux de configurer, gérer, sécuriser et d'optimiser très rapidement les ressources réseau par

1. Policy Engines : est un composant logiciel qui permet à une organisation de créer, contrôler et appliquer les règles sur la façon dont les ressources réseau et les données de l'organisation peuvent être accessibles.

l'intermédiaire de programmes SDN automatisés et dynamiques qu'ils peuvent écrire eux-mêmes, ces programmes ne dépendant pas d'un quelconque logiciel propriétaire.

- Basée sur des standards ouverts et non liée à un quelconque fournisseur : Lorsqu'il est mis en œuvre via des standards ouverts, le SDN simplifie la conception et l'exploitation des réseaux, car les instructions sont fournies par des contrôleurs SDN et non plus par une multitude de protocoles et de périphériques propriétaires [13].

La figure 2.1 illustre la structure SDN qui est constituée de trois couches distinctes qui sont accessibles par le biais des OpenAPI [2].

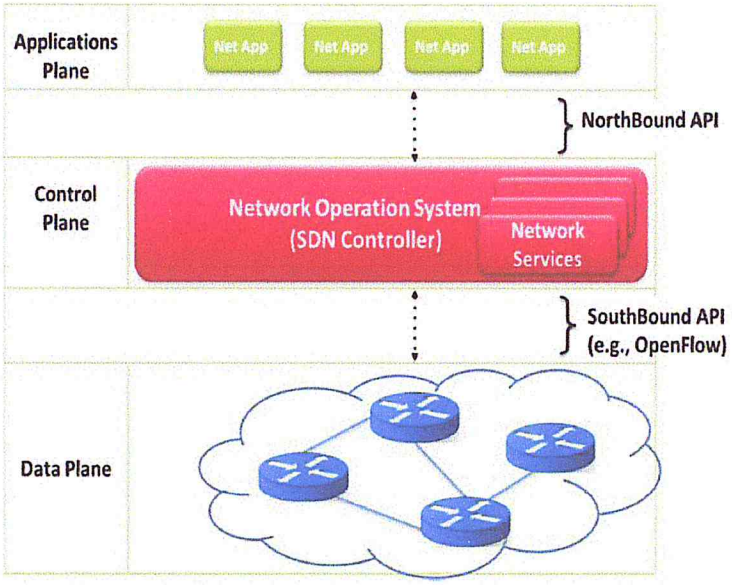


FIGURE 2.1 – architecture SDN.

- La couche d'application : se compose des applications réseau qui peuvent introduire de nouvelles caractéristiques au réseau telles que la sécurité, la gérabilité et les systèmes de transmission ou aider la couche de contrôle dans la configuration du réseau. La frontière entre la couche d'application et la couche de contrôle est traversée par une API orientée vers le nord [14].

- La couche de contrôle : (appelée aussi control plane) fournit une fonctionnalité de contrôle logiquement centralisée qui supervise le comportement de transmission du réseau grâce à une interface ouverte OpenFlow [14].

- La couche d'infrastructure/commutation : (appelée aussi data plane) se compose des éléments de réseau (Network Eléments) et des dispositifs qui fournissent la commutation de paquets et la transmission [14].

2. OpenAPI : est un terme utilisé pour le processus API qui utilise des ensembles de technologies qui permettent aux sites Web d'interagir les uns avec les autres en utilisant REST, SOAP, JavaScript et d'autres technologies Web.

Une description des principaux concepts composant l'architecture SDN illustrée dans la Figure 2.1 :

- **Applications d'entreprise** : Fait référence aux applications directement consommables par les utilisateurs. Par exemple, applications de vidéoconférence [13].
- **Services réseau** : Fait référence à la fonctionnalité qui permet aux applications d'entreprise de fonctionner de façon efficace et sécurisée, toutes les fonctionnalités de contrôle d'un commutateur traditionnel (c'est-à-dire les protocoles de routage qui sont utilisés pour concevoir les bases d'information de retransmission) s'exécutent au sein du contrôleur central. La fonctionnalité à l'intérieur du commutateur se limite exclusivement au panneau de données [13].
- **API orientée vers le Nord (Northbound API)** : permet les communications entre la couche de contrôle et la couche des applications d'entreprise. Il n'existe actuellement aucune norme en matière d'API orientée vers le Nord [13].
- **API orientée vers le Sud (Southbound API)** : permet les communications entre la couche de contrôle et la couche d'infrastructure. Parmi les protocoles permettant ces communications : OpenFlow [13].

La figure 2.2 présente des normes et protocoles qui peuvent être utilisés dans une architecture SDN et qui sont [15] :

- **OpenDaylight** : est une plate-forme ouverte dédiée à la programmation des réseaux et au déploiement du SDN et du NFV [3]. Le cœur du projet est constitué d'un contrôleur SDN, permettant ainsi la modularité et extensibilité. Le contrôleur expose :
- Des interfaces REST et OSGi [4] (avec REST l'API est utilisée pour des applications qui ne s'exécutent pas dans le même espace d'adressage que le contrôleur tandis que OSGi est utilisé pour des applications qui s'exécutent dans le même espace d'adresses).

Il dispose aussi d'une architecture modulaire autorisant différente protocoles réseau south-bound, les protocoles couramment utilisés sont : OpenFlow, OVSDB, NETCONF, BGP-LS, PCEP, LISP et SNMP. Le tableau 2.1 décrit de manière générale leurs fonctionnalités [44].

- **Le contrôleur** : il joue le rôle d'une pièce centrale du système qui permet d'héberger de multiples services, il inclut une couche d'abstraction de services qui permet de

3. NFV : Network Function Virtualization.
4. OSGi : Open Services Gateway initiative, est une organisation qui spécifie une plate-forme de services fondée sur le langage Java qui peut être gérée de manière distante.

supporter plusieurs protocoles south-bound grâce à l'utilisation de plug-ins, propose aussi des interfaces north-bound aux consommateurs tels qu'Openstack [5] [15].

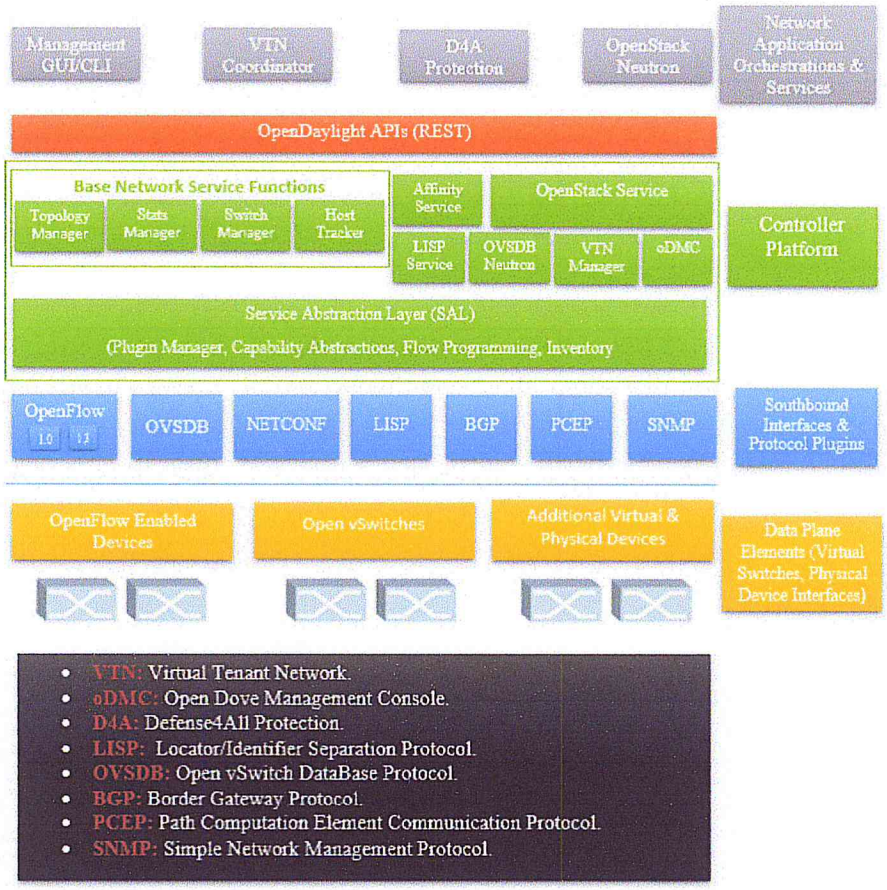


FIGURE 2.2 – Architecture détaillée du SDN.

- VTN (OpenDaylight Virtual Tenant Network) : est une application qui fournit un réseau virtuel multi-tenant [6] à un contrôleur SDN, il fournit un plan d'abstraction logique qui permet une séparation complète entre la vue logique et la vue physique du réseau. Les utilisateurs peuvent ainsi concevoir et déployer n'importe quel type de réseau, sans avoir à se soucier de la topologie sous-jacente.

- OpenDove :DOVE (distributed overlay virtual Ethernet) est une plate-forme de virtualisation réseau qui fournit des réseaux multi-tenant isolés sur un réseau IP.

- Affinity Metadata Service : il s'agit d'un langage qui permet aux applications d'exprimer leurs besoins au réseau, pour permettre au contrôleur SDN de configurer le réseau en fonction des besoins de l'application, sans qu'elle soit au courant de la topologie du

5. OpenStack : Il s'agit d'un système d'exploitation conçu pour le cloud et servant à contrôler de larges entités de calcul et de stockage ainsi que les ressources réseau au sein du centre de données.

6. multi-tenant : ou Multi-locataire, désigne une des architectures logiciel permettant à un logiciel de servir plusieurs organisations clientes à partir d'une seule installation. Elle permet de mutualiser les ressources et les couts nécessaires à l'exécution de l'application.

réseau et des détails spécifiques du réseau physique.

- **LISP Mapping Service** : il s'agit d'un standard IETF [7], basée sur la séparation entre les plan logiques et physique il peut être utilisé pour la virtualisation.

- **Yang Tools** : yang est un langage de modélisation utilisé pour exprimer des modèles pour différents protocoles south-bound. L'objectif est de développer des outils permettant le support de l'infrastructure OpenDaylight.

- **Defense4All** :C'est une application qui vise à doter l'administrateur système de moyens lui permettant de faire face à des attaques par déni de service distribué. Defense4All est composé de quatre sous-systèmes (statistiques, détection d'anomalies, redirection du trafic, système d'arbitrage).

Protocole	Description
OpenFlow	Permet de centraliser l'orchestration [8] en offrant une automatisation et une simplification du provisionning [9].
OVSDB	L'implémentation d'Open vSwitch [10] Database management protocole (avec OpenDaylight) permet la configuration de l'interface south-bound des vSwitches. C'est un protocole essentiel pour la virtualisation réseau.
NETCONF	Network Configuration Protocol fournit des mécanismes pour installer, manipuler et supprimer la configuration des périphériques réseau à l'aide de RPC [11] (appel de procédure distante).
BGP-LS/PCEP	BGP-LS amène le support de BGP Linkstate Distribution à l'interface south-bound du contrôleur. PCEP ajoute le support du protocole Patch Computation Elément afin d'implémenter des chemins dans le réseau sous-jacent.
LISP	Locator/Identifier Séparation Protocol permet de séparer deux rôles distincts joués par les adresses IP : celui de localisateur utilisé par le routage et celui d'identificateur de machine. LISP permet d'améliorer la Scalabilité, la mobilité, le simple déploiement, la virtualisation du réseau et la haute disponibilité.
SNMP	Simple Network Management Protocol permet aux administrateurs réseau de gérer les équipements du réseau, de superviser et de diagnostiquer les problèmes réseaux à distance.

TABLE 2.1 – les protocoles de communication de soutn-bound API.

7. IETF : Internet Engineering Task Force.

8. L'orchestration : décrit le processus d'organisation automatique, de coordination, et de gestion de systèmes informatiques complexes, de middleware, et de services.

9. Provisionning : désigne l'allocation automatique de ressources permettant d'installer et de configurer des logiciels à distance, ou encore d'allouer de l'espace disque, de la puissance ou de la mémoire.

10. Open vSwitch : ou OVS est une production de qualité open-source pour l'implémentation d'un switch multicouche virtuel distribué. Le but principal d'Open vSwitch est de fournir une pile de commutation pour les environnements de virtualisation de matériel, tout en soutenant de multiples protocoles et les normes utilisées dans les réseaux informatiques.

11. RPC : Remote procedure call.

Parmi ces protocoles de communication, l'OpenFlow est le protocole le plus utilisé car il permet de combiner NFV et SDN qui visent à réduire la complexité et la ségrégation du plan de contrôle et de données [15].

2.3.3 Le protocole OpenFlow :

OpenFlow est la première interface de communication standard définie entre les couches d'une architecture SDN (entre le plan de données et le plan de contrôle) .Il permet un accès direct et une manipulation du plan d'acheminement des périphériques réseau tels que les commutateurs et les routeurs.
Le développement d'OpenFlow a commencé en 2007 dans le cadre d'une collaboration entre les mondes de l'université et des affaires. Etablie à l'origine par l'université de Stanford et l'université de Californie à Berkeley, cette norme est maintenant définie par l'Open Networking Foundation (ONF) [16][17].

OpenFlow permet l'exécution des opérations du plan de contrôle dans un contrôleur SDN. Il permet également d'ajuster les règles de routage selon les besoins dynamiques du réseau [18].

La technologie OpenFlow est utilisée pour la mise en œuvre de la virtualisation de réseaux dans l'environnement de Cloud Computing, elle est utilisée pour contrôler les réseaux overlay [12].

2.3.4 Avantages de l'OpenFlow-Based Software-Defined Networks :

Actuellement en cours de déploiement dans des centres de données ou sur des réseaux, SDN basée openflow offre des avantages substantiels [19] :

- **L'indépendance** : gestion et contrôle de commutateurs réseaux indépendants du fabricant ;
- **L'automatisation :** c'est à dire développer des outils qui automatisent de nombreuses taches de gestion de réseau qui sont faite manuellement, c'est outils d'automatisation permettrons de réduire le cout opérationnel et l'instabilité du réseau ;
- **La fiabilité et la sécurité du réseau :** l'architecture SDN basée OpenFlow élimine la nécessité de configurer individuellement les périphériques réseau, les services et applications ajoutés ou déplacés, ou une modification de la politique, ce qui réduit la probabilité de défaillances du réseau en raison d'une configuration ou une politique incohérente ;
- **Augmentation du taux de l'innovation :** l'adoption de SDN accélère l'innovation

12. Réseau overlay : ou réseau superposé est un réseau informatique bâti sur un autre réseau.Les nœuds du réseau overlay sont interconnectés par des liens logiques du réseau sous-jacent. La complexité du réseau sous-jacent n'est pas visible par le réseau overlay.

des entreprises en permettant aux opérateurs de réseaux IT de programmer et reprogrammer le réseau en temps réel pour répondre aux besoins spécifiques de chaque entreprise et aux besoins des utilisateurs.

2.3.5 L'innovation par les applications réseau SDN :

Au cours de ces dernières années l'approche SDN est considérée comme véhicule pour l'innovation du réseau, car elle permet une plus grande flexibilité et l'extensibilité par rapport aux autres architectures réseau existantes.

La séparation du plan de contrôle et de données permet aux deux plans d'évoluer plus indépendamment les uns des autres par rapport à d'autres architectures réseau. Les fonctions et les services réseau peuvent être ajoutés de façon dynamique sous la forme d'applications réseau, ce qui facilite leur d'déploiement. Plusieurs applications de réseau ont déjà été proposées et évaluées par la communauté de recherche SDN. Elles peuvent être résumé dans :

- Les domaines de la gestion de réseau et l'ingénierie du trafic ;
- L'équilibrage de charge (LoadBalancing) pour les serveurs d'applications ;
- La sécurité SDN et le contrôle d'accès au réseau ;
- La virtualisation du réseau et routage inter-domaine [20].

2.3.6 Les bénifices du SDN :

Le SDN permet plusieurs améliorations par rapport au fonctionnement classique [21] :

- La vue globale du réseau par le contrôleur permet de remplacer les protocoles de routage distribués (OSPF [13], EIGRP [14], BGP [15] ...) par des mécanismes plus simples (pas besoin de découvrir la topologie, pas de problème de convergence ...) ;
- Le management est par conception centralisé, ce qui simplifie l'administration d'une infrastructure de grande taille (Cloud computing par exemple) ;
- On peut introduire une nouvelle fonctionnalité très facilement, pas besoin de se soucier des switches, il suffit de l'implémenter sur le contrôleur ;
- Il n'y a plus de problème d'interopérabilité entre les différents matériels réseaux, du moment que ceux-ci comprennent OpenFlow.

13. OSPF : Open Shortest Path First.
14. EIGRP : Enhanced Interior Gateway Routing Protocol.
15. BGP : Border Gateway Protocol, est un protocole d'échange de route.

2.4 Network Functions Virtualization (NFV) :

Nous allons présenter dans cette partie, la technologie NFV, ce qu'elle apport, ses types ainsi que sa relation avec SDN, sans oublié ses avantages.

2.4.1 Définition :

Le NFV est une initiative destinée à virtualiser des fonctions réseau, c'est une nouvelle façon de concevoir, déployer et gérer les services réseau, le NFV dissocie les fonctions réseau (conversion des adresses réseau, pare-feu, détection d'intrusion, services de noms de domaines, mise en cache, etc.) des systèmes matériels propriétaires, pour les exécuter au niveau du logiciel. Le NFV exploite les technologies de virtualisation standard pour consolider de nombreux types d'équipement réseau sur des serveurs, commutateurs et systèmes de stockage standard, l'objectif étant de réduire les besoins du data center en matière d'équipements, d'alimentation et d'espace [24].

Donc le NFV propose d'extraire les fonctions réseaux des équipements dédiés pour les faire fonctionner dans un environnement virtualisé. Pour les opérateurs réseaux, NFV constitue une opportunité pour proposer des services de manière plus agile, capables de fonctionner à des échelles extrêmement importantes, mais surtout de les déployer de manière plus rapide en exploitant les propriétés intrinsèques à la virtualisation [22].

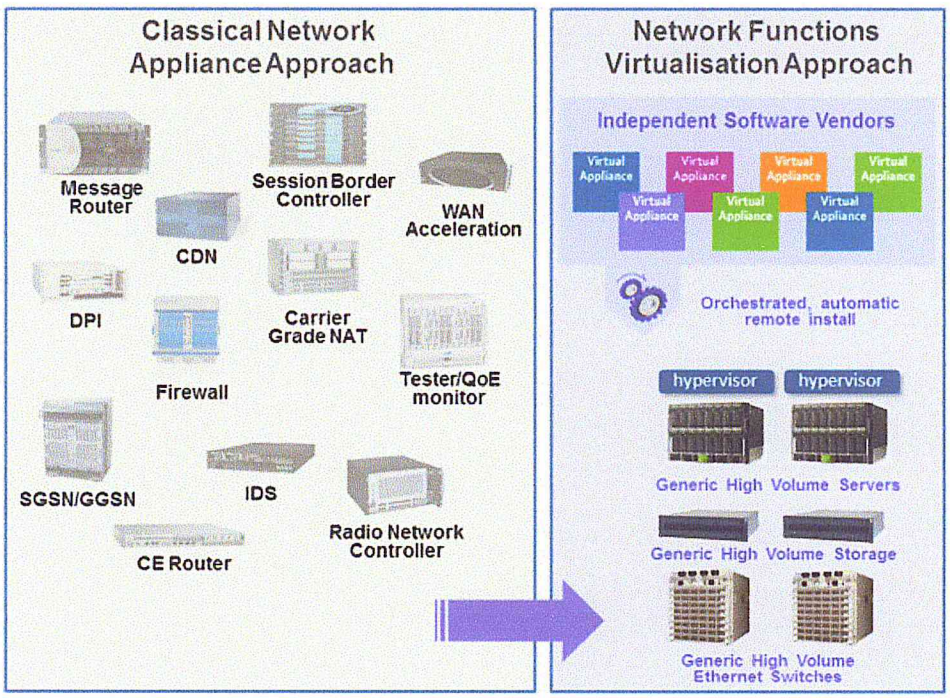


FIGURE 2.3 – l'approche de NFV.

2.4.2 Types de NFV :

Presque toutes les fonctions de réseau peuvent être virtualiser. Le NFV sur le marché aujourd'hui comprend [45] :

- **Commutateurs virtualisés** : ports physiques sont connectés à des ports virtuels sur les serveurs virtuels avec des routeurs virtuels en utilisant les passerelles VPN [16] et des routeurs IPSec [17] et SSL [18] ;
- **Les appareils réseau virtualisés** : fonctions de réseau qui nécessitent aujourd'hui une machine dédiée peuvent être remplacés par un appareil virtuel.
- **Les services réseau virtualisé** : on site par exemple les applications de gestion de réseau tels que l'analyse du trafic, des outils de surveillance de réseau, les équilibreurs de charge et les accélérateurs ;
- **Les applications virtualisées** : presque toutes les applications que vous pouvez imaginer, tels que le stockage virtualisé et les services d'imagerie pour supporter l'explosion de l'usage des tablettes et smartphones.

2.4.3 Les Exigences de NFV :

Pour avoir la possibilité d'implémenter le NFV il y un certain nombre de défis tels que :

- **Général :** une virtualisation partielle ou complète, performances prévisibles.
- **Portabilité :** découplé de l'infrastructure sous-jacente.
- **Performance :** optimisation de l'utilisation des périphériques réseaux et équipements de surveillance.
- **Elasticité :** Adaptable à répondre au service SLA [19]. Mobile vers d'autres serveurs.
- **Résilience :** Être capable de recréer après un échec : taux de perte de paquets spécifié, coupure d'appels , le temps de récupérer, etc.
- **Sécurité :** basé sur l'autorisation et l'authentification.
- **La continuité des services :** continuité après des défaillances ou des migrations.
- **Assurance Service :** horodatage [20] et la transmission des copies des paquets pour la détection des défauts.
- **Modèles de services :** les opérateurs peuvent utiliser l'infrastructure NFV exploités

16. VPN : réseau privé virtuel.
17. IPsec : (Internet Protocol Security), défini par l'IETF comme un cadre de standards ouverts pour assurer des communications privées et protégées sur des réseaux IP, par l'utilisation des services de sécurité cryptographiques.
18. SSL :(Secure Sockets Layer) protocole de sécurisation des échanges sur Internet.
19. SLA :(Service Level Agreement) est le contrat ou la partie du contrat spécifiant l'ensemble des niveaux de services à fournir par le prestataire informatique au(x) client(s).
20. Horodatage : est un mécanisme qui consiste à associer une date et une heure à un événement, une information ou une donnée informatique. Il a généralement pour but d'enregistrer l'instant auquel une opération a été effectuée.

par d'autres opérateurs [25].

2.4.4 Les avantages de NFV :

Le NFV apporte de nombreux avantages au réseau nous pouvant citer [22] :

- La virtualisation offre une grande souplesse et élasticité pour le déploiement ;
- La réduction des coûts de mise en place (CAPEX [21]) s'ajoutent des réductions sur le coût de fonctionnement (OPEX [22]) ;
- Avec NFV, le passage à l'échelle peut être réalisé de manière horizontale (« horizontal scalability ») : par exemple, en cas de congestion d'un service, il est facile de déployer une nouvelle occurrence de ce service, permettant ainsi de s'adapter à la montée en charge ;
- Enfin, la virtualisation permet de s'affranchir de la contrainte géographique en permettant de positionner le service indépendamment des contraintes physiques.

2.4.5 L'approche SDN-NFV :

Les technologies NFV et SDN suscitent un intérêt croissant, les fournisseurs de services sont nombreux à s'interroger sur les conditions de migration vers une infrastructure réseau virtuelle. D'un point de vue purement technique, si la réduction des coûts promise par le NFV fait débat, il est certain que la technologie peut améliorer l'évolutivité et la réactivité du réseau.

Tandis que le SDN consiste à séparer le contrôle et les données pour consolider et automatiser les réseaux distribués à grande échelle, le NFV vise à virtualiser les fonctions du réseau et supporte tous types de services de télécommunication déjà exécutés dans un Cloud IT [23] standard. La « plate-forme réseau » qui en résulte peut éventuellement être ouverte à des développeurs de logiciels tiers pour qu'ils exécutent leurs applications spécifiquement pour les réseaux des opérateurs avec de meilleures performances.

Le NFV découple les fonctions réseau clés, de traduction d'adresses réseau (NAT [24]),

21. CAPEX : les dépenses d'investissement de capital qui se réfèrent aux coûts de développement ou de fourniture des pièces non-consommables pour le produit ou le système.
22. OPEX : les dépenses d'exploitation qui sont les coûts courants pour exploiter un produit, des entreprises, ou un système.
23. IT : (Information Technology) désigne tout le secteur des technologies du traitement de l'information.
24. NAT :(Network address translation), un mécanisme informatique permettant de faire communiquer un réseau local avec l'Internet.

de pare-feu, de détection des intrusions et de distribution de noms de domaine (DNS [25]) notamment, pour permettre l'exécution des logiciels depuis le datacenter. Avec cette configuration, les opérateurs peuvent proposer plus facilement des services IP plus flexibles [26].

Comme le montre la figure 2.4, Network Functions Virtualisation (NFV) est très complémentaire au Software Defined Networking (SDN), mais ne dépend pas forcement d'elle (ou vice-versa). NFV peut être implémentée sans SDN, bien que les deux concepts puissent être combinés.

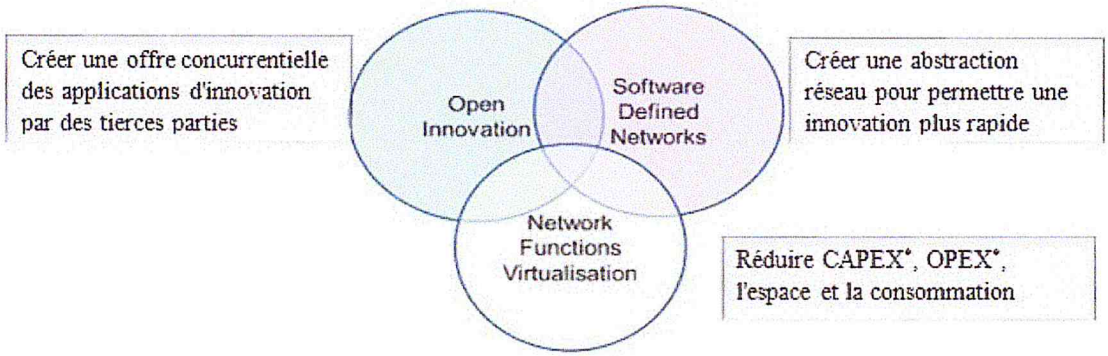


FIGURE 2.4 – La relation entre NFV et SDN.

NFV peut être utilisé sans SDN, en se fondant sur les techniques actuellement utilisées dans de nombreux datacentres. Mais les approches reposant sur la séparation des plans de contrôle et de transmission de données tel que proposé par le SDN peut améliorer les performances, de simplifier la compatibilité avec les déploiements existants, et de faciliter les procédures d'exploitation et de maintenance.

NFV est en mesure de supporter SDN en fournissant l'infrastructure sur laquelle le logiciel SDN peut être exécuté. En outre, le NFV s'aligne étroitement avec les objectifs du SDN pour utiliser des serveurs et des commutateurs [27].

2.4.6 La différences entre SDN et NFV :

Le tableau qui suit présente une petite comparaison entre SDN et NFV [27] :

25. DNS :(Domain Name System) service permettant de traduire un nom de domaine en informatique de plusieurs types qui y sont associées, notamment en adresses IP de la machine portant ce nom.

	SDN	NFV
Raison d'être	Séparation du contrôle et des données, la centralisation du contrôle et la programmabilité du réseau.	Délocalisation de fonctions de réseau des appareils dédiés aux serveurs génériques.
Périphériques cibles	Serveurs et commutateurs.	Serveurs et commutateurs.
Emplacement cible	data center / cloud.	Fournisseur de service réseau.
Applications initiales	Cloud orchestration et le networking (réseautage).	Les routeurs, pare-feu, passerelles, CDN [26], accélérateurs WAN [27], assurance SLA.
Nouveaux protocoles	openFlow.	Aucun pour l'instant.
formalisation	Open networking forum (ONF).	Groupe de travail NFV ETSI [28].

TABLE 2.2 – Comparaison des deux approches SDN et NFV.

2.5 Conclusion :

Dans ce chapitre, nous avons vu l'utilité de l'utilisation de l'approche SDN via le protocole OpenFlow ainsi que le NFV dans une architecture cloud.

Il paraît clair que dans une approche NFV, les fonctions de contrôle centralisé des SDN pourraient être définies comme une fonction virtuelle, de telle sorte que, par exemple, des commutateurs OpenFlow pourraient être pilotés par la couche NFV. En théorie, le contrôleur SDN pourrait être implémenté comme une fonction virtuelle.

Les fonctions de load-balancing (l'équilibrage de charge) sont aussi des cibles pour les NFV, car elles ont des comportements de contrôle et de forwarding [29] de type SDN. Ainsi, si NFV prend en charge le cas général du forwarding à base de règles, il pourrait devenir un sur-ensemble des spécifications SDN.

Dans le chapitre 3, nous allons présenter le load balancing et ses différents algorithmes que nous allons implémenter par la suite ainsi que ses bénéfices dans l'environnement cloud.

26. CDN : (content delivery network) est constitué d'ordinateurs reliés en réseau à travers Internet et qui coopèrent afin de mettre à disposition du contenu ou des données à des utilisateurs.
27. WAN : (Wide area network) réseauétendu.
28. ETSI : (EuropeanTelecommunications Standards Institute) son rôle est de produire des normes de télécommunications pour le présent et le futur.
29. forwarding :consiste à rediriger des paquets réseaux reçus sur un port donné.

3

Synthèses des techniques d'équilibrage de charges

3.1 Introduction :

De nos jours, nous utilisons de plus en plus des machines de type multi-processeurs ou multi-cœurs .ce qu'engendrent de nombreux problèmes par exemple :la capacité de distribuer la charge sur un certain nombre de systèmes ou de serveurs pour augmenter la performance globale de traitement des demandes entrantes .

Ici parvient Le réseau SDN, une technologie qui dissocie le réseau de la couche matérielle sous-jacente et lui permet de mieux prendre en charge les environnements virtualisés. En outre, la technologie NFV (virtualisation des fonctions réseau) tient ses promesses en virtualisant les fonctions réseau sans nécessiter de matériel propriétaire physique. Donc avec une architecture SDN, nous sommes en mesure de proposer un framework [1] flexible pour fournir des fonctions réseau virtualisées, qui pourraient inclure l'équilibrage de charge (LoadBalancing) qui fait la distribution de manière intelligente sur toute ces unité disponible (serveurs cloud ou VMs).

L'objectif final est d'assurer une haute disponibilité et de permettre l'amélioration des performances d'un réseau sans forcément augmenter ses capacités, en gérant mieux la charge de requêtes arrivant sur les ressources à disposition.

Dans ce chapitre on présentera le principe de l'équilibrage de charge, ces techniques et solution existantes, on citera aussi les différents algorithmes pour l'équilibrage de charge ainsi que ses avantages.

3.2 Définition :

Le principe de base de l'équilibrage de charge (en anglais load balancing) consiste à effectuer une distribution des tâches à des machines de façon intelligente. Pour cela il faut intégrer un dispositif entre les serveurs et les utilisateurs de la ressource. Ce dispositif (load balancer) aura pour tâche de rediriger les consommateurs de services en fonction de l'état d'occupation des serveurs.

En d'autre terme, Les requêtes arrivent au dispositif qui les distribue vers les serveurs en fonction de différents algorithmes (Round Robin, Random, Least Resources...). Une fois traitées, ces requêtes retournent soit au dispositif soit directement au demandeur [29].

1. Framework : est un ensemble d'outils et de composants logiciels organismes conformément à un plan d'architecture et de modèle.

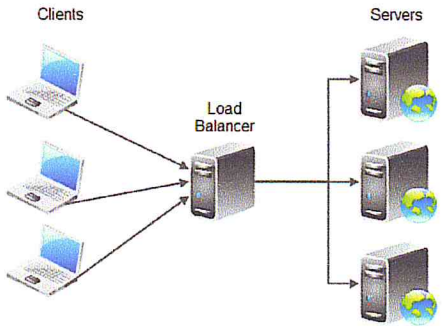


FIGURE 3.1 – Equilibrage de charge avec un Load Balancer.

3.3 La procédure générale de LoadBalancing :

La figure 3.2 illustre le mode de fonctionnement d'un Load Balancer exécuté au niveau d'un contrôleur SDN. Au lancement du contrôleur, le Load Balancer se met en état d'attente. Lorsque le contrôleur reçoit une requête destinée au Load Balancer, celui-ci vérifie la liste des serveurs disponibles et sélectionne un serveur selon la méthode d'équilibrage de charge implémentée. Ensuite, il modifie l'en-tête de la requête en remplaçant l'adresse de destination par l'adresse (IP et MAC) du serveur choisi [23].

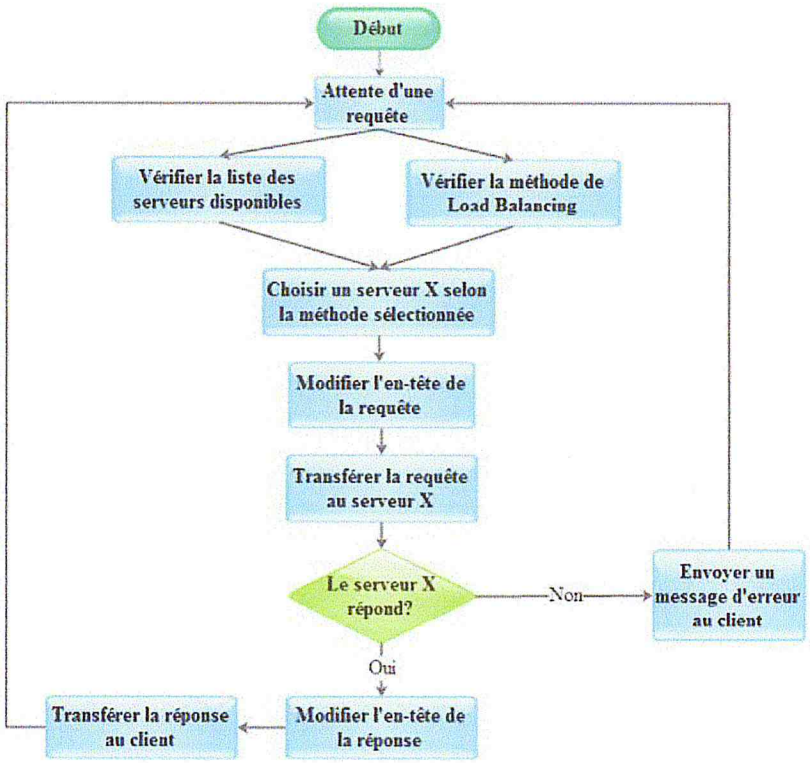


FIGURE 3.2 – Le processus général de Load Balancing.

A la réception de la réponse, le Load Balancer modifie l'en-tête en remplaçant cette fois-ci l'adresse source (IP et MAC) par la sienne. Dans le cas où le serveur ne répond pas, un message d'erreur sera transmis au client par le contrôleur.

3.4 Les techniques de loadbalancing dans l'environnement cloud :

Dans l'environnement cloud computing, l'équilibrage de charge est nécessaire pour atteindre un minimum de temps de réponse et un débit élevé du système. Pour un environnement cloud, diverses approches d'équilibrage de charge ont été proposées telles que [31] :

- **Server-base load balancing :** utilisé pour les serveurs web distribués. Elle utilise un protocole qui limite le taux de redirection afin d'éviter une surcharge des serveurs distants et utilise un middleware pour soutenir ce protocole. Elle utilise aussi une heuristique pour tolérer des changements brusques de charge.
- **Scheduling strategy on Load balancing :** utilisée pour établir l'équilibrage de charge des ressources des VM .Cette stratégie réalise un temps de migration dynamique réduit à l'aide d'un algorithme génétique ce qui fait une meilleure utilisation de ressource.
- **Biased Random Sampling :** utilisée dans les systèmes cloud à grande échelle. Elle réalise l'équilibrage de charge entre tous les nœuds du système en utilisant un échantillonnage aléatoire (RandomSampling) du domaine du système.
- **Active Clustering :** utilisée dans les systèmes cloud à grande échelle. Elle Optimise l'affectation du travail en connectant les services similaires par un re-câblage local.
- **Honeybee-based load balancing technique :** utilisée dans les systèmes cloud à grande échelle. Elle réalise un équilibrage de charge globale à travers une action sur un serveur local.

3.5 Les différentes solutions existantes du loadbalancing :

Il existe plusieurs solutions pour faire de l'équilibrage de charge, parmi eux [29] :

1. **DNS :**

   Dans cette solution le répartiteur de charge est le serveur DNS. Basé sur la configuration de BIND [2], le serveur DNS permet de résoudre les noms d'hôtes de manière différente à chaque requête. Un nom de domaine est associé à plusieurs adresses IP correspondant à chaque serveur du cluster. Bind, utilise alors un algorithme round-robin pour choisir le serveur destinataire de la requête. L'Avantage de ce système est sa grande simplicité, il suffit d'ajouter quelques lignes à la configuration de bind sur le serveur DNS. Par contre il ne permet pas de prendre en compte les performances d'un serveur, ou bien son taux d'occupation, puisque qu'il n'utilise qu'un algorithme round-robin.

2. **Pen :**

   Pen est un load balancer pour les protocoles basés sur TCP [3]. Il permet de distribuer les requêtes de clients sur différents serveurs en gardant une trace de ceux-ci pour renvoyer chaque client vers le serveur qui lui avait été affecté précédemment. Il intègre également un dispositif simple de haute disponibilité qui, si un serveur est hors service, envoie la requête vers un autre.

3. **Balance :**

   Balance est une solution simple qui offre une solution d'équilibrage de charge mais aussi de proxy TCP. Son Load Balancing est basé sur plusieurs algorithmes (RoundRobin, random, hash, least resources). Il offre une gestion totale via la ligne de commande en plus d'être très léger. Il existe en deux versions, une gratuite (balance) et une commerciale (balanceng).

3.6 La nature des algorithmes de loadbalancing :

Il existe deux grands types d'algorithmes d'équilibrage : les algorithmes d'équilibrage de charges dit statiques (dans un environnement cloud statique), que l'on peut définir à la compilation, et les algorithmes dit dynamiques (dans un environnement cloud dynamique), calculé pendant l'exécution du programme [32].

2. BIND :Berkeley Internet Name Daemon, parfois Berkeley Internet Name Domain) est le serveur DNS le plus utilisé sur Internet.
3. TCP : (Transmission Control Protocol) est un protocole de transport fiable.

- **Algorithmes statique :**

Les algorithmes d'équilibrage de charge statiques prennent la décision d'attribution de la charge à des VM dans un systéme basé sur la connaissance à priori des applications et des ressources du systéme. Les algorithmes d'équilibrage de charge statiques sont non préemptifs [4] et donc chaque machine a au moins une tâche assignée pour elle-même. Son objectif est de minimiser le temps d'exécution, le délai et limiter la surcharge de communication.

Example : Round Robin (RR), Weighted Round Robin (WRR).

- **Algorithme Dynamique :**

Les algorithmes dynamiques peuvent affecter de nouvelles ressources à des tâches en cours d'exécution. Concrètement, le système permet aux tâches en cours d'exécution d'être transférées d'une machine chargée à une machine moins chargée. Les algorithmes dynamiques sont particulièrement appropriés aux systèmes à mémoire partagée, en raison de la facilité procurée par ces systèmes pour migrer une tâche en cours d'exécution.

Example : Least Connection (LC), Weighted Least Connection (WLC).

3.7 Les différents algorithmes pour l'équilibrage de charge :

Les algorithmes définissent l'ordre dans lequel le load-balancer affectera les requêtes des clients aux différents serveurs. Il existe plusieurs algorithmes parmi eux [33] :

- **Round Robin :** ou 'tourniquet' est comme son nom l'indique un algorithme de file tournante : l'équilibrage choisit à chaque instant un serveur dans la file et boucle sur celle-ci.
- **Random :** après chaque flux transmis au contrôleur, celui-ci sélectionne au hasard un serveur à partir de la liste des serveurs pour traiter la demande du client.
- **Temporal Round-Robin :** pour chaque flux transmis au contrôleur et à chaque intervalle de temps Δt, le contrôleur sélectionne un serveur pour traiter la demande du client selon un ordre circulaire.
- **Le Least-Connection (LC) :** dirige les requêtes vers le serveur ayant le moins de connexions établies. IL compte le nombre de connexion de chaque serveur pour estimer sa charge.

4. Un algorithme préemptif : une tâche élue peut perdre le processeur au profit d'une autre jugée plus prioritaire. Par contre, dans un algorithme non préemptif, l'ordonnanceur n'arrête pas une tâche élue.

3.8 Les paramètres de performance affecté par le load-balancing :

Il existe plusieurs algorithmes de LoadBalancing pour l'amélioration et l'optimisation des paramètres de performance du cloud qu'on peut résumer dans le tableau 3.1 [34] :

Paramètres	Illustration
Débit	Le Débit est utilisé pour calculer le pas des tâches dont l'exécution est terminée. Il devrait être élevé pour améliorer les performances du système.
La tolérance aux pannes	c'est la capacité d'exécution correcte et uniforme même dans des conditions de défaillance au niveau de n'importe quel nœud dans le système.
Le temps de réponse	C'est le temps nécessaire pour répondre à une requête par un système distribuée qu'exécute un algorithme de loadbalancing spécifique .ce paramètre doit être minimisé.
L'utilisation des ressources	c'est le degré auquel les ressources du système sont utilisées. Un bon algorithme fournit une utilisation maximale des ressources.
La performance	elle représente l'efficacité du système après l'exécution de l'équilibrage de charge. Si tous les paramètres précédents sont satisfaits de façon optimale, alors la performance du système sera hautement améliorée.
Le temps de migration	le temps de transfert d'une tâche d'une machine à une autre. Ce temps doit être minimal pour améliorer la performance du système.
La surcharge associée	le volume de surcharge produit par l'exécution de l'algorithme. Un minimum de surcharge est nécessaire pour une mise en œuvre réussite de l'algorithme.
L'extensibilité	C'est la capacité d'un algorithme pour effectuer de l'équilibrage de charge d'un système avec un nombre fini de processus. Cette mesure devrait être améliorée.

TABLE 3.1 – Les paramètres de performance affecté par le loadbalancing.

3.9 Les bénéfices du LoadBalancing :

L'équilibrage de charge est un élément important lors de la mise en place de services amenés à croître. Il faut s'assurer que la capacité à monter en charge soit la plus optimale possible afin d'éviter toute dégradation que ce soit en terme de performances ou de fiabilité [30].

Les gains sont non négligeables :

- amélioration des temps de réponse des services ;
- ajout de nouveaux serveurs sans interruption de service ;
- Une meilleure performance, qu'on ne peut obtenir avec un seul serveur ;
- Une meilleure utilisation des ressources.

3.10 Conclusion :

Dans ce chapitre on a présenté le principe de l'équilibrage de charge, les techniques existantes, ainsi que les algorithmes du load balancing qui ont pour but d'équilibrer la charge entre les serveurs pour optimiser leur utilisation et diminuer le temps d'exécution des requêtes. On a étudié aussi les paramètres de performances ainsi que les avantages de l'équilibrage de charge dans un environnement cloud.

Le load balancing permet de résoudre des problématiques de gestion des ressources, de performances et de tolérance aux pannes, en utilisant des techniques permettent à la fois de répondre à une charge trop importante d'un service en la distribuant sur plusieurs serveurs, et de réduire l'indisponibilité potentielle de ce service que pourrait provoquer la panne logicielle ou matérielle d'un unique serveur.

Dans le chapitre 4, nous allons détailler la conception de SDN ainsi qu'une technique d'équilibrage de charge dans un réseau SDN/NFV.

4

Conception d'une technique flexible d'équilibrage de charge pour un environnement SDN-NFV

4.1 Introduction :

La Virtualisation des Fonctions Réseau (NFV en anglais) est un élément déterminant pour optimiser l'utilisation des ressources du réseau en « virtualisant » des fonctions habituellement mises en œuvre dans le matériel propriétaire, réduisant ainsi pour les opérateurs les coûts d'investissement et d'exploitation. La solution NFV est basée sur le principe de séparation entre une couche matérielle et une couche logicielle applicative implémentant des fonctions nécessaires au fonctionnement du réseau d'un opérateur (services IP de routage et communications multimédia, fonctions de terminaux virtuels, fonctions d'accès réseau, etc ...) [42].

Les architectures de matériel de routage et switching IP évoluent en parallèle suivant la standardisation poussée par l'ONF (Open Networking Foundation) appelée SDN (Software Defined Networking), visant à séparer la couche contrôle de la couche données, avec la mise en place d'un protocole « ouvert » appelé Openflow, permettant à la couche de contrôle d'inter-opérer avec des matériels de constructeurs différents. Le SDN est donc complémentaire de la technologie NFV et permet de mettre en place des solutions purement logicielles rendant possible le contrôle d'un réseau.

Dans ce chapitre nous allons expliquer le concept de la configuration d'un environnement SDN/NFV adoptés avec un protocole de communication standard OpenFlow. En précisant le comportement de chaque composant et la connexion entre eux. Par la suite nous allons présenter les stratégies que nous avons adopté pour la technique de load balancing dans une plateforme SDN/NFV.

4.2 Présentation des approches adoptées :

4.2.1 L'approche SDN :

C'est un modèle d'architecture réseau qui ajoute un niveau d'abstraction aux fonctionnalités des équipements réseau afin de pouvoir les gérer de façon globale. La partie décisionnelle des équipements est séparée de leur partie opérationnelle et déportée vers un unique point de contrôle qui dirige l'ensemble de façon cohérente. Ce Découplage permet de déployer le plan de contrôle sur des plateformes dont les capacités sont plus grandes que celles des commutateurs réseaux classiques. Enfin, Cette abstraction à travers une API Réseau standard permet un développement de services réseaux.

Il y a trois composantes importantes qui définissent l'architecture de SDN :

- La décorréllation [1] du plan de contrôle et du plan de données.
- Déploiment du plan de contrôle sur des plateformes du plus grande capacités.
- La Programmabilité du réseau a partir des API [35].

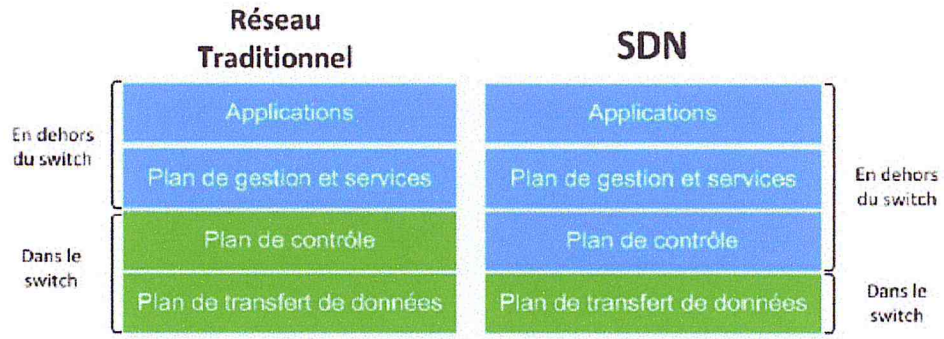


FIGURE 4.1 – Le nouveau paradigme des réseaux.

SDN permet d'avoir une abstraction globale du réseau, c'est-à-dire que le contrôleur permet de regrouper le plan de contrôle de différents switches en une seule entité. Cela implique une centralisation du contrôle et une facilité de programmation réseau puisque le développeur interagit avec les différents commutateurs comme s'il s'agissait d'un seul.

La différence entre SDN et les réseaux traditionnels, c'est qu'en déplaçant le plan de contrôle dans la partie logicielle, SDN permet un accès et une administration dynamique du réseau.

En résumé, le SDN :

- Est un modèle de réseau basé sur un contrôleur.
- Le contrôle du réseau est directement programmable.
- Management centralisé : l'intelligence du réseau est centralisée dans un logiciel appelé SDN controller, qui maintient une vue globale du réseau.
- Configuration automatique : SDN permet aux administrateurs réseaux de configurer, administrer, sécuriser et optimiser les réseaux rapidement grâce à des programmes SDN [35].

1. décorréllation : la séparation.

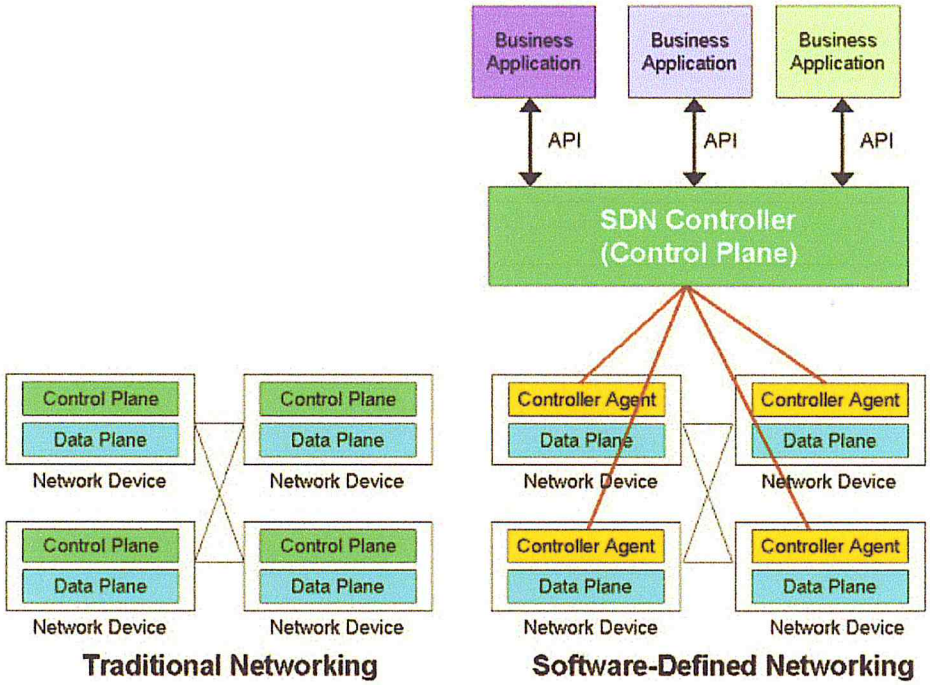


FIGURE 4.2 – Différence entre un réseau traditionnel et un réseau SDN.

Pour mettre en œuvre cette approche, nous avons choisi OpenFlow comme protocole de communication entre les switches et le contrôleur qui est actuellement la méthode la plus importante qui met en œuvre le concept SDN. C'est un protocole définit entre les couches de contrôle et de données d'une architecture SDN. Il permet de séparer le plan de données et le plan de contrôle, et il offre une configuration flexible des plans de transmission (Forwarding Plane) [2]. La technologie OpenFlow est utilisée pour la mise en œuvre de la virtualisation de réseaux dans l'environnement de Cloud Computing.

4.2.2 L'approche OpenFlow :

Le protocole de communication le plus avancé entre un plan de contrôle logiquement centralisé (un ou plusieurs contrôleurs) et le plan de données (des commutateurs réseaux) est OpenFlow. Il est standardisé par l'Open Networking Foundation (ONF) et implémenté par de nombreux équipementiers, dont HP [3], Cisco [4], IBM. Dans OpenFlow, les décisions de routage sont prises par le contrôleur pour chaque flux de données et poussées dans les switches sous forme de simples instructions de commutation [36].

2. Forwarding plane : dans le routage, le plan de transmission, parfois appelé le plan des données, définit la partie de l'architecture du routeur qui décide quoi faire avec les paquets arrivant sur une interface d'entrée.

3. HP : (Hewlett-Packard Company), est une entreprise multinationale américaine initialement d'électronique et d'instrumentation qui a évolué vers l'informatique, les imprimantes, les Serveurs & Réseaux et lemultimédia.

4. Cisco :est une entreprise informatique américaine spécialisée, à l'origine, dans le matériel réseau (routeur etcommutateur ethernet), et depuis 2009 dans les serveurs et les réseaux.

Les Messages Openflow :

***ofpt_hello* :**

Le switch initie la connexion avec le contrôleur en TCP lorsque la connexion est établie, le switch et le contrôleur envoient, tous les deux, un message *ofpt_hello* avec la version Openflow approprié [35].

***Feature Echo_Request* :**

Est utilisé pour échanger des informations sur la latence [5] et la bande passante. Echo Request Timeout indique la déconnexion [35].

***Feature Echo_Reply* :**

Le Switch répond avec ce message en indiquant l'ID de datapath [6] et les capacités du switch, que le contrôleur pourrait accepter ou refuser [35].

***ofrp_set_config* :**

Message envoyé par le contrôleur. Il contient la longueur du paquet maximale que le switch devrait envoyer au contrôleur [35].

***Packet-in* :**

Ce message se compose d'un entête, suivi d'un buffer-id, une valeur unique indiquant le numéro du buffer où le paquet est stocké [35].

***FlowMod* :**

Ceci est l'un des principaux messages, il permet au contrôleur de modifier l'état d'un switch OpenFlow. Tous les messages *FlowMod* commencent avec l'en-tête d'OpenFlow standard, contenant la version et le type des valeurs appropriées, suivis par la structure FlowMod [35].

***Packet_out* :**

Message envoyé par le contrôleur au switch. Ce message permet au contrôleur d'injecter le paquet reçu dans le plan de données du switch [35].

***ofpt_error* :**

Le commutateur est en mesure de notifier le contrôleur des problèmes en utilisant des messages d'erreur [35].

5. latence : Il désigne le temps nécessaire à un paquet de données pour passer de la source à la destination à travers un réseau.

6. datapath : est un ensemble d'unités fonctionnelles qui effectuent le traitement des données des opérations.

4.3 Le choix du contrôleur SDN :

Le contrôleur SDN embarque un certain nombre de directives qui contrôlent le comportement des éléments de réseau sous-jacents, de sorte que le réseau fournira les services souhaités.
Les contrôleurs SDN open source les plus populaires sont : NOX, POX, RYU, Beacon, Floodlight et OpenDaylight [37].

- **NOX :** le contrôleur NOX fait partie de la première génération des contrôleurs OpenFlow, il a été développé initialement par l'entreprise Nicira Networks, il permet de développer des applications de contrôle réseau. Nicira a donné NOX à la communauté de la recherche scientifique en 2008.
- **POX :** le contrôleur POX est une implémentation en Python du contrôleur NOX, il a été créé pour avoir une plateforme SDN rapide pour le prototypage [7] et l'expérimentation. Il offre de meilleures performances par rapport aux applications Python déployées sur NOX.
- **RYU :** est un contrôleur SDN capable de configurer les équipements réseaux en utilisant OpenFlow, NetConf [8] et OF-config [9].
- **Beacon :** est un contrôleur OpenFlow rapide, multiplateforme, modulaire, basé sur Java.
- **Floodlight :** est un contrôleur OpenFlow basé sur Java soutenu par une communauté de développeurs open source. Il a été créé à partir du contrôleur Beacon.
- **OpenDaylight :** est un contrôleur SDN basé sur Java qui fournit une plateforme de programmation réseau complète pour SDN. Il a été créé comme un projet de collaboration hébergé par la Fondation Linux en 2013. IL contient un contrôleur modulaire et souple.

7. Le prototypage : est la démarche qui consiste à réaliser un prototype. Ce prototype ne simule que quelques aspects du logiciel et peut être très différent du produit final.
8. NetConf : Network Configuration Protocol, est un protocole de gestion du réseau développé et standardisé par l'IETF.
9. OF-Config : OpenFlow Configuration and Management Protocol, est un ensemble de règles qui définit un mécanisme pour les contrôleurs OpenFlow permettant d'accéder et de modifier la configuration des données sur un commutateur OpenFlow.

	NOX	POX	RYU	Floodlight	OpenDaylight
Langage	C++	Python	Python	Java	Java
Performance	Haut	Bas	Bas	Haut	Haut
OpenFlow	1.0	1.0	1.0, 1,2-1,4	1.0, 1.3	1.0, 1,3
Courbe d'apprentissage [10]	Modéré	Facile	Modéré	Excessif	Excessif
Virtualisation	Mininet et Open vSwitch	Mininet et Open vSwitch	Minnet et Open vSwitch	Minnet et Open vSwitch	Minnet et Open vSwitch
Open source	Oui	Oui	Oui	Oui	Oui
Plateforme	Linux	Linux,Mac OS et Windows	Linux	Linux,Mac Os et Windows	Linux

TABLE 4.1 – Caractéristiques des contrôleurs SDN [37].

Dans notre travail, nous avons choisi le contrôleur POX pour les raisons suivantes :

- IL est multiplateforme.
- Il supporte Python qui est un langage simple et portable.
- Il est le plus utilisé pour la recherche, l'expérimentation et la démonstration.
- POX a des composants réutilisables qui permettent la détection de topologie, la sélection de chemin.

4.4 Combinaison des approches SDN et NFV :

Les technologies SDN et NFV sont complémentaires, où NFV est destinée à optimiser le déploiement des fonctions et des équipements de réseau (telles que DNS, pare-feu, équilibreurs de charge "load balancer", etc.), et SDN se concentre sur la configuration automatique des réseaux. Ainsi que les quatre points communs entre eux :

1. les deux proviennent de la demande de flexibilité et d'interopérabilité des clients ;
2. la synchronisation : ces concepts ont émergé l'un après l'autre ;
3. La coexistence : ces technologies peuvent être utilisées ensemble ;
4. Les technologies NFV et SDN sont utilisées conjointement avec d'autres nouvelles technologies.

10. Courbes d'apprentissage : La courbe d'apprentissage de la plateforme de contrôle est une métrique fondamentale à considérer lors du démarrage d'un projet. Elle mesure l'expérience nécessaire pour connaitre la plateforme du contrôleur SDN.

C'est pour cela Nous avons choisi de combiner les approches SDN et NFV pour atteindre une plus grande agilité de réseau tout en réduisant les dépenses d'exploitation et les dépenses d'investissement [43].

4.5 Connexion d'un switch OpenFlow au contrôleur :

Cas n° 1 : connexion réussite

Au démarrage, le commutateur initie un canal sécurisé SSL avec le contrôleur sur TCP qui est utilisé par le contrôleur pour gérer le commutateur via le protocole openflow. Il faut renseigner l'adresse IP du contrôleur au niveau du switch. Une fois la connexion SSL est établie, le switch OpenFlow envoie un paquet *OFPT_HELLO* avec le numéro de version d'OpenFlow supporté. Le contrôleur vérifie la version d'OpenFlow supporté par le switch et lui répond par un *OFPT_HELLO* en indiquant la version d'OpenFlow avec laquelle ils communiqueront.
Le contrôleur demande les caractéristiques du commutateur connecté par l'intermédiaire d'un message de demande de caractéristiques (*features request message*). Le commutateur recevant ce message répond avec un message de réponse de caractéristiques (*reply message*) et informe ainsi le contrôleur de ses fonctionnalités prises en charge [38].

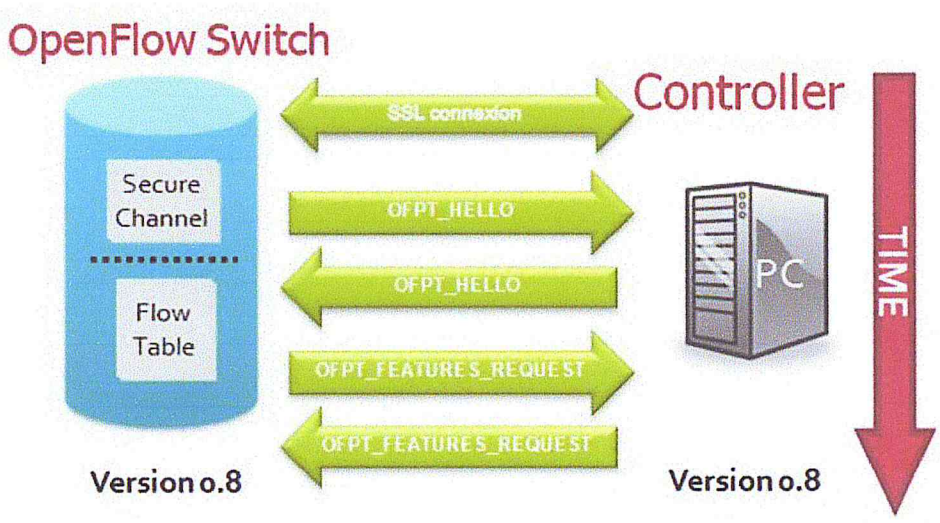


FIGURE 4.3 – Connexion entre le switch et le contrôleur.

Cas n° 2 : échec de connexion

Comme dans le cas précédent le switch OpenFLow envoie son paquet *OFPT_HELLO* avec la version 0.5, le contrôleur s'aperçoit qu'il ne supporte pas la version OpenFlow du switch. Il lui retourne donc un paquet *OFPT_ERROR* en indiquant que c'est un problème de compatibilité [38]. Ce mécanisme est expliqué dans la figure 4.4 :

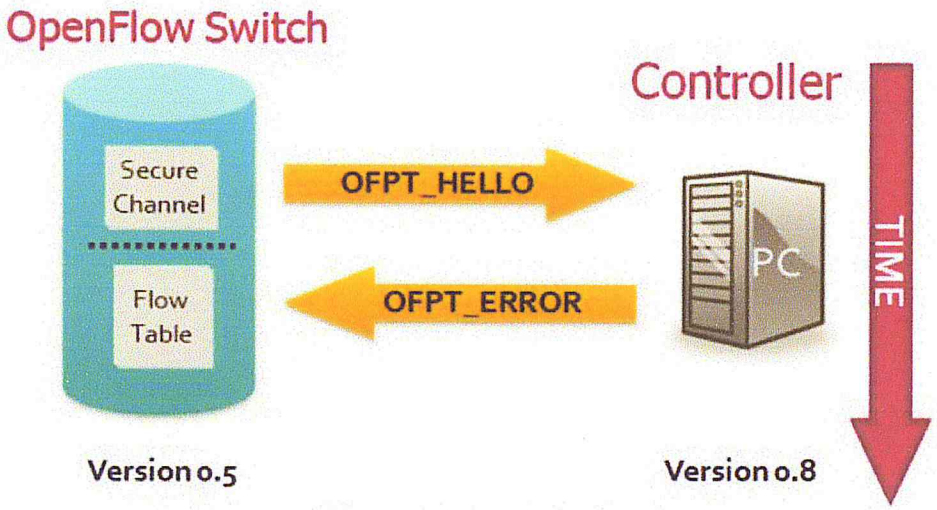


FIGURE 4.4 – Echec de connexion .

4.6 Fonctionnement du commutateur OpenFlow :

Les éléments de base du switch OpenFlow sont les tables OpenFlow. Ces tables vont assurer la classification des paquets en se basant sur plusieurs champs des entêtes.

Dans les réseaux traditionnels, un commutateur vérifie l'adresse MAC de destination dans sa table MAC pour transférer la trame sur le port de sortie. Par contre, un commutateur OpenFlow contient une ou plusieurs tables de flux (flow tables). Comme le montre la Figure 4.5, chaque table de flux contient un ensemble d'entrées de flux (flow entries) où chaque entrée contient des champs de correspondances, des compteurs, et des actions.

Pour les paquets reçus sur le chemin de données, le commutateur effectue le matching (qu'il essaie de faire correspondre le port d'entrée et les en-têtes des paquets avec les champs de correspondance dans les différentes entrées de flux). Dans le cas où des entrées définies sont similaires, un champ de priorité indique quelle entrée doit être choisie.

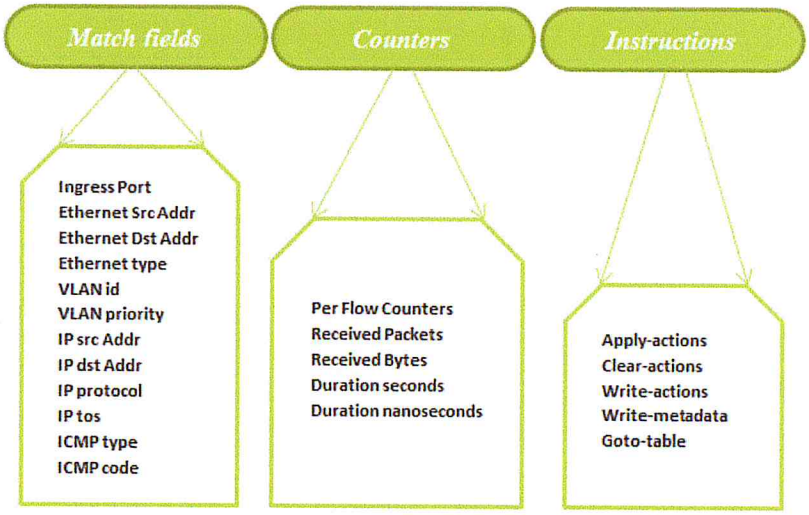


FIGURE 4.5 – Les éléments d'une entrée de flux dans la table de flux.

- Match fields : Utilisés lors de la recherche de l'entrée correspondante au paquet. Ils sont constitués essentiellement des entêtes des paquets et des ports d'entrées.
- Counters : Servent essentiellement à garder des statistiques sur les flux pour ensuite décider si une entrée de flux est active ou non. Pour chaque table, chaque flux, chaque port, chaque file d'attente et chaque groupe de tables, des compteurs de statistiques sont maintenus.
- Instructions : Représentent l'ensemble des instructions OpenFlow qui servent à modifier le traitement que va subir le paquet. Les instructions supportées sont :

1. **Apply-Actions :** Pour appliquer les actions sur le paquet immédiatement.
2. **Clear-Actions :** Pour supprimer une liste des actions du paquet.
3. **Write-Actions :** Ajouter une liste d'actions au paquet.
4. **Write-Metadata :** Ajouter des données utiles pour le séquencement entre les tables OpenFlow.
5. **Goto-Table :** Indique que le paquet doit être acheminé vers une table d'indice supérieur.

Un flux de paquets à l'entrée du switch va être traité par une succession de tables (qui est de 2 types : une qui ne contient que le traitement d'OpenFlow, et une autre (hybride) qui ajoute à ce traitement le traitement normal du switch pour le flux de production par exemple). Ce traitement spécifiera le comportement des paquets vis-à-vis de l'ensemble des "Flow Tables". Les instructions d'une entrée de flux peuvent explicitement diriger le paquet vers un autre "Flow Table" (via l'instruction "Goto"). Si l'entrée de flux ne

dirige pas le paquet vers une autre table, le traitement s'arrête à cette table. Dans ce cas le paquet subira les actions accumulées qui lui sont associées durant le traitement. Si le paquet ne correspond à aucune entrée de flux, le comportement du switch vis-à-vis de ce paquet dépendra de la configuration de la table ; le comportement par défaut c'est d'envoyer le paquet au contrôleur avec un message *OFPT_PACKET_IN*, ou bien une autre option qui consiste à supprimer le paquet. La table peut aussi spécifier que dans ce cas le paquet doit continuer son chemin dans le processus de traitement vers la table suivante. La figure 4.6 présente le diagramme de traitement des paquets dans un réseau OpenFlow [39].

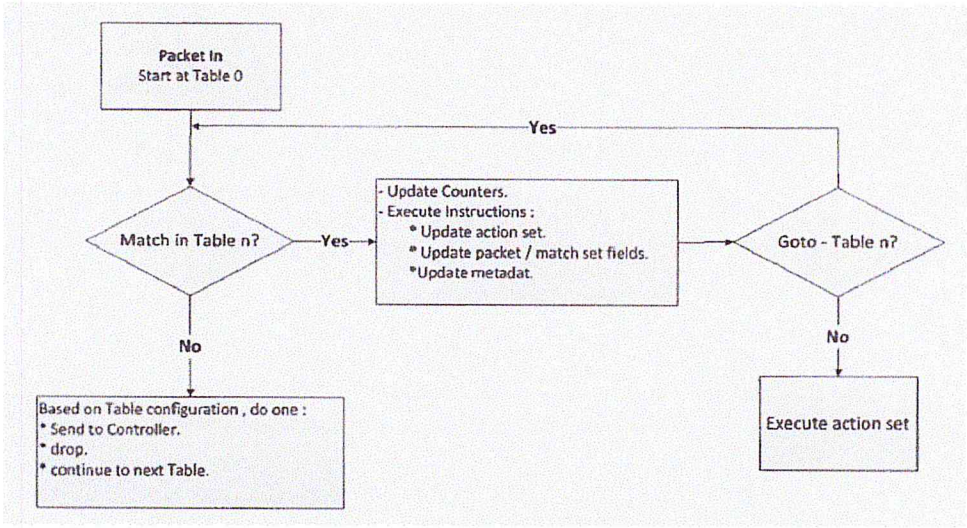


FIGURE 4.6 – Traitement des paquets dans un réseau OpenFlow.

4.7 Fonctionnement du contrôleur SDN :

Pour gérer les entrées de flux dans la table de flux (flow table) du commutateur OpenFlow, le contrôleur SDN utilise le canal sécurisé [40].

• Lors de la réception d'un *«packet-in»* du commutateur, le contrôleur examine l'en-tête et vérifie si une nouvelle entrée de flux doit être crée ou pas. Si c'est le cas, le contrôleur envoie un message *«flow-mod»* au commutateur qui installe l'entrée de flux correspondante.

• On outre, le contrôleur envoie un message *«packet-out »* qui indique au commutateur d'envoyer le paquet sur un port de sortie spécifique.

Le contrôleur peut ajouter, modifier et supprimer les entrées de flux dans les tables de flux en utilisant le protocole OpenFlow. La figure 4.7 présente une partie de l'algorithme qui montre le fonctionnement du switch et du contrôleur openFlow.

Switch	Contrôleur
Receive packet ; *Check flow table ;* ***If*** *{* *The packet rule is in the flow table ;* *Send out the packet following the rule ;* *}* ***Else*** *Send packet to the controller*	*Receive packets from the switch* ***If*** *Non Ethernet packet // from the switch* *Packet dropped ;* ***Else*** *{* *Check Mac and Ip address* *Send rule to the switch // flow-mod* *(exemple: source: 10.0.0.1 – destination: 10.0.0.3-port:1)* *}* *Switch can send the packet to the destination.*

FIGURE 4.7 – Le comportement du switch et du contrôleur openFlow.

4.8 Exemple de communication dans un réseau SDN en utilisant le protocole OpenFlow :

La figure 4.8 présente un exemple d'un un réseau OpenFlow composé de deux hôtes, d'un commutateur et d'un contrôleur [40][35].

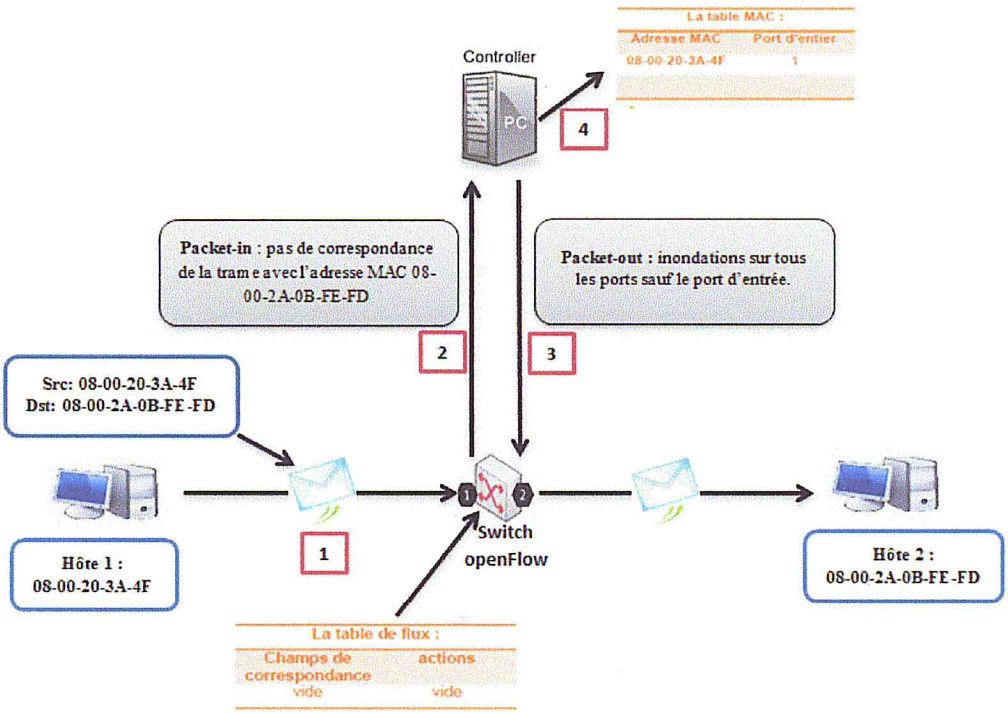


FIGURE 4.8 – La communication *(packet-in)* entre deux hôtes dans un réseau SDN.

Pour bien comprendre l'exemple, nous allons expliquer les étapes numérotés point par point :

1. L'hôte 1 veut envoyer un paquet à l'hôte 2.
2. Au début, la table de flux est vide, donc le commutateur envoie le paquet au contrôleur connecté.
3. Le contrôleur décide des actions qui doivent être appliquées pour ce nouveau flux et comment ce premier paquet est gèré. Dans le cas où le contrôleur implémente le comportement d'apprentissage du commutateur, l'action à entreprendre pour le message *packet-out* est l'inondation sur tous les ports du commutateur OpenFlow à l'exception du port d'entrée.
4. En outre, le mappage de l'adresse MAC de la source au port d'entrée est appris par le contrôleur, mais aucun flux d'entrée n'est installé sur le commutateur puisque l'adresse MAC de destination n'a pas encore été apprise. Les entrées de flux sont installées via les messages *flow-mod* une fois les mappages pour les deux adresses ont été appris.

Le sens inverse de cette communication va remplir la table de flux du commutateur. La figure 4.9 explique ce mécanisme :

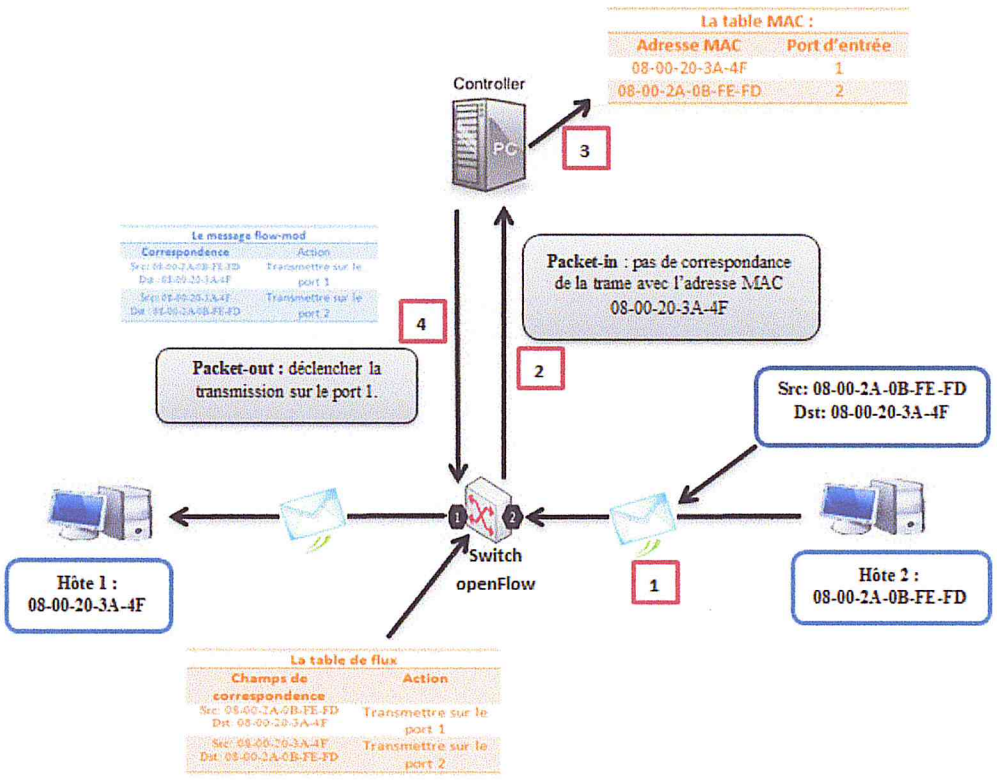


FIGURE 4.9 – La communication *(packet-out)* entre deux hôtes dans un réseau SDN.

1. L'hôte 2 envoie maintenant un paquet de retour à l'hôte 1 qui arrive de nouveau au commutateur OpenFlow.
2. Puisque la table de flux est toujours vide, le commutateur envoie le paquet au contrôleur connecté.
3. Le contrôleur apprend le mappage de l'adresse MAC de la source avec le port d'entrée et cherche le mappage pour l'adresse MAC de destination.
4. Actuellement, les deux mappages sont disponibles donc, le contrôleur installe les entrées de flux correspondantes dans la table de flux du commutateur connecté via les messages *« flow-mod »* et déclenche la transmission du paquet sur le port 1 par un message *« packet-out »* vers le commutateur connecté. Les paquets subséquents de ce flux peuvent être envoyés sans interactions avec le contrôleur grâce aux entrées installées dans la table de flux du commutateur.

4.9 Utilisation de la technique d'équilibrage de charge (load balancing) :

Le Load Balancing (LB) peut être décrit comme la distribution de calcul et de communication de façon égale entre les ressources, ou la division de nombreuses requêtes des clients entre plusieurs serveurs. Dans la majorité des cas, un module physique de load balancing est installé sur les serveurs ou les VMs. Ce module distribue le trafic sur les serveurs en fonction d'un groupe de règles d'équilibrage définies [46].

On a besoin d'une gestion dynamique des ressources réseau pour une performance élevée et une faible latence de transmission de données, et cela on utilisant les systèmes d'équilibrage de charge pour pouvoir créer des infrastructures capables de distribuer la charge entrante sur plusieurs serveurs cloud ou VMs.

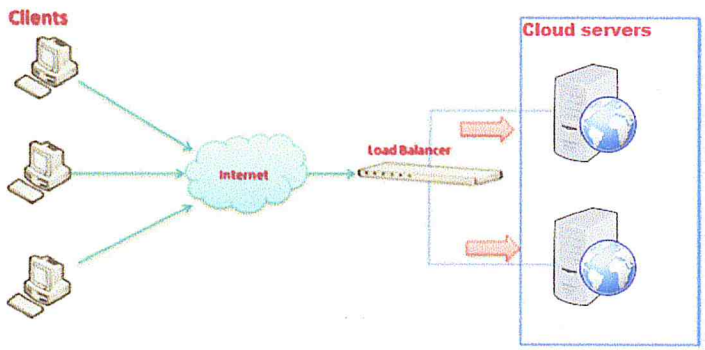


FIGURE 4.10 – Equilibrage de charge avec un load banacer.

Nous avons implémenté trois stratégies de Load Balancing : Random, Round-Robin et Temporal Round-Robin (voir l'annexe 2).

- Random :

Une répartition purement aléatoire, qui peut donner un équilibrage satisfaisant, sur des volumes importants. Après chaque flux transmis au contrôleur, celui-ci sélectionne au hasard un serveur à partir de la liste des serveurs pour traiter la demande du client [41].

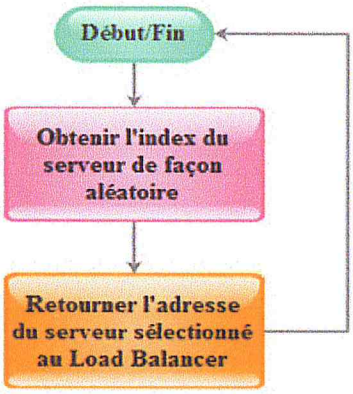


FIGURE 4.11 – L'algorithme Random Load Blanacer.

- Round-Robin :

L'expression signifie une ronde, une permutation circulaire. Il s'agit d'une affectation cyclique : A puis B puis C puis A puis B puis C, pour chaque flux transmis au contrôleur, il sélectionne un serveur pour traiter la demande du client selon un ordre circulaire [41].

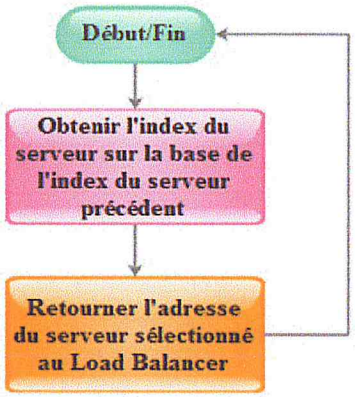


FIGURE 4.12 – L'algorithme Round-Robin Load Blanacer.

- Temporal Round-Robin :

Pour chaque flux transmis au contrôleur et à chaque intervalle de temps Δt, le contrôleur sélectionne un serveur pour traiter la demande du client selon un ordre circulaire.

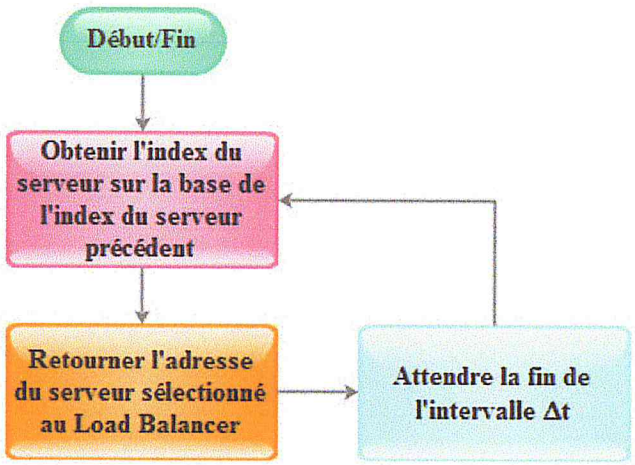


FIGURE 4.13 – L'algorithme Temporal Round-Robin Load Blanacer.

4.10 Les topologies SDN/NFV avec un load balancer :

Pour étudier la performance du load balancing, nous avons utilisé plusieurs topologies SDN qui seront détaillé dans le chapitre d'implémentation.

Les hôtes et les serveurs sont connectés aux commutateurs Open vSwitch qui maintient une liaison sécurisée avec le contrôleur OpenFlow, celui-ci héberge le Load Balancer pour recevoir les stratégies d'acheminement de paquets (entrées de flux).

La Figure 4.14 montre que le Load Balancer est intégré dans le contrôleur POX. Quand une requête d'un client arrive, le contrôleur notifie la stratégie du Load Balancing qui va choisir un serveur pour établir une connexion réseau entre le serveur et le client afin qu'ils puissent communiquer selon la procédure générale de Load Balancing.

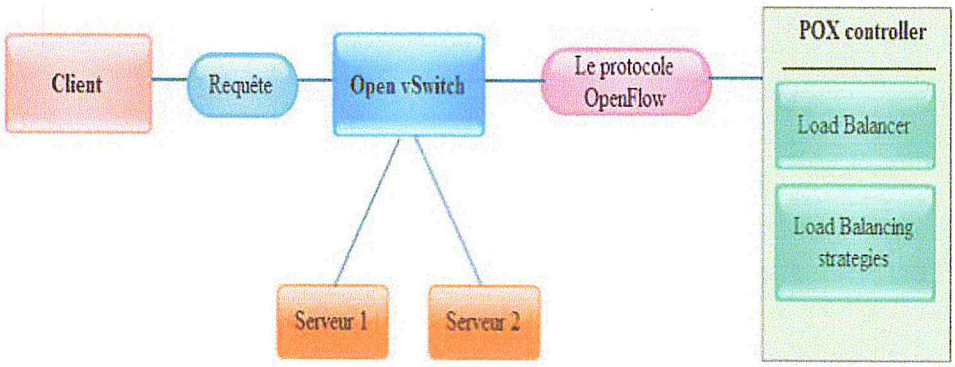


FIGURE 4.14 – Les unités fonctionnelles du Load Balancer.

Pour cela, on a mis en place une étude comparative de différentes approches de load balancing dans un réseau utilisant SDN et NFV pour évaluer la capacité et la performance du réseau. On a choisi les outils suivants :

- Un contrôleur POX et un émulateur MININET pour atteindre l'approche SDN ;
- Des commutateurs virtuels Open vSwitch ;
- Les trois algorithmes de load balancing cités précédemment.

4.11 Conclusion :

À partir de ce chapitre, nous avons présenté les différentes approches réseau, comme l'approche SDN en la comparant avec les réseaux traditionnels pour voir ce qu'elle apporte de nouveau.

Ensuite, l'openFlow qui est la base d'un SDN, permet une programmation simplifiée via une interface standard pour les périphériques réseau. La facilité de programmation permet de concevoir une couche de contrôle stable et performante, afin de centraliser l'intelligence dans le réseau et de fournir la programmabilité promise par SDN. Puis nous avons expliqué comment combiner NFV avec SDN, en expliquant son intérêt.

En dernier, nous avons présenté les stratégies du load balancing tout en concevant un Load Balancer pour effectuer des simulations et apporter des analyses qu'on va détailler dans le chapitre qui suit.

5

Implémentation et évaluation de performances

5.1 Introduction :

Dans ce chapitre nous allons d'abord faire la mise en place et la configuration d'un environnement SDN/NFV, afin de voir l'intérêt et le comportement d'un load balancer lors de son implémentation.

Après plusieurs configuration, nous procèderons à la création de deux types de topologie (single, linéaire) pour les utilisé dans la partie de simulation de tests en appliquant les 3 algorithmes du load balancing choisi (sur une topologie simple) ainsi que deux nouvelle approches que nous allons proposer (pour une topologie linéaire), dans le but d'analyser les paramètres de performance pour déterminer quelle est la meilleure approche observée.

5.2 Présentation de l'environnement de travail :

5.2.1 Installation et simulation :

Les tests et simulations ont été effectués sur un ordinateur portable ayant les caractéristiques suivantes :

- Processeur : Intel® Core(TM) i5 CPU M 560 @2.67GHz 2.67GHz.
- RAM : 3.00 GO.
- OS : windows 7.
- VirtualBox.
- Ubuntu 14.04.2 LTS sous VirtualBox avec les capacités suivantes :

- Processeur : CPU 1 Core.
- RAM : 786 Mo.

- MININET : un émulateur SDN OpenFlow sous Ubuntu.
- POX : un contrôleur SDN utilisant Python.
- Open vSwitch : un commutateur virtuel.

5.2.2 Environnement de développement :

Pour développer les approches de LoadBalancing que nous avons proposé nous avons choisi le Python [1], car ce langage est portable et libre (open source), sa syntaxe est facile à apprendre et permet de programmer plus rapidement et plus efficacement, en plus les scripts ou les modules sous Python ont une extension « py ».

1. Apprendre le Python : https ://openclassrooms.com/courses/apprenez-a-programmer-en-python

5.2.3 L'émulateur MININET :

A. Présentation :

Mininet [2] est un émulateur réseau qui permet à l'utilisateur de créer facilement, personnaliser, partager et tester des applications SDN ainsi que la simulation d'une plateforme SDN-NFV. Il est disponible gratuitement, c'est un logiciel open source qui émule les périphériques OpenFlow et les contrôleurs SDN. Il permet de créer des hôtes, des commutateurs et des contrôleurs via : Ligne de commande, Interface utilisateur interactive ou bien Application Python [35] [49].

Un hôte MININET se comporte exactement comme une véritable machine, nous pouvons exécuter différents programmes. Ces programmes peuvent envoyer des paquets, en indiquant la vitesse et le délai des liaisons. Parmi les avantages de mininet les plus importants, est qu'il adopte un comportement proche du réseau réel [47].

A partir de mininet :
- Le déploiement se fait en quelques secondes.
- Les topologies réseaux sont personnalisables avec un script python.
- Un hôte peut faire tourner des processus (wireshark).
- Les switches sont programmables en utilisant le protocole OpenFlow.

B. Installation :

Après l'installation d'Ubuntu 14.04.2 LTS sur une machine virtuelle (Virtual Box) considérée comme notre serveur, il va falloir installer l'émulateur Mininet.

Il existe 4 méthodes pour l'installer :
- Installation de l'image mininet (VM).
- Installation via des commandes (à partir de la source).
- Installation depuis les packages.
- Mise à jour de l'émulateur mininet existant [51].

Lors de notre déploiement, le choix s'est porté sur l'installation via des commandes (voir Annexe 1).

5.2.4 Le contrôleur SDN POX :

Une fois le Mininet installé, nous passons au contrôleur POX, qui est un Framework d'interaction avec les commutateurs SDN en utilisant le protocole OpenFlow, avec ce

2. Telechargement : http ://mininet.org/download/

contrôleur nous pouvons créer un contrôleur SDN en utilisant le langage de programmation python [50]. Nous l'avons choisi pour mettre en place l'application d'équilibrage de charge.

5.2.5 Open vSwitch :

Comme son nom l'indique, ce logiciel permet de créer des commutateurs (switches) virtuels. Avec les services qui se virtualisent de plus en plus, la gestion des interconnexions entre les machines virtuelles (et les machines réelles) nécessite une solution performante pour manipuler ce transit de paquet IP, d'où l'idée de faire des commutateurs virtuels. L'objectif d'Open vSwitch est d'obtenir un commutateur ayant les mêmes fonctionnalités qu'un vrai switch administrable et pouvant s'étendre sur plusieurs serveurs physiques dans le cadre de la virtualisation [48].

5.3 Mise en place des approches du load balancing :

MININET est une plate-forme d'émulation de réseau qui est très utile pour tester les applications SDN. Il peut prendre en charge différents types de topologies. Nous allons présenter deux configurations qui seront utiles pour nos tests.

5.3.1 Une simple topologie :

Pour débuter, et pour une simple utilisation nous avons commencé avec une topologie simple.

A. Présentation :

La topologie simple c'est la topologie la plus courante actuellement, où tous les hôtes sont reliés à un seul équipement central : le concentrateur réseau. Ici le concentrateur réseau est un switch.

Les avantages de ce réseau : est que la panne d'un hôte ne cause pas la panne du réseau, ainsi nous pouvons retirer ou ajouter facilement un hôte sans perturber le réseau. Il est aussi très facile à mettre en place. Pour les inconvénients, le coût est un peu élevé, la panne du concentrateur centrale entraine le disfonctionnement du réseau.

B. Configuration :

Dans cet exemple nous avons choisi une simple topologie de 3 hôtes (h1, h2, h3), un switch open vSwitch (s1) et un contrôleur POX (c0). Par la suite, nous allons varier le nombre d'hôtes pour effectuer nos tests. Nous désignons les hôtes H1 et H2 comme

serveurs et le reste comme clients dans le cas où nous lançons le contrôleur avec le Load Balancer.

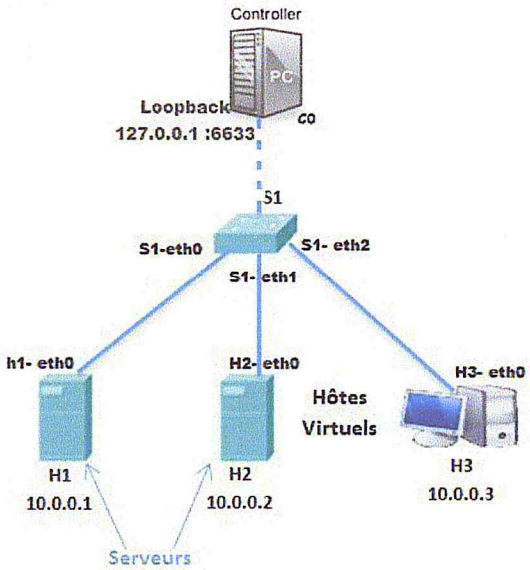


FIGURE 5.1 – La topologie simple avec un seul switch.

La commande suivante génère un switch unique avec 3 hôtes qui lui sont attachés (figure 5.1). Les hôtes se verront attribuer des adresses IP statiques et les adresses MAC.

sudo mn –arp –topo single,3 –mac –switch ovsk –controller remote

Dans la commande ci-dessus, il y a quelques mots clés importants :

- *Mn* : Cette simple commande permet de réaliser un réseau avec deux hôtes, un vswitch et un contrôleur.
- *Single, 3* : pour crée une simple topologie de 3 hôtes.
- – *arp* : pour remplir les tables ARP des hôtes à l'avance.
- – *mac* : L'adresse MAC de chaque host sera égal à son adresse ip(exemple : 10.0.0.1 → 00 :00 :00 :00 :00 :01).
- – *switch ovsk* : pour lancer un switch OpenFlow de type Open vSwitch.
- – *controller remote* : pour se connecter au contrôleur POX distant.

Le résultat de l'exécution de la commande précédente est présenté dans la figure 5.2.

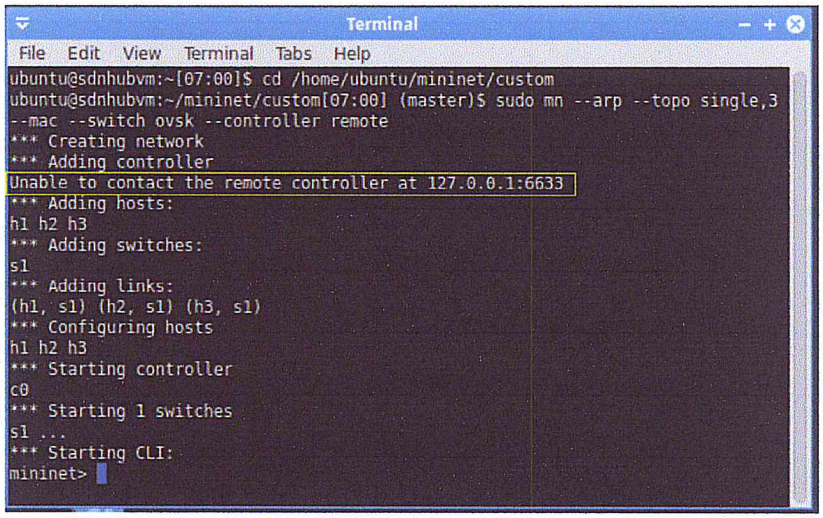


FIGURE 5.2 – Création de topologie Mininet.

Nous remarquons que le switch OpenFlow n'est pas encore connecté avec le contrôleur distant POX qui utilise par défaut le port 6633 sur l'adresse de bouclage 127.0.0.1, car celui-ci n'a pas encore été lancé. Ce dernier peut être lancé en utilisant une commande en précisant le comportement d'acheminement du trafic souhaité. Dans cette exemple nous avons choisi de lui indiquer d'ordonner au switch OpenFlow de se comporter comme un switch de niveau 2.

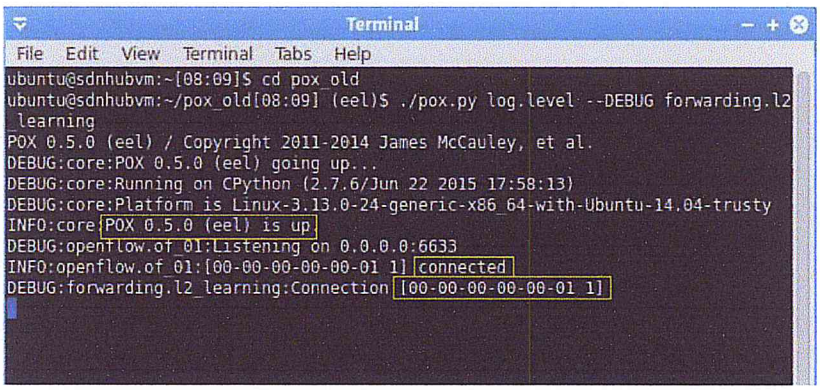


FIGURE 5.3 – Lancement du contrôleur SDN POX.

Avec ce module *" l2_learning.py "*, le contrôleur POX a ordonné au switch de se comporter comme un switch de niveau 2 donc il se base sur les adresses MAC pour orienter les paquets.

- le mot clé *« is up »* indique que le contrôleur est lancé.
- Le mot clé *« connected »* indique que la connexion entre le contrôleur et le switch OpenFlow a été établie.

- [00-00-00-00-00-01] est l'identifiant ou le DPID [3] du switch (s1) et le numéro *'1'* qui suit le DPID est l'ID de l'objet de connexion (spécifique à Python) que le contrôleur POX utilise pour communiquer avec le switch.

Maintenant, les hôtes (h1, h2, h3) peuvent communiquer entre eux, on peut le tester en utilisant la commande ***pingall*** qui teste la connectivité en essayant de pinger entre tous les hôtes du réseau. La figure 4.5 montre le résultat de ce teste.

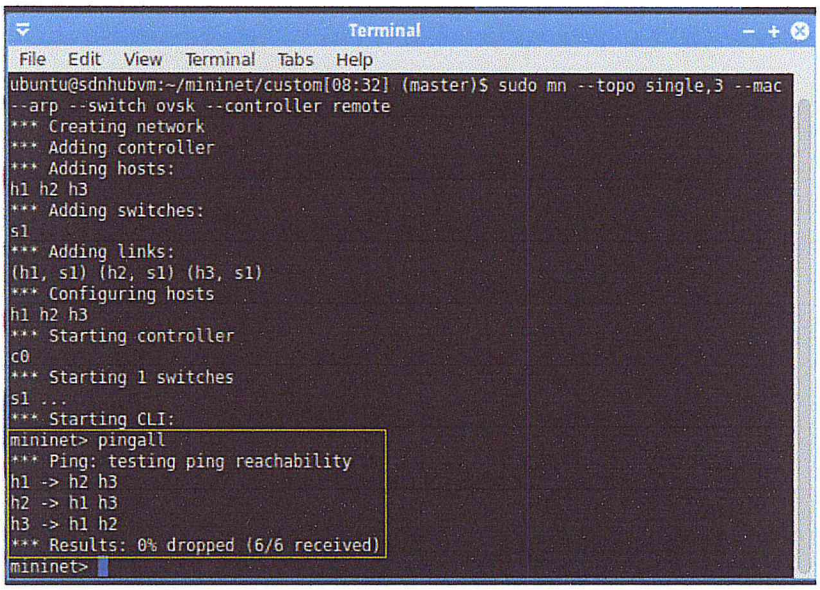


FIGURE 5.4 – Teste de connectivité entre les hôtes.

Dans le cas de l'utilisation du contrôleur POX avec un Load Balancer, le traitement des requêtes sera le suivant :

Au début nous devront préciser que au niveau du contrôleur notre Load Balancer possède une adresse IP/MAC virtuelle (VIP [4]/ MAC (10.0.0.252 /00.00.00.00.00.EF). Donc quand le switch OpenFlow reçoit pour la première fois une requête d'un client qui a comme adresse de destination 10.0.0.254 (l'adresse fixée au LB), il envoie cette requête vers le contrôleur POX sous forme de *packet-in*, car il ne sait pas encore quoi en faire. A la réception du *paket-in* le contrôleur l'envoi au LoadBalncer qui traite en premier lieu la requête ARP du client en lui indiquant son adresse ARP virtuelle. Ensuite il traite la requête IP du client selon la stratégie de LoadBalancing choisi.

3. DPID : OpenFlowDatapath Identifier, chaque instance OpenFlow d'un commutateur est identifiée par un Datapath Identifier.
4. VIP : Virtual Internet Protocol.

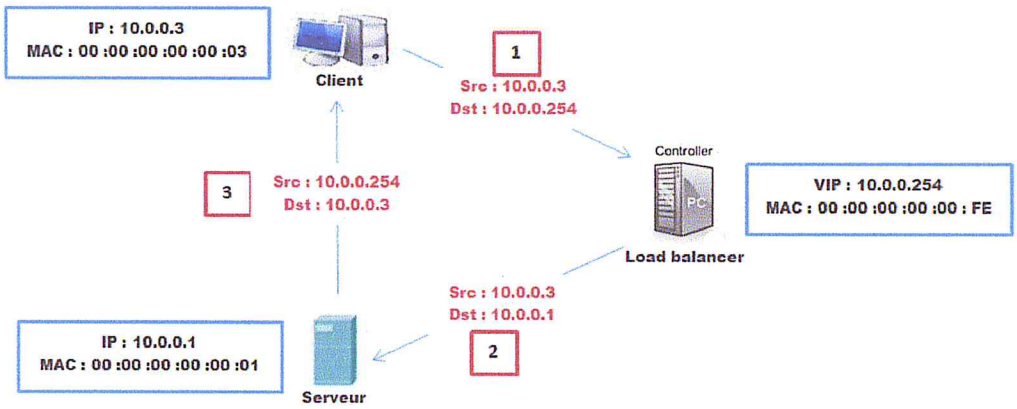


FIGURE 5.5 – Envoie de requête entre le client et le load balancer.

La figure 5.5 explique le traitement d'une requête IP par le Load Balancer qui se déroule comme suit :

1. Le client envoie la requête à l'adresse virtuelle du Load Balancer.
2. Le Load Balancer choisie le serveur à qui expédier la requête selon la stratégie de LoadBalancing implémentée (Round-Robin, Random, . . .).
3. Le Load Balancer envoie deux messages *flow-mod* au switch : le premier lui indique d'installer l'entrée de flux de la route inverse et précise une action qui est de modifier l'adresse source (IP et MAC) de la réponse par celle du Load Balancer. Le second lui indique d'installer la route entre le client et le serveur avec une action qui modifier l'adresse de destination de la requête par celle du serveur sélectionné, tout en lui passant la requête IP. (voir la table de flux du switch S1 dans la figure 5.7).
4. Le switch modifie l'adresse de destination du paquet par celle du serveur (la seconde entrée de flux), et fait le transfert de la requête IP sur le port adéquat.
5. Le serveur répond, et cette fois-ci le switch envoie la réponse au client sans contacter le Load Balancer, cela en suivant la première entrée de flux ; c'est à dire qu'il modifie l'adresse source de la réponse par celle du Load Balancer.

Nous remarquons que le contrôleur POX a dirigé la requête du client vers le serveur 1 dont l'adresse IP est 10.0.0.1. La figure 5.6 illustre le résultat.

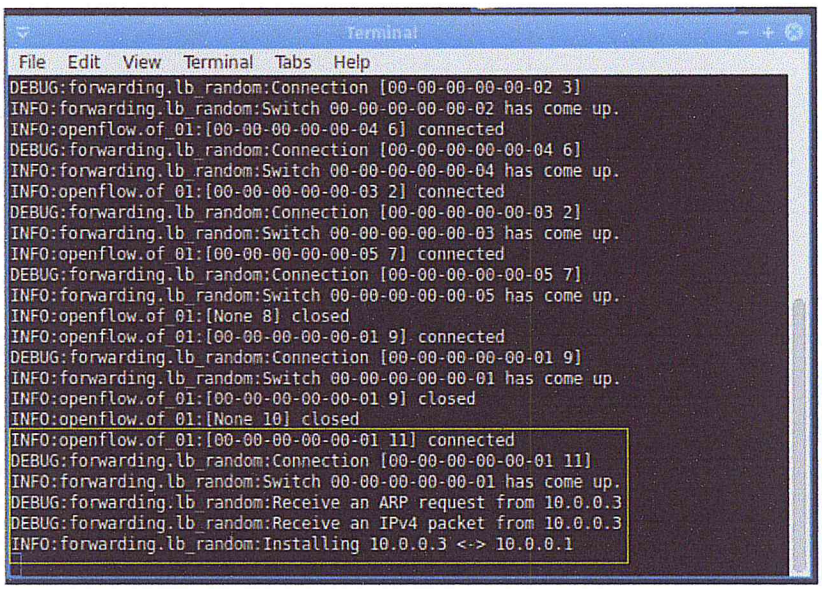


FIGURE 5.6 – déviation de la requête du client vers le serveur 1.

L'un des avantage du LB c'est que maintenant le client peut contacter le load balancer sans se préoccuper de savoir quel serveur va traiter sa requête.

La commande ***dpctl dump-flows*** permet de voir la table de flux d'un switch. La figure 5.7 illustre le résultat de l'exécution de cette commande.

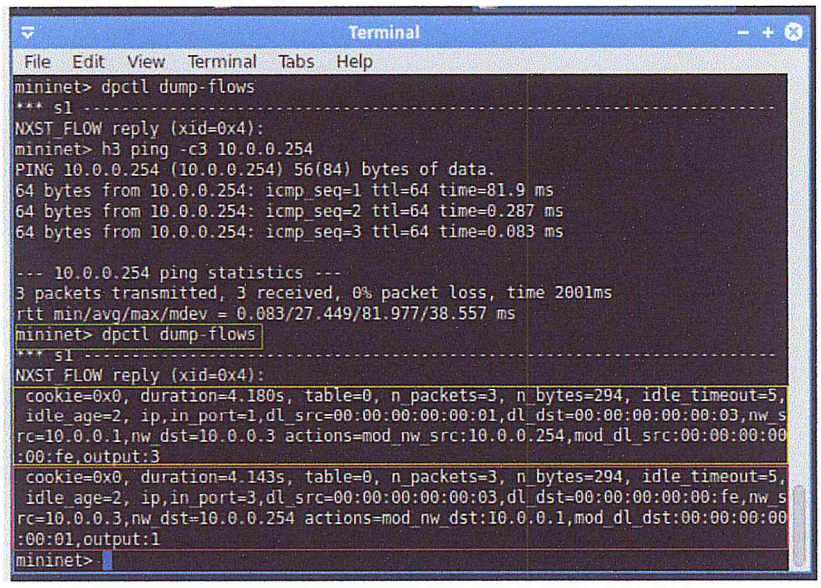


FIGURE 5.7 – La table de flux du switch S1.

Pour expliquer les entrées de flux de switch S1, on explique les mots clés de la figure 5.7 :

- *Duration, n_packets, etc.* : des compteurs.

- *ip* : le type du paquet à traiter.
- *in_port* : le port d'entrée du paquet au switch.
- *dl_src* : l'adresse MAC de la source.
- *dl_dst* : l'adresse MAC de la destination.
- *Nw_src* : l'adresse IP de la source.
- *nw_dst* : l'adresse IP de la destination.
- *actions* : les actions à entreprendre dans le cas où le paquet correspond aux champs de l'en-tête (Header Fields) cités précédemment.

5.3.2 Une topologie à multi-saut :

A. Présentation :

Pour mieux voir et évaluer la performance du contrôleur POX avec et sans le Load Balancer. Nous avons varié le nombre de switch en utilisant une topologie à multi-saut (topologie linéaire), en précisant que chaque switch est relié avec un seul hôte. La figure 5.8 illustre la topologie linéaire.

Cette topologie a pour avantage son faible coût de déploiement, et comme inconvénient la défaillance d'un switch peut scinder le réseau en deux sous-réseaux.

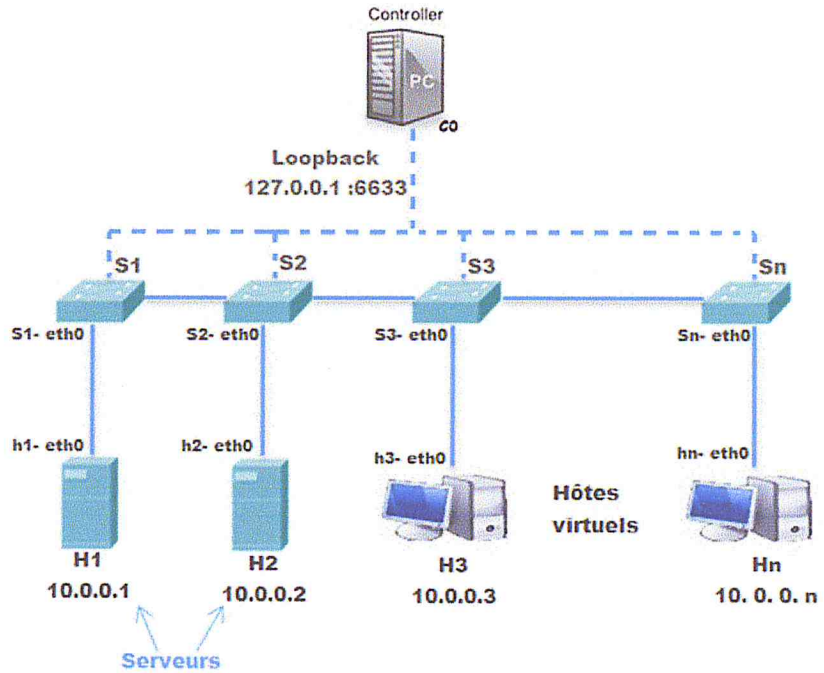


FIGURE 5.8 – La topologie linéaire avec plusieurs switches.

B. Configuration :

La commande suivante génère une topologie à plusieurs switches, dans notre exemple nous avons choisi 10 pour indiquer le nombre de switch, où chaque switch est relié avec un seul hôte avec le port 1 et au contrôleur POX avec l'interface de bouclage 127.0.0.1. Les hôtes se verront attribuer des adresses IP statiques et les adresses MAC.

Sudo mn –topo linear,10 –mac –arp –switch ovsk –controller

- *Linear* : pour créer une topologie linéaire à multi-sauts.

Toutes les autres options sont similaires à la commande précédente (section 5.3.1).

Le lancement du contrôleur POX avec cette topologie est similaire au lancement sur une simple topologie. Il peut ordonner aux switches de se comporter comme un Hub qui effectue des inondations de la requête sur tous les ports en choisissant le module «***hub.py***», ou un switch de niveau 2 en choisissant le module « ***l2_ learning.py*** ».
nous avons proposé deux nouvelles approches d'équilibrage de charge dans une topologie SDN à multi-sauts que nous allons détailler.

Approche 1 : combinaison du load balancing avec l'inondation

Dans cet exemple, la figure 5.9 présente le comportement du loadbalancing avec l'inondation (flooding ou bien le hub).

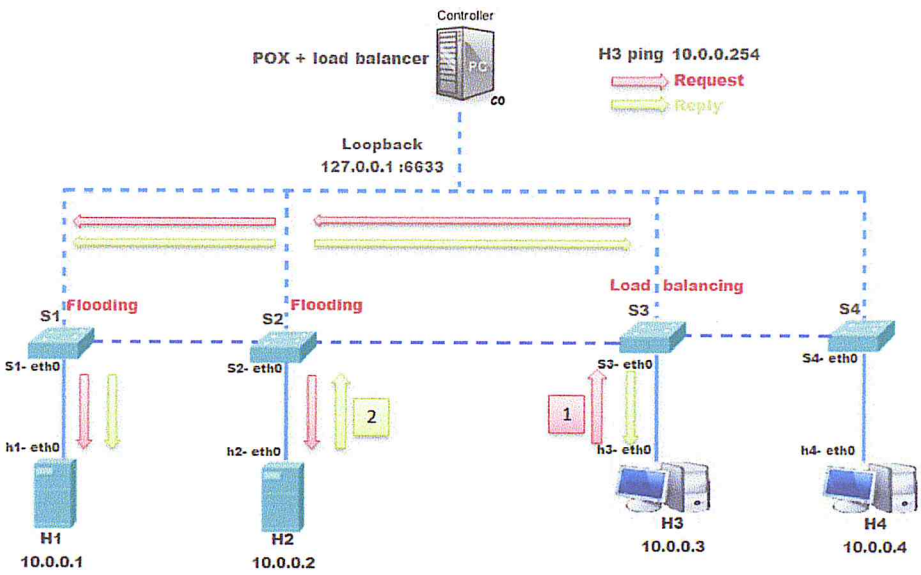


FIGURE 5.9 – Combinaison du load balancing avec le flooding.

1. Le client h3 contacte le load balancer.
2. Quand le switch S3 reçoit la requête du client h3 avec la VIP du Load Balancer comme adresse de destination, il contacte le Load Balancer au niveau du contrôleur pour avoir les entrées de flux nécessaires afin de diriger la requête vers un serveur choisi (dans cet exemple h2).
3. Quand les autres switches (S2 et S1 dans cet exemple) reçoivent la requête, ils contactent le contrôleur C0 (POX) en lui envoyant un *packet-in* pour avoir les entrées de flux nécessaires.
4. Le contrôleur va leurs ordonner (par l'intermédiaire d'un message *flow-mod*) d'adopter le comportement d'un *Hub* et d'inonder la requête.
5. Le serveur h1 vérifie l'adresse de destination de la requête et trouve qu'elle ne correspond pas à son adresse alors il la supprime, tandis que h2 voit qu'il est le destinataire et répond donc au client.
6. Pour le chemin de retour, le comportement des switches reste le même pour acheminer la réponse jusqu'à h3.

Approche 2 : combinaison du load balancing avec la technique d'apprentissage

Dans cet exemple, la figure 5.10 présente le comportement du load balancing avec l'apprentissage (learning) qui signifie que le switch de niveau 2 ou de niveau 3 apprenne les routes et remplie sa table au fur et à mesure de recevoir des requêtes et des réponses. Aprés le remplissage de la table de flux, Les paquets subséquents de ce flux peuvent être envoyés sans interactions avec le contrôleur.

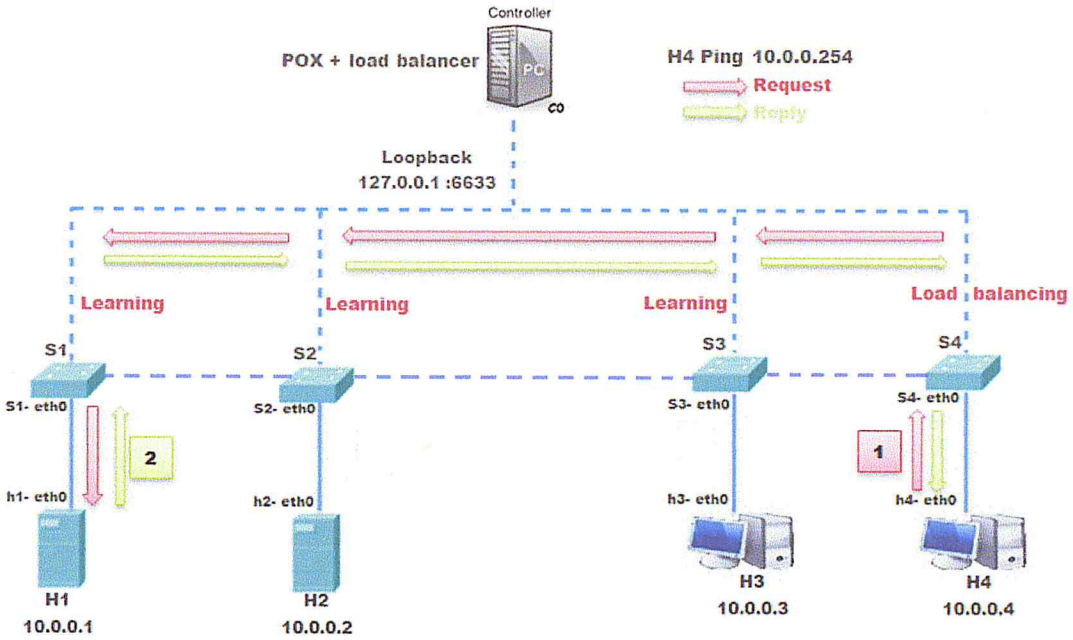


FIGURE 5.10 – Combinaison du load balancing avec l'appretissage.

1. Le client h4 contacte le Load Balancer.
2. Le switch S4 reçoit la requête du client h4 destinée au Load Balancer, il contacte donc le contrôleur pour avoir les entrées de flux nécessaires pour diriger la requête vers un serveur choisi (dans cet exemple h1).
3. Quand les autres switches (S3, S2 et S1 dans cet exemple) reçoivent la requête, ils contactent le contrôleur C0 (POX) pour avoir les entrées de flux nécessaires.
4. Ce dernier va leurs ordonner d'adopter le comportement d'un switch d'apprentissage *« learning switch »* (de niveau 2 ou 3) et de diriger la requête vers le port de sortie adéquat. Quand h1 reçoit la requête, il répond au client en suivant le chemin inverse.

5.4 Testes et résultats :

Nous avons fait une étude comparative entre les différents scénarios que nous avons présentés dans la section 5.3, en utilisant le contrôleur SDN POX sans et avec le Load Balancer pour essayer de trouver la meilleure approche en tenant compte des paramètres de performance de base : la surcharge du réseau, le RTT [5], le débit fixer à (20 Mbits/s). Le paramètre RTT, nous permet d'observer, de déduire et de calculer toutes instabilités d'échanges sur cette infrastructure. Il intervient de façon cruciale dans l'efficacité du réseau. L'efficacité augmente lorsque le RTT diminue.

5. RTT : Round Trip Time. Le temps que met un signal pour parcourir l'ensemble d'un circuit fermé.

5.4.1 Comparaison des algorithmes de Load Balancing dans une topologie simple :

Au départ nous avons choisi de faire la comparaison dans une simple topologie (avec un seul switch S1), pour cela le contrôleur POX se comporte de 2 manières différentes :

- Dans un premier cas le POX ordonne le switch de ce comporté comme un switch de niveau 2 (L2_Learning) cela veut dire que S1 va diriger toutes les requêtes du client vers un seul serveur.
- Dans un autre cas nous lancons le controleur POX avec le load balancer, et il ordonne le switch de ce comporter selon la stratégie du LB choisi (Round-Roubin, Random ou Temporal Round-Robin) que nous avons développés comme modules en python sous le nom de (*lb_roundRobin_YS.py, lb_random_YS.py et lb_temporalRR_YS.py*)

Pour faire cette comparaison nous avons mesuré le RTT pour différentes charge du trafic (paquets ICMP [6]) qui sont traitée par les serveurs. Nous avons effectué les mesures selon les deux phases qui suivent, dans le but de vérifier le comportement de SDN avec ou sans le Load Balancer.

- **Phase 1 :** comparaison en utilisant deux serveurs :

Dans cette phase nous avons varié le nombre d'hôtes, le premier cas une topologie avec 10 hôtes, et le deuxième une topologie avec 128 hôtes.

Cas 1 : (10 hôtes) :

	Moyenne de RTT (ms)			
Nombre de paquets	L2_learning	Round-Roubin	Random	Temporal Round-Robin
50	1.567	0.825	0.830	1.250
100	1.424	0.648	0.613	0.706
200	1.268	0.381	0.327	0.427
300	1.031	0.218	0.292	0.340
500	1.242	0.157	0.197	0.250
700	1.222	0.188	0.197	0.276
1000	1.172	0.158	0.144	0.210

TABLE 5.1 – Les mesures du RTT de load balancing avec 10 hôtes et 2 serveurs.

Pour bien analyser les valeurs du tableau 5.1, nous avons tracé le graphe de la figure 5.11 :

6. ICMP : Internet Control Message Protocol.

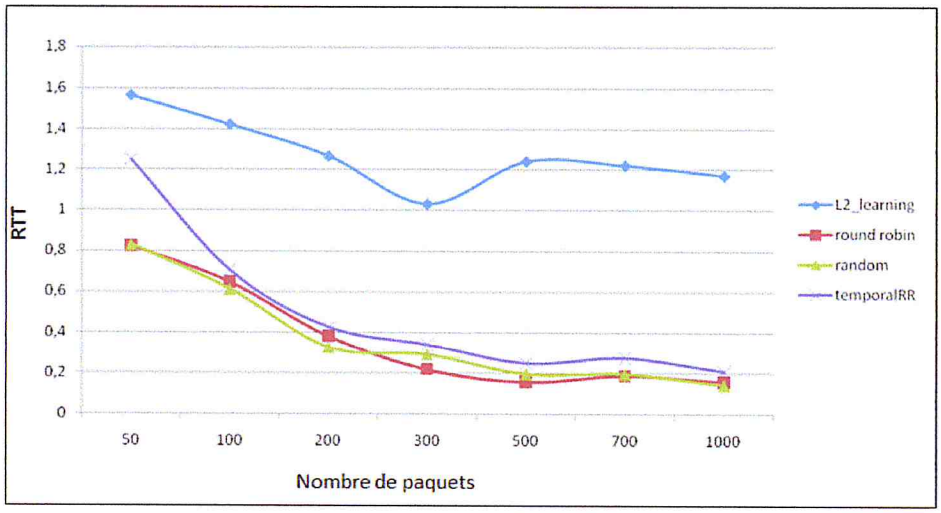


FIGURE 5.11 – comparaison des algotithmes dans une topologie simple à 10 hôtes avec 2 serveurs.

Cas2 : (128 hôtes)

	Moyenne de RTT (ms)			
Nombre de paquets	L2_learning	Round-Roubin	Random	Temporal Round-Robin
50	1.546	0.844	0.952	1.353
100	1.210	0.68	0.721	0.933
200	1.046	0.711	0.767	0.794
300	0.994	0.613	0.645	0.699
500	1.146	0.553	0.607	0.652
700	1.061	0.563	0.624	0.652
1000	1.16	0.613	0.572	0.619

TABLE 5.2 – Les mesures du RTT de load balancing avec 128 hôtes et 2 serveurs.

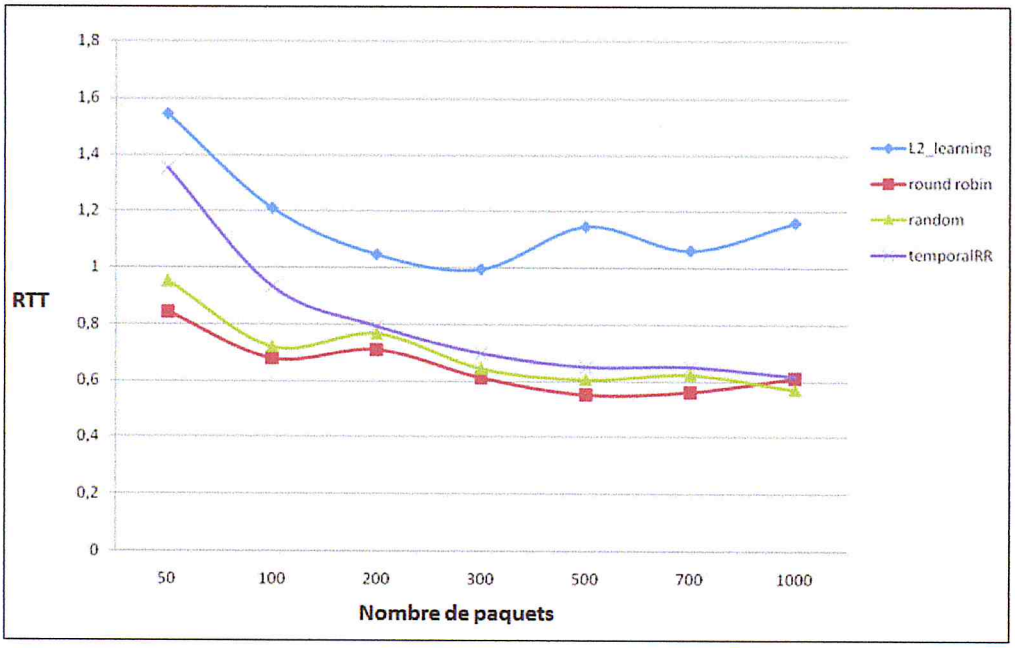


FIGURE 5.12 – Comparaison des algorithmes dans une topologie simple à 128 hôtes avec 2 serveurs.

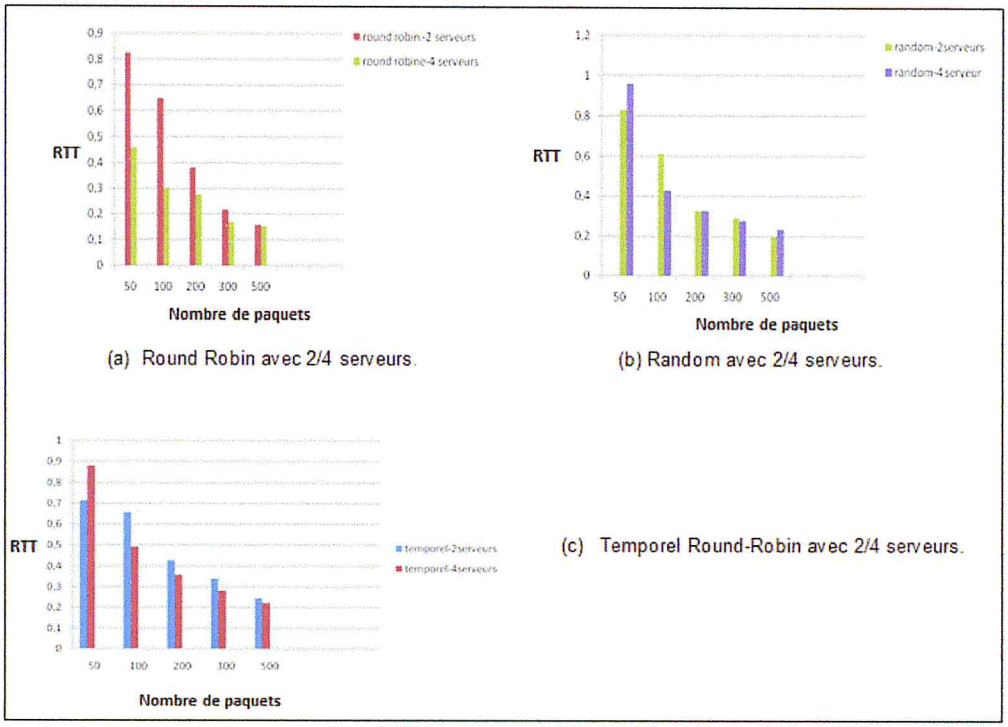
D'après les deux cas (figure 5.11 et 5.12), nous remarquons que le comportement des quatre stratégies passe par 2 phases et le temps de réponse moyen (RTT) du réseau qui adopte l'apprentissage (l2_learning) est élevé et non stable puisque le trafic est dirigé vers un seul serveur qui pourra être saturé à n'importe quel moment, par contre les algorithmes du Load balancing apporte une amélioration remarquable du RTT, cela est dû au partage du trafic entre les deux serveurs (h1, h2).
En ce qui concerne le Temporal Round-Robin, nous avons fixé Δt à 50 secondes. La courbe résultante montre qu'il a amélioré le RTT du réseau mais il reste moins efficace que les deux autres algorithmes de LB. Cela est dù au fait que le basculement d'un serveur à un autre à chaque laps de temps provoque une surcharge en messages de controle, ce qui a influencé négativement sur l'eficacité de cet algorithme.
Le Round-Roubin et le Random ont presque la même efficacité, cela veut dire que dans une simple topologie on peut utiliser les 2 algorithmes. Toutefois, le Random peut solliciter le même serveur plusieurs fois d'affilé, donc en théorie, le Round-Robin est meilleur. Pour confirmer cela, nous avons effectué une deuxième phase de comparaison dans laquelle on a augmenté le nombre de serveurs.

- **Phase 2 :** comparaison en utilisant quatre serveurs :

Dans cette phase, nous avons augmenté le nombre de serveurs à quatre en gardant les même algorithmes de LB et le même nombre de clients pour garder la même charge de trafic que celle de la phase 1. Les mesures effectuées figurent dans le tableau 5.3

Nombre de paquets	Moyenne de RTT (ms)					
	Round-Roubin		Random		Temporal Round-Robin	
	2 serveurs	4 serveurs	2 serveurs	4 serveurs	2 serveurs	4 serveurs
50	0.825	0.457	0.830	0.958	0.714	0.879
100	0.648	0.303	0.613	0.427	0.658	0.493
200	0.381	0.277	0.327	0.328	0.427	0.360
300	0.218	0.167	0.292	0.277	0.340	0.282
500	0.157	0.153	0.197	0.234	0.248	0.224

TABLE 5.3 – Les mesures du RTT de load balancing avec 10 hôtes et 4 serveurs.



(a) Round Robin avec 2/4 serveurs.

(b) Random avec 2/4 serveurs.

(c) Temporel Round-Robin avec 2/4 serveurs.

FIGURE 5.13 – Comparaison des load balancers en variant le nombre de serveurs.

La partie (a) de la figure 5.13 montre que l'efficacité de Round-Robin augmente en parallèle avec le nombre de serveurs puisque le RTT diminue, contrairement au Random et au temporal qui peuvent être parfois meilleur avec deux serveurs qu'avec quatre comme le montre la partie (b) et (c) ,ce qui veut dire qu'il n'exploite pas efficacement les serveurs.

On conclut donc que dans une topologie simple, l'algorithme Round-Robin est le meilleur puisque il partage la charge entre les serveurs de façon circulaire.

5.4.2 Comparaison des algorithmes de Load Balancing dans une topologie à multi-sauts :

les approches que nous avons proposé serons testé sur une topologie à multi-sauts, où chaque switch à un hôte connecté à lui et les switches sont interconnectés de façon linéaire. On a varié le nombre de switches, et en fixant le nombre de serveurs à deux (h1 et h2).
Nous précisons que dans ces approches, nous avons considéré que le Load Balancing doit être effectué par un seul switch, car dans le cas contraire, il aurait fallu trouver un mécanisme.
L'avantage des deux approches que nous avons proposé pour une topologie SDN à multi-sauts est la possibilité de leur intégrer n'importe quel type d'algorithme du loadbalancing.

A. Combinaison du LB avec l'inondation :

Dans le but de tester l'extensibilité du Load Balancer, nous avons développé trois modules en python qui adoptent l'approche 1 (combinaison du loadbalancing avec l'inondation).

- **Étape 1 :** comparaison en utilisant 10 switches

Dans cette étape, nous avons combiné le load balancing (Round-Robin, Random et le TemporalRR) avec le flooding (l'inondation). Les résultats du RTT moyen figurent dans le tableau 5.4 :

	Moyenne de RTT (ms)			
Nombre de paquets	HUB	Round-Roubin + hub	Random + hub	TemporalRR + hub
50	0.372	0.237	0.314	0.263
100	0.324	0.227	0.272	0.274
200	0.315	0.231	0.251	0.259
300	0.312	0.241	0.246	0.266
500	0.318	0.234	0.273	0.259
700	0.255	0.224	0.237	0.233
1000	0.236	0.226	0.224	0.231

TABLE 5.4 – Les mesures du RTT de load balancing avec 10 switches.

Avec les résultats du tableau 5.4, nous avons tracé les courbes dans la figure 5.14 :

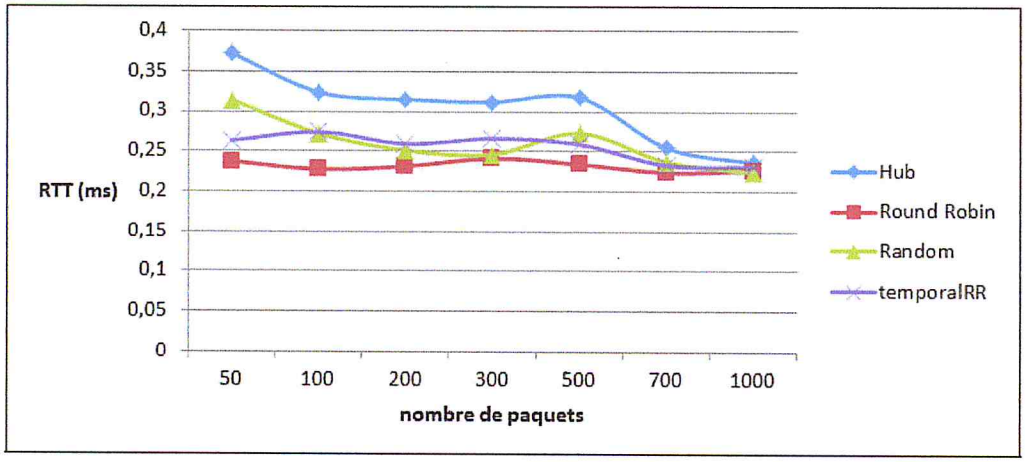


FIGURE 5.14 – Comparaison des algorithmes dans une topologie à 10 switches.

D'après la figure 5.14 , nous observons que les courbes passent par trois phases, la 1er phase(50 à 100 paquets) :le comportement des 4 stratégies est différents, où le RTT du réseau adoptant le hub, le Random ou le Round-Roubin diminue, sauf le temporal round-Roubin qui augmente . Dans la 2eme phase nous remarquons une stabilité pour les 4 courbes , mais en arrivant à 500 paquets où débute la 3eme phase le RTT des 4 stratégies diminu jusqu'à 700 paquets où elles deviennent stables, et le choix d'une stratégies n'influe pas sur le RTT puisque la différence entre eux est négligeable.
Donc, nous remarquons que même dans une topologie à multi-sauts, le RTT du réseau adoptant le flooding (hub) est élevé par rapport au load balancing, mais les courbes du round-robin, random et temporalRR ont presque la même efficacité sauf que le round-robin passe mieux à l'échelle que les autres, car même si nous augmentons la charge du réseau le RTT moyen du round-robin reste toujours le meilleur.

- **Etape 2 :** comparaison en utilisant 15 switches

Pour confirmer les tests, nous avons augmenté le nombre de switches à 15 pour analyser le comportement du load balancer, Les résultats du RTT moyen figurent dans le tableau 5.5 :

	Moyenne de RTT (ms)			
Nombre de paquets	HUB	Round-Roubin + hub	Random + hub	TemporalRR + hub
50	0.35	0.315	0.322	0.327
100	0.404	0.307	0.354	0.343
200	0.39	0.316	0.347	0.334
300	0.44	0.342	0.351	0.357
500	0.416	0.335	0.336	0.34
700	0.343	0.306	0.317	0.321
1000	0.31	0.31	0.322	0.323

TABLE 5.5 – Les mesures du RTT de load balancing avec 15 switches.

Avec les résultats du tableau 5.5, nous avons tracé les courbes dans la figure 5.15 :

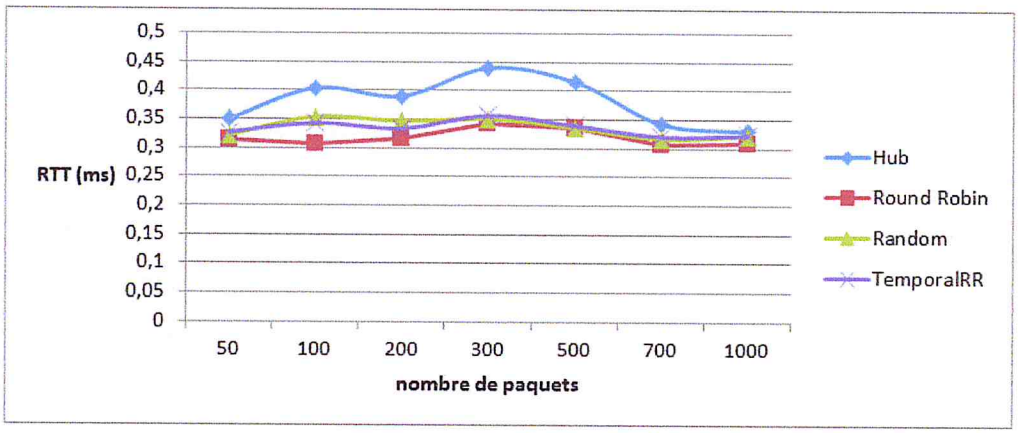


FIGURE 5.15 – Comparaison des algorithmes du load balancing dans une topologie à 15 switches.

D'après la figure 5.15, nous observons que le graphe passe par 3 phases, la 1er phase (50 à 200) : la mesure RTT pour les 4 stratégies est stables et les algorithmes du LB passent mieux à l'échelle que le hub. Par contre, dans la 2éme phase (200 à 700 paquets) nous remarquons un grand changement puisque le RTT du réseau adoptant le hub augmente beaucoup par rapport aux 3 stratégies du LB. Mais en dépassant 700 paquets (3éme phase) tout redevient stable, donc le choix n'influe pas sur le RTT.
Nous remarquons, qu'on appliquant le flooding (Hub), le RTT du réseau est élevé par rapport au algorithmes du load balancing, par contre, ces derniers améliorent le taux du RTT, malgré qu'ils ont presque la même efficacité, le round Robin reste meilleur que le Random et le temporal Round-Roubin.

- **Etape 3 :** comparaison en utilisant 30 switches

Nous avons augmenté encore plus le nombre de switches à 30 pour analyser le comportement du loadbalancer, Les résultats du RTT moyen figurent dans le tableau5.6 :

	Moyenne de RTT (ms)			
Nombre de paquets	HUB	Round-Roubin + hub	Random + hub	TemporalRR + hub
50	0.753	0.573	0.566	0.567
100	0.745	0.566	0.579	0.547
200	0.722	0.316	0.565	0.563
300	0.716	0.564	0.553	0.549
500	0.716	0.583	0.564	0.559
700	0.674	0.57	0.56	0.564
1000	0.601	0.556	0.534	0.527

TABLE 5.6 – Les mesures du RTT de load balancing avec 30 switches.

Avec les résultats du tableau 5.6, nous avons tracé les courbes dans la figure 5.16 :

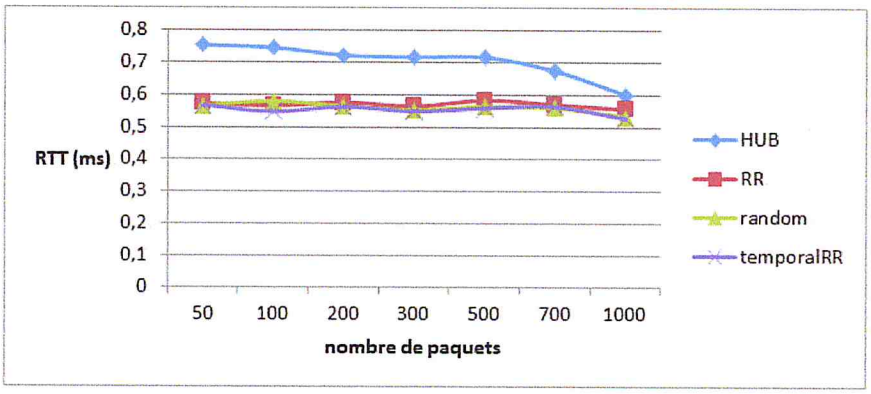


FIGURE 5.16 – Comparaison des algorithmes dans une topologie à 30 switches.

D'aprés la figure 5.16, nous remarquons toujours qu'ils ont presque la même efficacité mais cette fois ci le temporalRR passe mieux à l'échelle que les deux autres. Les courbes présentées viennent confirmer nos résultats qui montrent que l'adoption du load balancing diminue le RTT du réseau et passe mieux à l'échelle par rapport au flooding.

Pour conclure la partie combinaison avec l'inondation, et d'aprés la majorité des tests, le Round Robin est l'un des meilleurs algorithmes statiques du load balancing cela est dû au fait qu'il partage équitablement la charge entre les serveurs.

B. combinaison avec l'apprentissage :

Dans le but de tester l'extensibilité du Load Balancer, nous avons développé trois modules en python qui adoptent l'approche 2 (combinaison du load balancing avec l'apprentissage).

- **Phase 1 : comparaison en utilisant 10 switches**

Dans cette étape, nous avons combiné le loadbalancing (Round-Robin, Random et le TemporalRR) avec l'apprentissage (learning). Les résultats du RTT moyen figurent dans le tableau 5.7 :

	Moyenne de RTT (ms)			
Nombre de paquets	l2_learning	Round-Roubin + l2_learning	Random + l2_learning	TemporalRR + l2_learning
50	4.937	0.241	0.256	0.252
100	8.531	0.233	0.245	0.241
200	9.592	0.240	0.250	0.244
300	8.364	0.239	0.238	0.236
500	8.872	0.242	0.255	0.263
700	7.433	0.237	0.241	0.282
1000	8.09	0.245	0.244	0.273

TABLE 5.7 – Les mesures du RTT de load balancing avec 10 switches.

La figure 5.17 représente les courbes des trois algorithmes du load balancing en utilisant le tableau 5.7, pour faire une comparaison entre eux sans prendre en charge le l2_learning car les mesure RTT du du réseau adoptant l'apprentissage est trop élevé par rapport au LB.

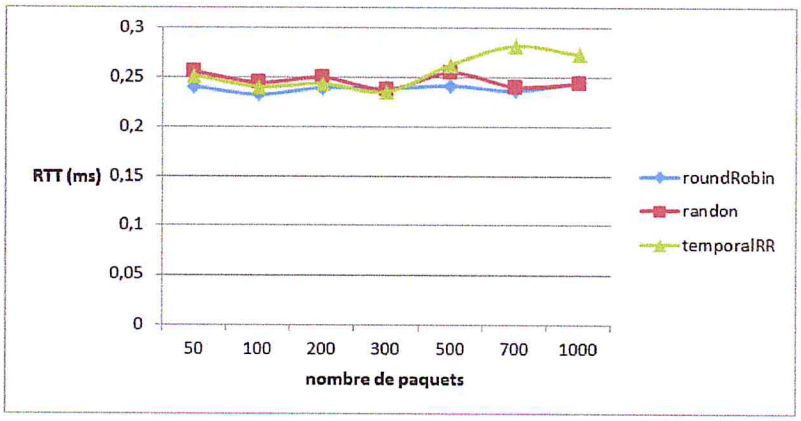


FIGURE 5.17 – Comparaison des algorithmes dans une topologie à 10 switches.

D'après la figure 5.17, nous observons que les courbes du load balancing passent par 2 phases ,où la 1ére phase indique une très bonne stabilité des 3 stratégies, mais à partie de 300 paquets, où débute la 2éme phase le temporal Round-Robin change de comportement et augmente la mesure RTT contrairement au Round-Robin et Random qui restent stables, et ont presque la même efficacité. Nous constatant aussi que le round-robin est stable et passe mieux à l'échelle que les autres algorithmes.

- **Phase 2 :** comparaison en utilisant 15 switches

Nous avons augmenté le nombre de switches à 15 pour analyser le comportement du load balancer, Les résultats du RTT moyen figurent dans le tableau 5.8 :

	Moyenne de RTT (ms)			
Nombre de paquets	l2_learning	Round-Roubin + l2_learning	Random + l2_learning	TemporalRR + l2_learning
50	9.666	0.283	0.289	0.289
100	13.961	0.260	0.297	0.293
200	15.267	0.261	0.288	0.287
300	15.133	0.278	0.264	0.294
500	17.889	0.270	0.265	0.311
700	18.033	0.267	0.264	0.305
1000	15.96	0.267	0.282	0.322

TABLE 5.8 – Les mesures du RTT de load balancing avec 15 switches.

La figure 5.18 montre les courbes des trois algorithmes du LB en utilisant le tableau 5.8 pour faire une comparaison entre eux sans prendre en considération le l2_learning car les mesure RTT du réseau adoptant l'apprentissage est trop élevé par rapport au LB.

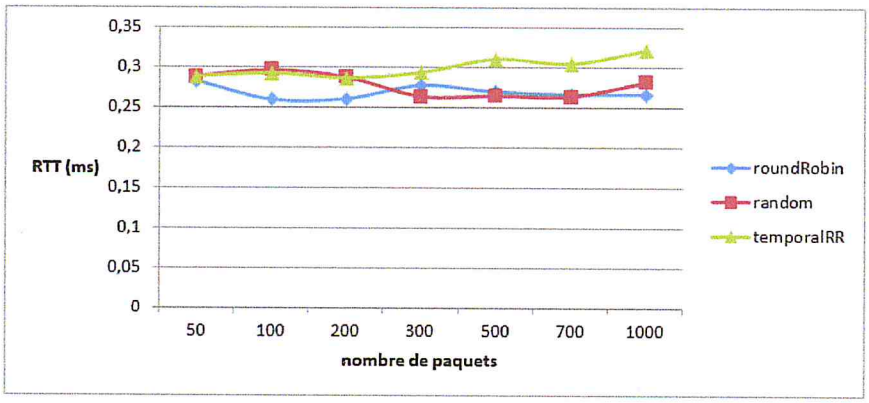


FIGURE 5.18 – Comparaison des algorithmes dans une topologie à 15 switches.

D'après la figure 5.18, nous observons que le comportement des courbes passe par 3 phases. La 1ére et la 3éme phase qui sont respectivement (50 à 200 paquets) et (300 à 1000 paquets) montrent une stabilité et une éfficacité du Round-Robin par rapport au temporal Round-Robin et Random. Par contre la 2éme phase (200 à 300 paquets) présente une instabilité car les 3 algorithmes changent de comportement. D'après les 3 phases nous constatons que le Round-Roubin est toujours meilleurs que les 2 autres algorithmes.

Au final, nos résultats annoncent que le LB apporte un grand plus pour le réseau puisqu'il réduit le temps de réponse (vu d'après le RTT) en équilibrant la charge sur nos serveurs et cela permet une meilleurs utilisation des ressources, et d'après ces comparaison nous pouvons dire que le round-robin est l'un des meilleurs algorithmes statique.

Remarque :

Avec le système que nous avons utilisé, l'émulateur mininet est limité à 128 hôtes. Dans le cas d'une topologie linéaire, si nous dépassons 30 hôtes on obtient des résultats inexploitables (les mesures RTT) comme le montre la figure 5.19.

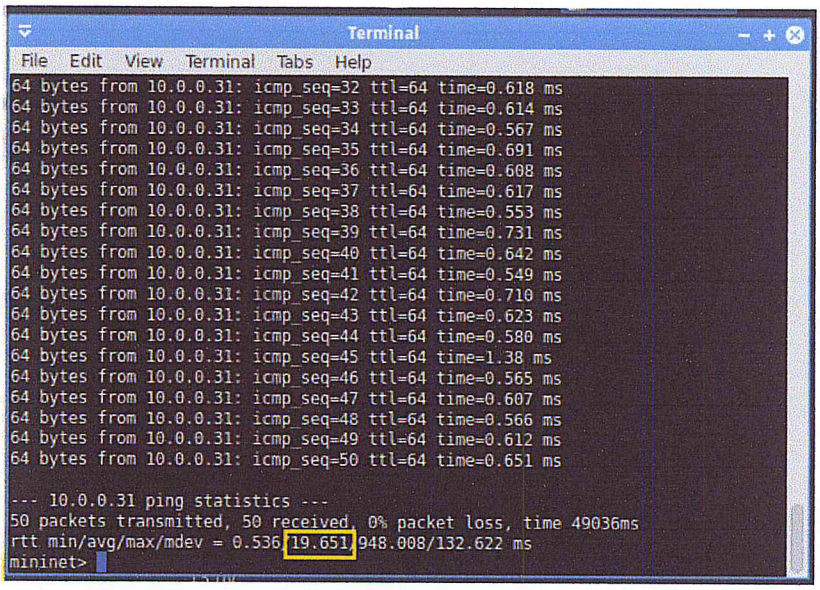


FIGURE 5.19 – Mesure RTT dans une topologie linéaire de 31 switches.

5.5 Conclusion :

Dans ce dernier chapitre nous avons fait une étude et une évaluation sur l'approche SDN combinée à NFV et qui est basé sue le protocole OpenFlow, ainsi que les stratégies de Load Balancing que nous avons proposé et développé pour voir leur influence sur le comportement de cette approche.

Dans une topologie SDN simple, l'efficacité du Load Balancing est assurée par rapport à l'apprentissage (sans le load Balancer) quel que soit le charge du réseau à condition que les serveurs ne soient pas saturés. Par contre l'efficacité peut changer selon l'algorithme choisi et sa complexité ainsi que le nombre de serveurs existant.

Pour une topologie linéaire (multi-saut), nous avons appliqué deux nouvelles approches que nous avons proposées pour mieux voir l'efficacité des techniques du load balancing (combinaison du load balancing avec l'apprentissage et avec l'inondation), les résultats montre que les deux approche améliore les mesures RTT même en augmentant la charge du trafic.

D'après l'ensemble des tests, nous remarquant que le round-robin passe mieux à l'échelle que les autres algorithmes quelque soit la charge du trafic et la taille du réseau. Enfin, nous pouvons affirmer que le SDN fonctionne mieux avec un load balancer.

Conclusion générale

La technologie SDN basé sur le protocole OpenFlow et NFV devrait révolutionner à terme les architectures réseau et permettre de déployer de nouveaux services de manière beaucoup plus rapide et avec des couts significativement réduits.

Dans ce mémoire nous avons implémenté cet environnement afin de développer des techniques d'équilibrage de charge en utilisant des stratégies existantes de load balancing (Round-Roubin, Random, Temporal Round-Robin) qui fonctionnent sur des topologies simple à un saut, ainsi que deux nouvelles approches de load balancing que nous avons proposé pour les topologies SDN à multi-sauts.

Dans une topologie multi-sauts, nous avons développé six modules, trois d'entre eux adoptent l'approche 1 qui est la combinaison du Load Balancing avec l'inondation (flooding), d'après les tests, nous avons constaté qu'il consomme beaucoup de bande passante. Donc nous avons proposé l'approche 2 qui est la combinaison du Load Balancing avec l'apprentissage (le learning) en développant 3 autres modules qui l'adoptent. De cette approche deux méthodes peuvent en découler. La première est la combinaison de LB avec le learning de niveau 2 (LB + l2 learning) ; c'est-à-dire que l'apprentissage se base sur les adresses MAC seulement. La deuxième est la combinaison de LB avec le learning de niveau 3 (LB + l3 learning) ; c'est-à-dire que cette fois-ci l'apprentissage se base sur les adresses MAC et IP. Dans notre travail, le choix s'est porté sur la méthode qui utilise le L2_learning. L'avantage des deux approches que nous avons proposé est qu'ils peuvent intégrer n'importe quelle stratégie de load balancing. Cependant, nous constatons qu'on dépassant 700 paquets, Random passe mieux à l'échelle que le Round-Robin malgré qu'ils ont presque la même efficacité.

Nos perspectives sont de développer un algorithme qui utilise les 2 stratégies (Round-Robin, Random) de tel sort que si le nombre de paquets est moins de 700 paquets, l'algo-

rithme adopte la stratégie du Round-Roubin, et dans le cas contraire il adopte la stratégie du Random, afin d'améliorer l'efficacité et le temps des réponses pour avoir un réseau plus performant.

Annexes

Annexe 1 :

Les étapes d'installation de l'émulateur Mininet et ces périphériques :

1. Récupéré le code source :

Pour cela il faut ouvrir un terminal au niveau d'Ubuntu et taper la commande :

```
git clone git://github.com/mininet/mininet
```

La commande git permet d'avoir la meilleur version de mininet.

2. Installer Mininet :

Une fois le code source est prêt on exécute la commande pour installer mininet :

```
mininet/util/install.sh [options]
```

« Install. sh» inclus de différents paramètres selon le besoin de l'utilisateur :

- *-a* : installer tout ce qui est inclus dans MININET y compris OpenvSwitch le protocole OpenFlow, wireshark et le POX. Par défaut, ces outils seront construits dans des répertoires créés dans votre répertoire personnel.

- *-nfv* : installer MININET, le commutateur de référence OpenFlow et Open vSwitch.

- *-s mydir* : utiliser cette option avant d'autres options pour placer la source / construire des arbres dans un répertoire spécifié plutôt que dans votre répertoire personnel.

```
To install everything (using your home directory): install.sh -a
To install everything (using another directory for build): install.sh -s mydir -a
To install Mininet + user switch + OVS (using your home dir): install.sh -nfv
To install Mininet + user switch + OVS (using another dir:) install.sh -s mydir -nfv
```

Nous avons utilisé l'option *−a* puisque elle répond à nos besoins.

3. Tester Mininet :

Une fois l'installation est terminée, nous avons testé la fonctionnalité de mininet avec cette simple commande :

```
sudo mn --test pingall
```

S'il n'y a pas d'erreur mininet est prêt a l'utilisation .

Annexe 2 :

Dans un algorithme d'équilibrage de charge le choix d'un serveur se fait selon la stratégie choisi. Les points suivant montrent la partie du code ou ce fait la sélection du serveur.

1. Random :

```
def get_next_server (self):
  global last_server #
  # Random load the servers
  last_server = randrange(len(self.servers))% len(self.servers)
  return self.servers[last_server]
```

2. Round-Robin :

```
def get_next_server (self):
  global last_server #
  # Round-robin load the servers
  last_server = (last_server + 1) % len(self.servers)
  return self.servers[last_server]
```

3. Temporal Round-Robin :

le choix se fait de la méme manière que Round-Robine juste en fixant un Δt à 50 seconde.

```
HARD_TIMEOUT = 50
```

Bibliographie

[1] kherbache Vincent, Mousalih Mouhamed, Kuhn Yannick, and Lefort Allan. cloud computing. *IUT nancy charlemagne*, 2009/2010.

[2] Figer Jean-Paul. L'informatique en nuage [cloud computing]. http ://www.figer.com/Publications/nuage.htm#.VycROYThDIU, 25 fevrier 2012. consulté le : 12 décembre 2015.

[3] Imen Laribi. Etude et mise en place d'une solution cloudcomputing privée pour une entreprise. *Universite Abou Bekr Belkaid Tlemcen UABT*, 26 mai 2014.

[4] Blogger Cloud Computing. Comprendre le cloud computing 2 : Cinq caractéristiques essentielles du cloud computing. http ://cloudcomputinginfrench.blogspot.com/2013/01/comprendre-le-cloud-computing-2-cinq.html, 21 janvier 2013. consulté le 27 décembre 2015.

[5] Blogger Cloud Computing. Comprendre le cloud computing 4 : Modèles de déploiement. http ://cloudcomputinginfrench.blogspot.com/2013/01/comprendre-le-cloud-computing-4- modeles.html, 21 janvier 2013. consulté le : 27 décembre 2015.

[6] Figer Jean-Paul. L'informatique en nuage [cloud computing]. http ://www.figer.com/Publications/nuage.htm#550, 25 fevrier 2012. consulté le : 29 décembre 2015.

[7] Benkemoun Antoine and Hinfray Romain. Ac-etude de la virtualisation et du fonctionnement de la solution libre xen. *Université de technologie Troyes*, Semestre Automne 2008.

[8] VMware. Virtualisation. https ://www.vmware.com/fr/virtualization/overview, 2014. consulté le : 30 décembre 2015.

[9] Renaud Venet. Cloud computing : avantages et inconvénients. http ://www.renaudvenet.com/cloud-computing-avantages-et-inconvenients-2011-01-26.html, 26 janvier 2011. consulté le : 29 décembre 2015.

[10] Damien HUBAUX. Cloud computing. *Centre d'Excellence en Technologies de l'Information et de la communication*, 2013.

[11] ComputerLand. Au début était l'onf. http ://www.computerland.fr/cloud-computing/office-365-cloud-microsoft/. consulté le : 3 mai 2016.

[12] Roux Philippe. Au début était l'onf. http ://www.cloud-experience.fr/le-software-defined-networking-une-revolution-a-disposition-du-cloud/, 24 Février 2013. consulté le : 5 janvier 2016.

[13] citrix.fr. Sdn 101 : An introduction to software-defined networking. https ://www.citrix.fr/products/netscaler-adc/resources/sdn-101.html, 26 janvier 2014. consulté le : 28 novembre 2015.

[14] Michael Zimmerman. Openflow-enabled sdn and network functions virtualization. *Open Networking Foundation*, 17 Février 2014.

[15] Openstack, opendaylight et openflow. http ://meetageek.net/blog/ ?p=56, mars 2015. consulté le : 28 novembre 2015.

[16] Bayshore Road E and Alto Palo. Software-defined networking :the new norm for networks. *Open Networking Foundation*, 13 Avril 2012.

[17] HP networking. Openflow : une technologie habilitante pour le software-defined networking. http ://h17007.www1.hp.com/fr/fr/solutions/technology/openflow/index.aspx. consulté le : 24 février 2016.

[18] Seddiki Mohamed Said. Allocation dynamique des ressources et gestion de la qualite de service dans la virtualisation des reseaux. *Réseaux et télécommunications Université de Lorraine*, 14 décembre 2015.

[19] Bruyere Marc and Delavennat David. Utilisation d'openflow et des modules splite data plane de dell pour traiter le duid-mac-spoofing des requetes dhcpv6. *Les Journées Réseaux 2013-JRES*, décembre 2013. Montpellier, France.

[20] Wolfgang Braun and Michael Menth. Software-dened networking using openflow : Protocols, applications and architectural design choices. *future internet*, 12 mai 2014.

[21] GUIRLINGER Julien. Sdn et openflow. https ://www.randco.fr/actualites/2014/sdn-et-openflow/, 26 juin 2014. consulté le : 24 février 2016.

[22] Jeannin Xavier and Loui Frédéric. Sdn/nfv, une approche hybride. *JRES, Montpellier*, 2015.

[23] TECHNIQUES DE L'INGÉNIEUR. Etude et avenir de la sécurité des solutions de virtualisation. http ://www.techniques-ingenieur.fr, octobre 2014. Ref : H6035.

[24] Brocade Communications Systems. Votre centre de données est-il prêt pour le sdn ? *IT CORPORATE information HUB*, page 13, 10 septembre 2014.

[25] Jain Raj. Openflow, software defined networking (sdn) and network function virtualization (nfv). *Washington University in Saint Louis*, 1 juillet 2014.

[26] Moelands Marcel. Nfv, la lumière au bout du tunnel pour les opérateurs télécoms ? http ://www.journaldunet.com/ebusiness/expert/56636/nfv–la-lumiere-au-bout-du-tunnel-pour-les-operateurs-telecoms.shtml, 24 Février 2014. consulté le : 7 mars 2016.

[27] Pate Prayson. Nfv and sdn : What's the difference ? https ://www.sdxcentral.com/articles/contributed/nfv-and-sdn-whats-the-difference/2013/03/, 30 mars 2013. consulté le : 7 mars 2016.

[28] Jain Raj. Introduction to software defined networking (sdn). *Washington University in Saint Louis*, 2013.

[29] Gaspard Gatien, Jachniewicz Rémi, Lacava Julien, and Meslard Vincent. Équilibrage de charge et haute disponibilité pour applications web ruby on rails. *LP ASRALL Licence Professionnelle Administration de Systèmes, Réseaux et Applications à base de Logiciels Libres*, 22 avril 2009.

[30] MOHAMED MASSAOUDI, Hachaichi Chahinaz, Dhawefi Amani, Maijed Erij, and Boughanmi Emna. Load balancing. *SECURILIGHT*, 2013.

[31] Jain Kansal Nidhi and Chana Inderveer. loud load balancing techniques : A step towards green computing. *International Journal of Distributed and Cloud Computing*, janvier 2012. Vol 9 , Issue 1 ISSN (Online) : 1694-0814 ;www.IJCSI.org.

[32] Katyal Mayanka and Mishra Atul. A comparative study of load balancing algorithms in cloud computing environment. *International Journal of Distributed and Cloud Computing*, Décembre 2013. Volume 1, Issue 2.

[33] Garit Yann, Gilliot Yann, Lacroix Steve, Lamandé Dorian, Panczak Maxime, and Vroomhout Aymeric. Load-balancing avec linux virtual server. *Projet CSIII*, 2008.

[34] Kaur Sukhvir and Kinger Supriya. Review on load balancing techniques in cloud computing environment. *International Journal of Science and Research (IJSR)*, juin 2014. Volume 3 Issue 6, ISSN (Online) : 2319-7064.

[35] IFAKREN Ouafae. Software defined network. *Travail de semestre Encadré par : Gérald Litzistorf professeur HES*, juin 2014.

[36] Université de Marne-la Vallée. Software-defined networking. http ://www-igm.univ-mlv.fr/ dr/XPOSE2014/software-defined-networking/openflow.html. consulté le : 2 avril 2016.

[37] Medeiros Bruno, Antonio Simplicio Marcos, Cristina Melo de Brito Carvalho Tereza, and Antonio Torrez Rojas and Marco. Chapitre 1 : Applying software-defined networks to cloud computing.

[38] Le projet OpenFlow. Openflow l'évolution des réseaux. https ://wapiti.telecom-lille.fr/commun/ens/peda/options/st/rio/pub/exposes/exposesrio2009-ttnfa2010/cloarec-ferreira/controller.html, 2010. consulté le : 26 mars 2016.

[39] IDOUDI KARIM. ImplÉmentation d'un plan de contrÔle unifiÉ pour les rÉseaux multicouches ip /dwdm. *UNIVERSITÉ DU QUÉBEC À MONTRÉAL*, mai 2014.

[40] Dominik Klein and Michael Jarschel. An openflow extension for the omnet++ inet framework. *Cannes, France*, mars 2013.

[41] Smile Open source solutions. RÉpartition de charge de niveau tcp. http ://infrastructure.smile.eu/Tout-savoir-sur/Principes-d-architecture-et-outils-open-source/Repartition-de-charge/Repartition-de-charge-de-niveau-tcp. consulté le : 20 janvier 2016.

[42] SA LALTEN. Technologies sdn et nfv : Pourquoi s'y intÉresser ? http ://www.alten.fr/reseaux/technologie-sdn-pourquoi-sy-interesser, 19 Novembre 2013. consulté le : 3 mai 2016.

[43] Beckmann Curt. RÉseaux : Comprendre le sdn et le nfv. http ://www.solutions-numeriques.com/reseaux-comprendre-le-sdn-et-le-nfv/, 23 Février 2016. consulté le : 24 avril 2016.

[44] Narimane BENHELLAL and Fares ZAIDI. Enabling nfv with openflow-enabled software defined networking (sdn) and load balancing techniques. *Université des Sciences et de la Technologie Houari Boumediene*, 2015.

[45] Matt. Software-defined networking. http ://www.6wind.com, aout 2013. consulté le : 24 avril 2016.

[46] Load balancers. http ://www.netbenefit.fr/infogerance/load-balancing/qu-est-ce-que-le-load-balancing, Mars 2015. consulté le : 30 avril 2016.

[47] Introduction to mininet. https ://github.com/mininet/mininet/wiki/Introduction-to-Mininet, 17 décembre 2015. consulté le : 4 mai 2016.

[48] Modon Sytoka. Open vswitch, le commutateur virtuel bientôt sur votre serveur. http ://linuxfr.org/news/open-vswitch-le-commutateur-virtuel-bient21 mai 2010. consulté le : 4 mai 2016.

[49] Kaur Karamjeet, Singh Japinder, and Singh Ghumman Navtej. Mininet as software defined networking testing platform. *International Conference on Communication, Computing and Systems*, 2014.

[50] Using the pox sdn controller. http ://www.brianlinkletter.com/using-the-pox-sdn-controller/, 20 Avril 2015. consulté le : 4 mai 2016.

[51] Milev Stéfan, Mathieu Anthony, Courtiol Thibault, and Nguyen Joël. Sdn avec grid'5000. *LP ASRALL , Université de LORRAINE*, 12 Fevrier 2016. consulté le : 4 mai 2016.