Républic Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid - Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du Diplôme de Master en Informatique

Option: *Réseaux et Systèmes Distribués (R.S.D)*

Thème :

Etude de performance d'une solution SDN pour la garantie de QoS

Réalisé par :

- ✓ **TOUMI FATIMA ZOHRA**
- ✓ **MOKEDDEM ASSIA**

Présenté le 24 Septembre 2020 devant le jury de :

- Labraoui Nabila (Président)
- Benbrik Ilyas (Encadreur)
- Maouda Tarek (Encadreur)
- Belhocine Amine (Examinateur)

Année universitaire : 2019-2020

Dédicace

À mes anges, mes très chers parents dont l'existence
me tient en vie.
À mes adorables frères et sœurs, Alilo, Amine, Sihem et Abir.

À toute ma famille maternelle et paternelle qui m'ont soutenu durant les temps durs de ce mémoire et plus particulièrement ma tante et seconde maman « Malika ».

Sans jamais oser oublier ceux sans qui mes années universitaires n'auraient pas été aussi fabuleuses, mes gracieuses adorables amies : Meriem, Amitaf, Bouchra, Ibtissem, Fatiha, Amira.

Je vous aime.

Fatima

Dédicace

*I dedicate this project **to** my beloved mom 'Ouahiba Zerouati', my Allah bless your soul in Firdaous el Aala, I never thought you'd leave my side.*

I dedicate this in memory of your kindness and generosity towards me.

***To** my father who works hard for my success, thank you for your trust, your support, all the sacrifices you've made.*

***To** my dear brother Taha , my dear sister Meriem.*

*And last but not least, **to** the nicest and best girlfriend, who was patient with me and supported me in this tough period, God bless you and give you all the good that you deserve Fatima.*

All family: Mokeddem, Zerouati, Ben adjmeia.

Assia

Remerciement

Je remercie tout d'abord Dieu le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce modeste travail.

Nous voudrons tout d'abord adresser toute notre gratitude à l'encadreur de ce mémoire, docteur Ilyas Bembrik, pour avoir accepté d'encadrer ce travail, sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter notre réflexion.

Nous tenons à remercier également l'encadreur M.Tarek Maouda pour avoir proposé ce thème.

Nous adressons aussi nos vifs remerciements aux honorables membres de Jury à qui nous ont fait l'honneur de juger ce travail. Les critiques et suggestions que vous apportez seront les bienvenues et contribueront à l'amélioration le produit final.

Nous tenons à saisir cette occasion et adresser nos profonds remerciements et reconnaissances aux responsables et aux enseignants de la faculté de sciences, département d'informatique.

À monsieur le professeur Badr Benmammar, nous tenions à vous dire que vous avez été un excellent professeur, nous vous remercions d'avoir partagé vos connaissances avec nous, d'avoir toujours été juste dans votre enseignement et de nous avoir toujours soutenus.

Nos derniers mots s'adressent à la lumière de nos yeux, nos parents, toutes nos familles et nos amies du fond du cœur pour l'aide et l'amour qu'ils nous ont apportés.

Abréviations

RIB : Routing Information Base

SDN : Software Defined Network

API : Application Programming Interface

SBI : Sounthbound Interface

NBI : Northbound Interface

Diffserv : Differenciated Service

DSCP : Differentiated Services Code Point

IETF : Internet Engineering Task Force

RSVP : Resource Reservation Protocol

IntServ : Integrated Service

LAN : Local Area Network

NOS : Network Operating System

ODL : OpenDaylight

OF : OpenFlow

ONF : Open Networking Foundation

OVSDB : Open VSwitch DataBase

QOS : Quality Of Service

OVS : OpenvSwitch

TLS : Transport Layer Security

Sommaire

INTRODUCTION GENERALE **1**

CHAPITRE I : PRESENTATION DE LA TECHNOLOGIE SDN **3**

I.1 Introduction **10**

I.2 Présentation de l'organisme d'accueil **10**

I.3 Problématique **14**

I.4 Introduction aux réseaux SDN **15**

I.5 Fonctionnement de base **16**

I.6 Architecture SDN : **18**

I.7 Les interfaces SDN **19**

I.8 Activités autour de SDN / OpenFlow : **22**

I.9 Protocole OpenFlow : **23**

I.10 Contrôleur : **30**

I.11 Défis SDN : **32**

I.12 Conclusion : **33**

CHAPITRE II : LA QOS DANS LE RESEAU SDN **30**

II.1 Introduction : **31**

II.2 Définition de QoS : **31**

II.3 Paramètres de QoS: **31**

II.4 Modèles de QoS : **32**

II.5 Mécanismes de la QoS : **34**

II.6 L'état de l'art de la QoS dans les SDN **36**

II.7 Conclusion **45**

CHAPITRE III : ETUDE DE PERFORMANCES DE LA GESTION QOS DANS UN RESEAU SDN **46**

III.1 Introduction **47**

III.2 Outils logiciels 47

III.3 Expérimentations : 50

III.4 Conclusion 66

CONCLUSION GENERALE 67

ANNEXE 68

Liste de figures

Figure I- 1 : Organigramme d'Algérie Télécom. 12

Figure I- 2 : L'emplacement géographique d'Algérie Télécom 13

Figure I- 3 : Comparaison entre réseaux traditionnels et SDN (14). 17

Figure I- 4 : L'architecture du réseau SDN (10). 19

Figure I- 5 : Les interfaces de SDN (9). 20

Figure I- 6 : Le protocole Openflow (20). 23

Figure I- 7 : Table de flux (3). 25

Figure I- 8 : Le processus de transmission d'un paquet avec Openflow (9). 26

Figure I- 9 : Interface OpenFlow et protocole de messagerie (3). 27

Figure II- 1 : Le protocole RSVP (45). 33

Figure II- 2 : Le protocol DiffServ (48). 34

Figure II- 3 : Traffic Shaping VS Traffic Policing (50). 36

Figure III- 1 : Emulateur Mininet (66) 48

Figure III- 2 : Architecture de D-ITG (69) 49

Figure III- 3 : Code source de la topologie de test 51

Figure III- 4 : Diagramme de topologie 52

Figure III- 5 : Démarche de découverte de la route optimale avec prise en charge de la QoS 56

Figure III- 6 : Le résultat de l'exécution de la commande pour lancé le composant l2_multi 57

Figure III- 7 : Le résultat de l'exécution de la commande pour lancé la topologie dans Mininet 58

Figure III- 8 : Le résultat dans le POX après le lancement de la topologie dans Mininet 58

Figure III- 9 : Commande ITGSend dans h1 59

Figure III- 10 : Commande ITGSend dans h4 59

Figure III- 11 : Décodage de fichier recv_log_file2 sur la machine h2 61

Figure III- 12 : Décodage de fichier recv_log_file3 sur la machine h3 61

Figure III- 13 : Le résultat de l'exécution de la commande pour lancé QoSOptRouting 62

Figure III- 14 : Graphes de débit avec et sans gestion de QoS 62

Figure III- 15 : Graphes de débit avec et sans gestion de QoS 63

Figure III- 16 : Graphes de délai avec et sans QoS par rapport au trafic ordinaire 64

Figure III- 17 : Graphes de délai avec et sans QoS par rapport au trafic intense 64

Figure III- 18 : Variation de délai dans le trafic ordinaire 100 – 280 paquets par seconde 65

Figure III- 19 : Variation de délai dans le trafic intense 500 – 5000 paquets par seconde 66

Liste des tableaux

Tableau I- 1 : L’évolution du protocole OpenFlow (16). 23

Tableau I- 2 : Comparaison entre les propriétés des différents contrôleurs SDN. 31

Tableau II- 1 : Tableau comparatif entre les solutions citées. 44

Tableau III- 1 : Switches et hôtes 53

Introduction Générale

La qualité de service (QoS) est la capacité d'un périphérique réseau d'avoir un certain niveau d'assurance afin de satisfaire les exigences des trafics utilisateurs. La QoS est particulièrement importante pour les applications et les services avec des exigences strictes telles que la voix sur IP, la vidéoconférence et le streaming, les jeux en ligne et le commerce électronique. Cependant, le contrôle de la QoS n'a jamais été une tâche facile. En effet, pour pouvoir transmettre les flux multimédias dans ces types d'applications, il est très important de mettre en œuvre des politiques de QoS prenant en charge leurs besoins en termes de bande passante, de délai, de gigue, du taux de pertes de paquets afin de garantir leur réception avec un niveau de qualité acceptable.

Dans un réseau traditionnel, les opérateurs configurent chaque périphérique individuellement en utilisant l'interface de ligne de commande, mais cette option peut être limitée aux fonctionnalités préinstallées. En outre, les grands réseaux peuvent contenir des milliers de composants qui doivent être configurer ou à reconfigurer uniformément pour mettre en œuvre de nouvelles politiques de routage avec QoS. D'ailleurs, les méthodes actuelles de configuration de la QoS prennent beaucoup de temps et ne contiennent pas de mécanisme d'intelligence centralisée pour configurer la QoS en fonction de l'évolution des conditions du réseau.

Dans ce contexte, pour répondre aux limites et difficultés issues par la prise en compte de mécanisme de QoS dans les réseaux traditionnels, l'objectif principal de Software Defined Networks (SDN) et de la technologie OpenFlow (1) est de séparer le matériel de la couche logicielle de contrôle permettant aux opérateurs de réseau de réduire les frais opérationnels du réseaux. Cela implique que l'automatisation de la gestion du réseau en tant que trafic de données peut être manipulée, ajusté et ajustée indépendamment des protocoles de routage grâce à la programmabilité du matériel. Ainsi, avec cette facilité de reconfiguration, plusieurs travaux ont été dirigés pour proposer des solutions de gestion QoS avec la technologie SDN.

Objectif de projet :

Algérie Télécom vise à améliorer son système réseau, de le gérer d'une manière programmable et de mettre en œuvre des services et des applications distribués sophistiqués, tout en assurant une meilleure qualité de service (Qos). Dans cette perspective, cette entreprise prestigieuse nous a proposé d'étudier la faisabilité d'une solution moderne, moins couteuse et plus facile à contrôler par rapport à l'architecture traditionnelle. Il s'agit de l'architecture SDN.

Dans ce travail, nous étudions l'architecture SDN en détail. Par la suite, on illustre la facilité d'implémentation d'une politique de QoS sur cette architecture réseau. La dernière section du mémoire est consacrée à une étude empirique qui démontre l'efficacité de l'approche QoS implémentée sur SDN en termes de débit et délais.

Plan du mémoire

Ce travail est réparti sur les chapitres suivants:

- **Chapitre 1 :** Présentation de la technologie SDN.
- **Chapitre 2 :** Qualité de Service (QoS) dans SDN.
- **Chapitre 3 :** Etude de performances de la gestion QoS dans un réseau SDN.

Chapitre I : Présentation de la technologie SDN

I.1 Introduction

Algérie Télécom est le leader actuel en matière de services télécommunication. Afin de conserver son avantage sur la concurrence, cette entreprise est activement à la recherche de solutions innovantes. C'est dans cette optique qu'Algérie Télécom étudie actuellement le déploiement du Cloud et du SDN afin de gérer son infrastructure efficacement. Ce chapitre est consacré à la présentation de l'organisme Algérie Télécom, ses activités et sa structure. Par la suite, on va définir le fonctionnement typique d'un réseau classique afin de déduire ses limites et inconvénients. Le reste du chapitre présente le SDN et ses avantages.

I.2 Présentation de l'organisme d'accueil

Algérie Télécom est une société par actions à capitaux publics opérant sur le marché des réseaux et services de communication électronique. Cette société est le leader sur le marché algérien des télécommunications même si ce secteur connaît une forte croissance. Offrant une gamme complète de services de voix et de données aux clients résidentiels et professionnels. Cette position s'est construite par une politique d'innovation forte et adaptée aux attentes des clients et orientée vers les nouveaux usages (2).

I.2.1 **Historique**

Les activités des services postaux, télécommunications et autres services connexes sont placés sous les auspices du ministère des postes et télécommunications, jusqu'à l'adoption de la loi n° 2000-03 5 aout 2000 qui séparent notamment les activités postales de celles des télécommunications (3).

Suite aux réformes tracées par le gouvernement et dans le cadre de la réorganisation en profondeur du secteur des postes et télécommunication, lui permettant l'évolution dans un environnement concurrentiel et transparent, la loi 2000-03 a ouvert le libre accès au secteur des télécommunications à tout opérateur obéissant aux prescriptions légales et réglementaires relatives au secteur (3).

Algérie Télécom a acquis la forme juridique de société par actions les 11 août 2001. La mise en œuvre de ses activités a été lancée à compter du 6 janvier 2002 et cela suite à la promulgation du décret exécutif 02-04 relatifs à la répartition des personnelles et des biens de l'administration des postes et des télécommunications (3).

Il est important de souligner que l'année 2002, a été une année dite « d'installation et d'organisation » d'Algérie Télécom alors que le 1er janvier 2003 a été la date de l'entrée officielle en activité (3).

I.2.2 **Structure de l'organigramme**

Algérie Télécom est organisée en Divisions, Directions Centrales, Directions Régionales, et Opérationnelles de télécommunication où il y a 13 directions régionales, et 53 directions opérationnelles distribués sur tout le territoire national dont trois à Alger et deux à Mohammadia. La figure 1 1illustre la structure de la société.

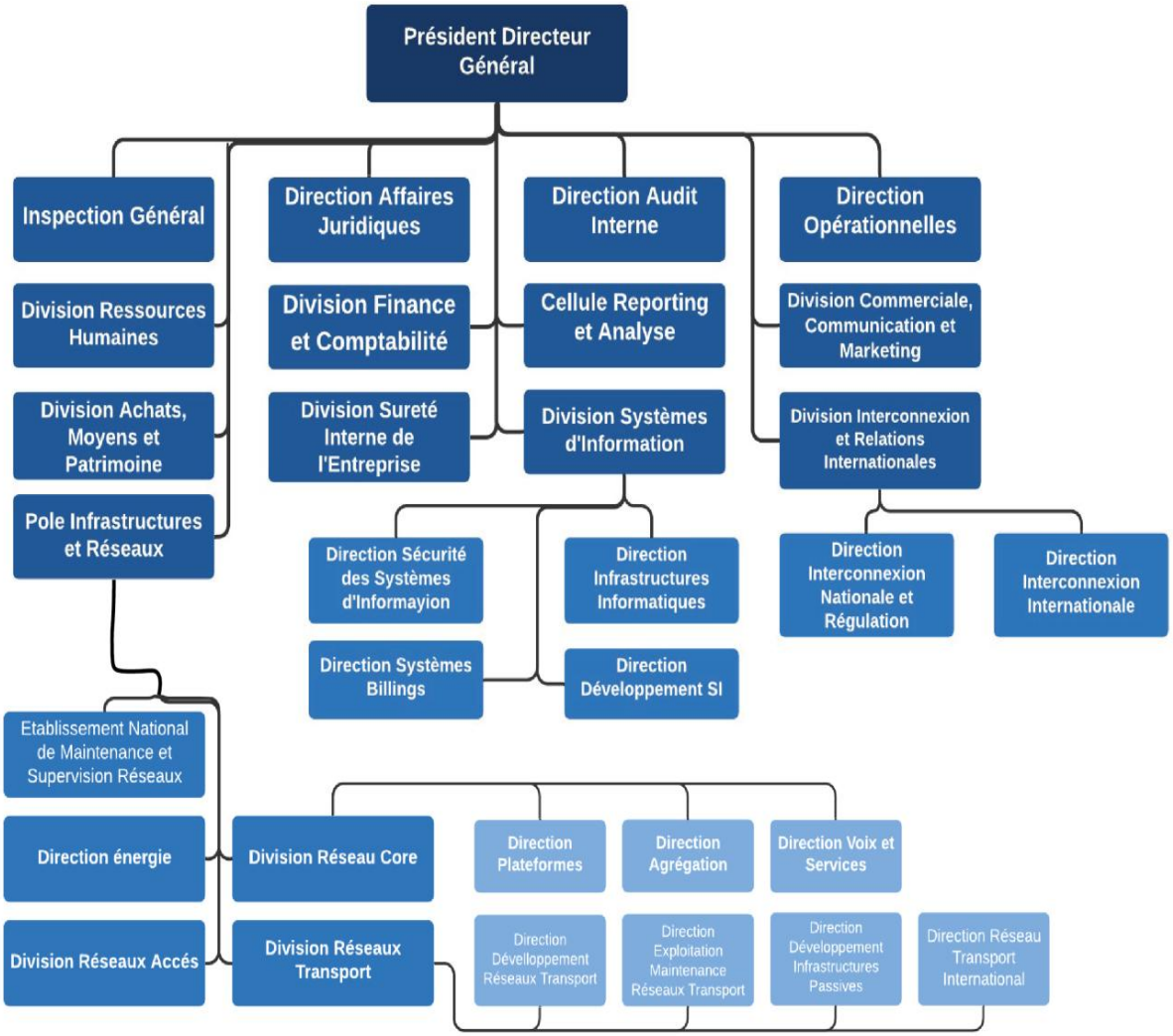


Figure I- 1 : Organigramme d'Algérie Télécom

Comme présenté dans la figure 2, le siège social d'Algérie Télécom est fixé à ALGER (route Nationale n° 5, cinq maisons EL HARRACH).

Figure I- 2 : L'emplacement géographique d'Algérie Télécom

I.2.3 **Activités commerciales**

Les activités majeures d'Algérie Télécom sont (3):

- ✓ Commerce en gros et détail, import/export, réparation, service après-vente pour les produit et accessoires de télécommunication.
- ✓ Montage et maintenance des structures et antennes dont elle dispose sur toute l'étendue du territoire national des équipements.
- ✓ Pièces de rechange et consommables liés au domaine des télécommunications, communications, téléphonie fixe et mobile.
- ✓ Exploitation des services Internet et généralement toutes les activités en relation avec les réseaux et services des communications électroniques.
- ✓ Fourniture des services de télécommunication permettant le transport et l'échange de la voix, de messages écrits, de données numériques, d'informations audiovisuelles.

I.3 Problématique

Les réseaux informatiques sont devenus incontournables aujourd'hui. Ils sont utilisés dans toutes les entreprises et même chez les particuliers. Ils permettent de mettre en œuvre des applications très diverses, des plus simples aux plus sophistiquées. Qu'il s'agisse de réseaux locaux, de réseaux sans fil, de réseaux d'exploitation ou de petits réseaux privés. Ces derniers sont tous soumis aux principes de structuration qu'il faut comprendre (4).

Les réseaux informatiques utilisent une structure en couches, où la communication entre les ordinateurs respecte des règles strictes définies par des protocoles de communication. Les protocoles les plus connus sont TCP et IP qui sont à l'origine du nom de l'architecture TCP / IP (4).

I.3.1 **Réseaux classiques**

Sur un réseau IP, la liaison entre un client et un serveur s'appuie sur des équipements de routage et des équipements de commutation. Le rôle du routeur est d'acheminer chaque paquet IP au nœud suivant afin que chaque paquet puisse être transporté de bout en bout, de la source vers la destination. Avant d'être déployés sur le réseau, les routeurs doivent être configurés. La configuration permet au minimum de définir les adresses IP des interfaces physiques et virtuelles du routeur et d'informer le routeur des protocoles de routage à appliquer pour définir comment acheminer les paquets Ip(5).

Les protocoles de routage permettent à chaque routeur de récupérer des informations des routeurs voisins afin de construire localement des informations de routage RIB. Ainsi, les informations de routage sont actualisées pour prendre en compte l'état de chaque nœud (saturé, hors ligne, …) de manière dynamique. Pour accomplir ceci, les routeurs échangent entre eux des informations par l'intermédiaire du protocole de routage choisi(5).

Par la suite, la table d'acheminement est exploitée par le routeur pour déterminer l'interface sur laquelle envoyer le paquet reçu selon l'adresse destination et la classe de service. Ainsi, le protocole de routage permet de constituer des règles qui sont synthétisées dans une table d'acheminement afin de router le paquet IP vers le prochain saut (next-hop) selon

la métrique utilisée pour mesurer la qualité de la route (nombre de saut, débit, délai, …) géré par le RIB (5).

I.3.2 Les inconvénients du réseaux classique:

Les réseaux TCP/IP traditionnels sont largement utiliser dans l'industrie. Changer cette architecture afin d'accommoder les nouvelles tendances en télécommunication est compliqué ainsi que couteux en temps et en argent. Cependant, cette architecture possède plusieurs désavantages (6) (7):

- ✓ L'infrastructure physique nécessite des configurations manuelles. Ceci ne permet tout simplement pas à suivre le rythme dévolution des applications modernes de télécommunication.
- ✓ L'emplacement physique du plan de contrôle empêche l'administrateur réseau de contrôler le flux de trafic.
- ✓ Les réseaux traditionnels ont tendance à être inflexibles, ce qui rend difficile pour l'exploitation et l'administration.
- ✓ L'accès à la couche de données doit se faire directement sur le matériel.
- ✓ Les protocoles de réseau sont spécifiques au fabricant.

I.4 Introduction aux réseaux SDN

Durant ces dernières années, les réseaux informatiques classiques ont connu de grands défis qui résultent principalement des applications modernes distribuées et leurs besoins. Toutefois, la refonte des réseaux classiques est réalisée en configurant manuellement chaque équipement ce qui peut produire plusieurs erreurs et incohérences. De plus, ces configurations fastidieuses prennent beaucoup de temps ce qui ne permet pas de faire face à l'expansion rapide du monde des techniques d'information (8).

Une des raisons de cette difficulté d'évoluer, ou d'administrateur simplement les réseaux, est le fort couplage qui existe entre le plan de contrôle et le plan de données des équipements d'interconnexions dans les architectures des réseaux classiques. C'est dans ce contexte que le

concept des réseaux SDN a émergé dans l'optique de combler les faiblesses du paradigme traditionnel (9).

SDN souvent désigné comme la nouvelle idée révolutionnaire dans les réseaux informatiques, promet de simplifier considérablement le contrôle et la gestion du réseau en permettant l'innovation grâce à la programmabilité de ce dernier (10).

I.5 Fonctionnement de base

Le SDN est un nouveau paradigme qui décrit une architecture réseau. Selon l'ONF (Open Network Fondation) (11) SDN est une architecture qui sépare le plan de contrôle du plan de données, et centralise toute l'intelligence du réseau dans une entité programmable appelée «Contrôleur», afin de gérer plusieurs équipements du plan de données (commutateurs et routeurs) via des APIs (9).

Plus concrètement, on peut dire qu'une architecture réseau suit le paradigme SDN si, et seulement si, elle vérifie les points suivants :

- ✓ **Séparation du plan de contrôle (Control Plane) et du plan de données (Data Plane):** Cela implique que ces deux plans résident indépendamment dans le réseau avec une communication étroite.
  - **La fonctionnalité de transfert** (plan de données) y compris la logique et les tableaux permettant de choisir comment traiter les paquets entrants en fonction de caractéristiques telles que l'adresse MAC, l'adresse IP et l'ID VLAN, réside dans les équipements réseau (12).
  - **Le plan de contrôle** détermine comment les tables de transfert et la logique dans le plan de données doivent être programmées. C'est pourquoi toute l'intelligence du réseau, qui est distribuée traditionnellement, est centralisée dans le contrôleur. Puisque dans un réseau traditionnel, chaque équipement dispose son propre plan de contrôle, la tâche principale de ce plan de contrôle est d'exécuter de créer des routes en remplissent les tables de transfert réparties des équipements réseau (12).

    Concrètement, un contrôleur SDN est une plate-forme logicielle de gestion qui fonctionne comme un serveur standard et facilite le provisionnement et la

programmation des périphériques de transfert sur la base d'une vue réseau globale, abstraite et centralisée (13). La figure 1 illustre la différence entre les réseaux traditionnels et SDN.

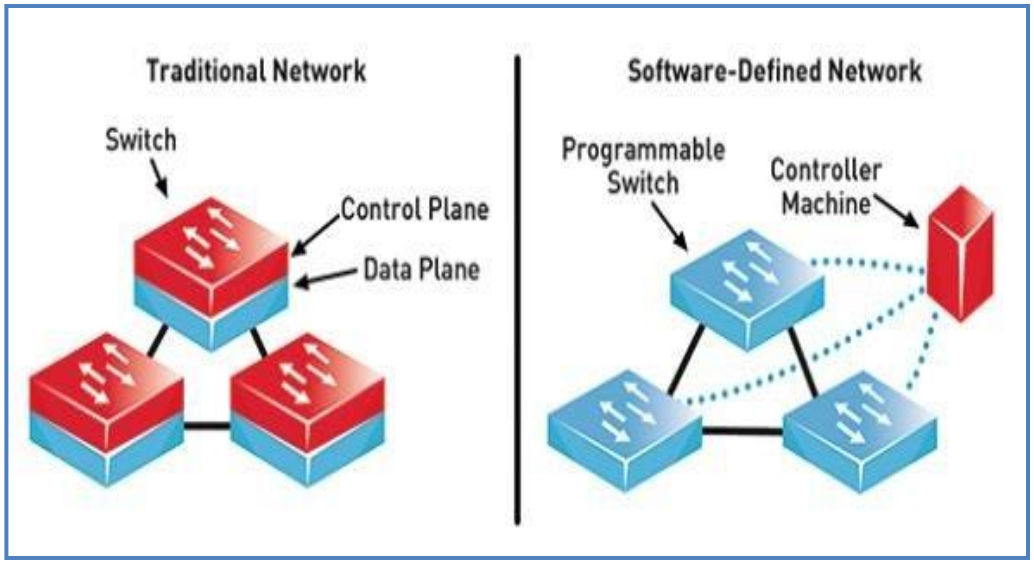


Figure I- 3 : Comparaison entre réseaux traditionnels et SDN (14)

Ainsi, au lieu de déployer le plan de contrôle d'une manière décentralisée, qui est un algorithme d'une complexité considérable, ce mécanisme est supprimé des équipements d'acheminement et placé dans un contrôleur centralisé. Ceci rend la configuration plus simple et moins fastidieuse. Le contrôleur est ensuite capable de transmettre des instructions primitives aux dispositifs réseau lorsque cela est nécessaire afin de leur permettre de prendre des décisions rapides concernant le traitement des paquets entrants (12).

- ✓ **Basé sur le flux:**

Contrairement au réseau traditionnel où le commutateur/routeur prend des décisions d'acheminement selon la destination désignée, le contrôleur SDN prend les décisions d'acheminement basées sur le flux. Un flux est considéré comme l'ensemble de paquets ayant des politiques de service identiques allant de la source à la destination (13).

I.6 Architecture SDN :

Selon l'ONF, SDN c'est une architecture émergente qui est dynamique, gérable, rentable et adaptable. Ceci rend cette technologie idéale pour la haute bande passante et la nature dynamique des applications d'aujourd'hui (11). L'architecture réseau basé sur SDN a été appliquée par plusieurs grandes entreprises et universités à savoir, Stanford et Google. En outre, les fabricants des équipements réseaux tels que Cisco, Juniper et HP offrent désormais des solutions SDN qui permettent de gérer les centres de données (8).

L'architecture SDN est :

- ✓ **Programmable :**
  Le contrôle du réseau est directement programmable car il est découplé des fonctions de transmission (11).
- ✓ **Agile :**
  Grâce à la retiration de contrôle du transfert, les administrateurs peuvent gérer dynamiquement les flux à l'échelle du réseau pour répondre à l'évolution des besoins (11).
- ✓ **Géré de manière centralisée :**
  L'intelligence du réseau est logiquement centralisée dans des contrôleurs SDN basés sur des logiciels (11). Néanmoins, plusieurs contrôleurs SDN peuvent coexister dans le même réseau.
- ✓ **Configuration automatisée** :
  Par l'utilisation des programmes SDN dynamiques et automatisés, les gestionnaires du réseau peuvent configurer, gérer, sécuriser et optimiser les ressources du réseau très rapidement (11).
- ✓ **Basé sur des normes ouvertes et indépendantes:**
  Lorsqu'il est mis en œuvre au moyen de normes ouvertes, l'architecture SDN simplifie la conception et le fonctionnement du réseau. Contrairement à la structure réseau traditionnelle, les instructions sont fournies par des contrôleurs SDN au lieu d'avoir multiples dispositifs qui exécutent des protocoles propres à chaque fournisseur (11).

La structure du SDN se compose de deux parties principales. Le niveau le plus bas inclut le plan de données alors que le plan de contrôle se trouve au dessus. Au plus haut niveau le plan application se situe qui peut gérer le réseau en interagissant avec le plan de contrôle.

La communication entre les contrôleurs et le plan de données est effectuée via le SBI (Southbound Interface) qui se trouve au niveau du commutateur SDN et la communication entre les applications et les contrôleurs est assurée par NBI (Northbound Interface) qui se trouve dans le plan de contrôle.

Ainsi, pour déployer de nouvelles politiques notamment (QoS, sécurité, équilibrage de charge), celles-ci sont programmés dans le plan application.. Ces applications et politiques sont construites avec les appelles de l'interface « Northbound API » (9). La figure 2 représente les couches de l'architecture SDN.

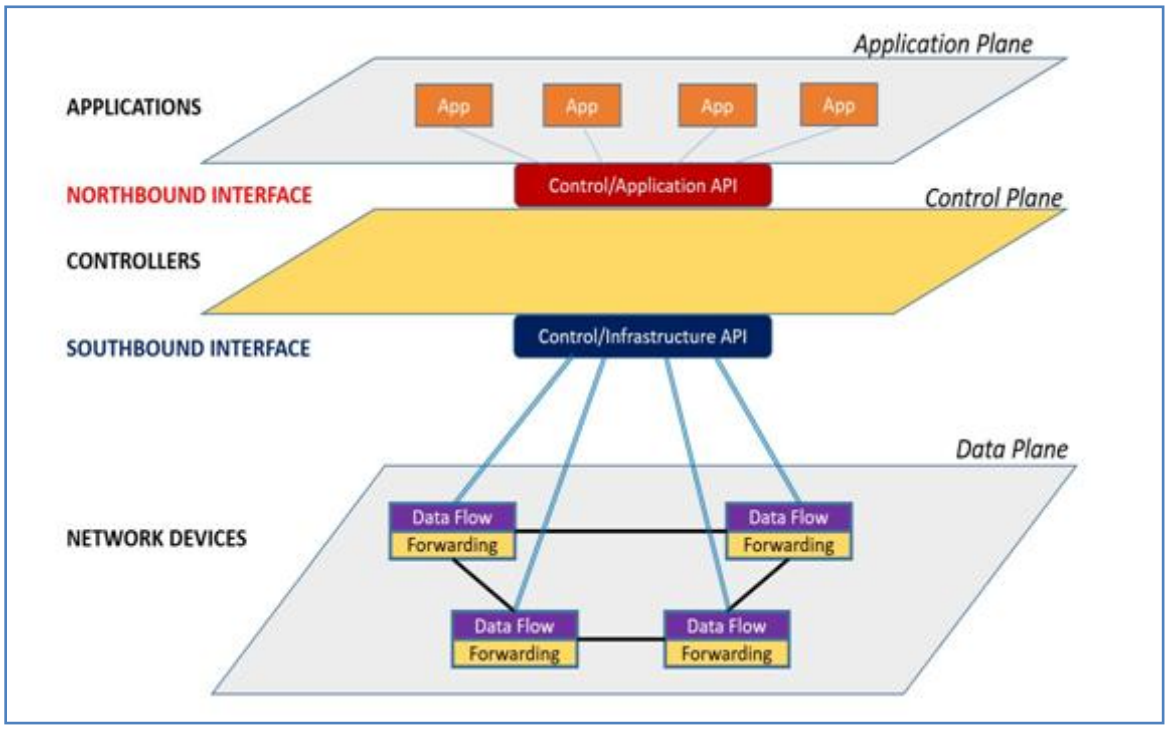


Figure I- 4 : L'architecture du réseau SDN (10)

I.7 Les interfaces SDN

Le SDN comporte trois APIs suivantes:

- l'interface en direction sud

- l'interface en direction nord
- l'interface en direction est-ouest

Nous illustrons ces interfaces dans figure 3 avec deux contrôleurs, les deux simultanément servant de NOS (Network Operating System) pour leur sous-réseau.

L'interface en direction nord permet aux programmes de contrôle (applications) de fournir des instructions au contrôleur. Via l'interface sud, le contrôleur envoie des instructions et reçoit des informations du plan de données. L'interface orientée est-ouest permet aux contrôleurs de partager une vue commune du réseau et de coordonner l'application des politiques et des protocoles (15). La figure 3 expose les trois types d'interfaces dans l'architecture SDN.

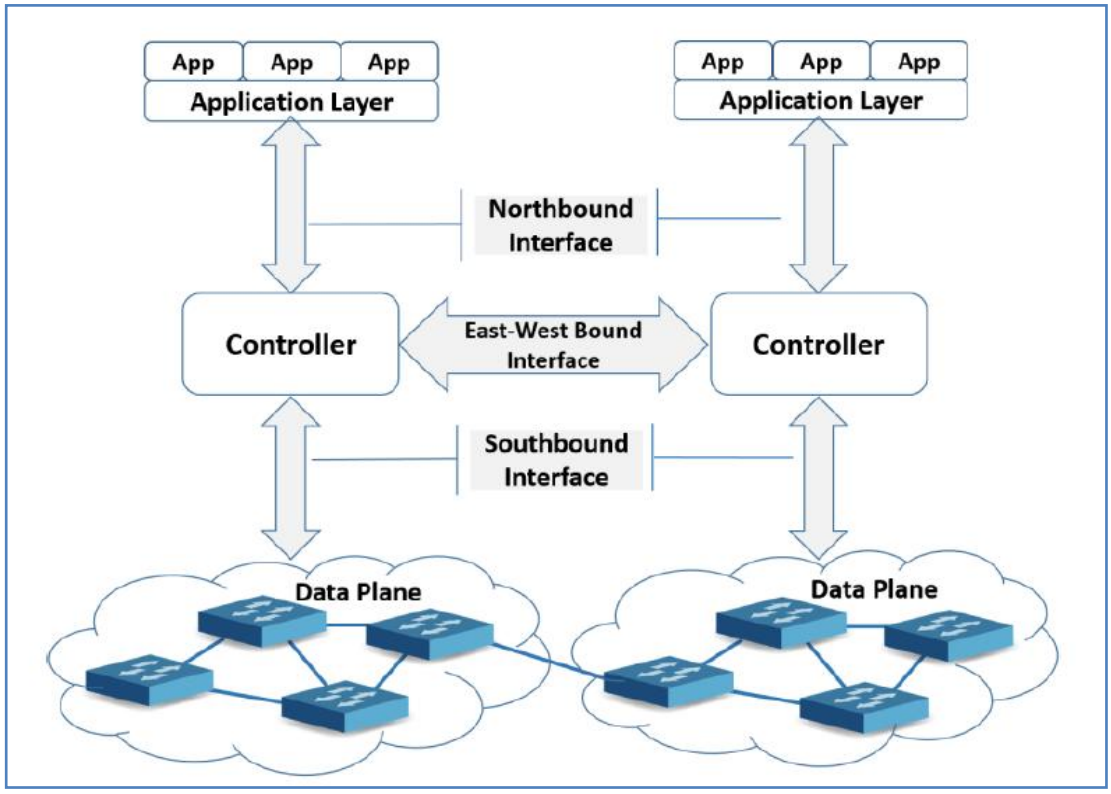


Figure I- 5 : Les interfaces de SDN (9)

I.7.1 **Southbound API**

Southbound API est l'une composante critique du système SDN, qui fait le pont entre les dispositifs d'acheminement et le plan de contrôle. Cette API permet au contrôleur de gérer le comportement du réseau en manipulant les tables des flux de tous les commutateurs sous-jacents.

L'API Southbound fournit une interface commune pour le contrôleur à utiliser celle-ci par exemple Open Flow, POF (16), OpFlex (16) et Open State (17) et les plugins de protocole pour gérer les périphériques, physiques ou virtuels, anciens ou récents (par exemple, SNMP, BGP et Net Conf). Ceci est essentiel pour assurer la compatibilité (18).

Par conséquent, sur le plan de données, un mélange de dispositifs physiques, virtuels par exemple, Open vSwitch, vRouter (19) et une variété d'interfaces de dispositifs par exemple, OpenFlow, OVSDB (20), OF-Config (OpenFlow Configuration and Management Protocol) , NetConf, et SNMP peuvent coexister (18). Actuellement, OpenFlow est le standard le plus adapté pour la norme cette interface.

I.7.2 **East/Westbound APIs**

East/Westbound API est un cas particulier d'interface requise par les contrôleurs distribués. Actuellement, chaque type de contrôleur implémente sa propre API en direction est-ouest. Les fonctions de ces interfaces incluent l'importation et l'exportation de données entre contrôleurs, algorithmes pour les modèles de cohérence des données, et capacités de surveillance et notification par exemple, vérifier si un contrôleur est en fonction ou notifier une prise en charge sur un ensemble de dispositifs de transmission (21).

I.7.3 **Northbound API**

La Southbound API a déjà une proposition largement acceptée c'est le OpenFlow, mais une Northbound API commune reste un problème ouvert. Pour l'instant, il est peut-être encore un peu trop tôt pour définir une Northbound API standard, car les cas d'utilisation sont encore en cours de développement (22). Principalement, cette API relie les programmes de contrôle (applications) au contrôleur. Grâce à cette API, les applications peuvent orchestrer et programmer le plan de données pour effectuer des tâches complexes telles que l'ingénierie du trafic, la découverte de la topologie, bien plus encore (23).

En fait, les contrôleurs existants tels que Floodlight et OpenDaylight (24) implémentent leur propre API vers le nord avec différentes spécifications et langages de programmation. En outre, les langages de programmation SDN tels que Frenetic (20), Nettle (25), NetCore (26), Procera (27)et Pyretic (28) ont également leurs propres spécifications et personnalisations de l'API vers le nord. (18)

I.8 Activités autour de SDN / OpenFlow :

Bien qu'OpenFlow a attiré l'attention de l'industrie, il est important de mentionner que l'idée de réseaux programmables et d'un plan de contrôle découplé du plan de données existe depuis de nombreuses années (12).

Le Groupe de travail sur la signalisation ouverte (OPENSIG) a lancé une série d'ateliers en 1995 pour rendre les réseaux ATM (Asynchronous Transfer Mode), Internet et mobiles plus ouverts, extensibles et programmables. Motivé par ces idées, un groupe de travail IETF a mis au point le protocole GSMP pour contrôler un commutateur d'étiquette. Plus tard, ce travail a été officiellement conclu par la publication de GSMPv3 en juin 2002 (12).

L'initiative Active Network a proposé l'idée d'une infrastructure de réseau qui serait programmable pour des services personnalisés. Cependant, Active Network n'a pas été adopté par la communauté principalement en raison de problèmes pratiques de sécurité et de performances (12).

À partir de 2004, le projet 4D a préconisé une conception claire qui mettait l'accent sur la séparation entre la logique de décision de routage et les protocoles régissant l'interaction entre les éléments du réseau. Les idées du projet 4D ont directement inspiré des travaux ultérieurs tels que NOX, qui proposait un système d'exploitation pour les réseaux dans le contexte d'un réseau compatible OpenFlow. Plus tard en 2006, le groupe de travail IETF Network Configuration Protocol a proposé NETCONF comme protocole de gestion pour modifier la configuration des périphériques réseau (12).

Le groupe de travail IETF Forwarding and Control Element Separation (ForCES) à conduit une approche parallèle du SDN avec ForCES. L'architecture de périphérique de réseau interne est redéfinie car l'élément de contrôle est séparé de l'élément de transmission, mais l'entité combinée est toujours représentée comme un élément de réseau unique pour le monde extérieur. Finalement, le prédécesseur immédiat d'OF était le projet SANE / Ethane de Stanford, qui a défini en 2006 une nouvelle architecture réseau pour les réseaux d'entreprise.(12) Ce projet s'est concentré sur l'utilisation d'un contrôleur centralisé pour gérer les politiques et la sécurité dans un réseau.

I.9 Protocole OpenFlow :

Le protocole OF a été initialement proposé et implémenté par l'université de Stanford, et plus tard il a été standardisé par la suite par l'ONF (29). Le protocole de communication le plus avancé entre un plan de contrôle logiquement centralisé et le plan de données est OF(30). Cependant, lors ses débuts ce protocole été la première implémentation réelle du concept SDN avec toute une architecture composée par les commutateurs, une canal de communication sécurisée et un contrôleur de référence (29).

La figure 4 schématise la position du protocole OF entre le plan de contrôle et le plan de données.

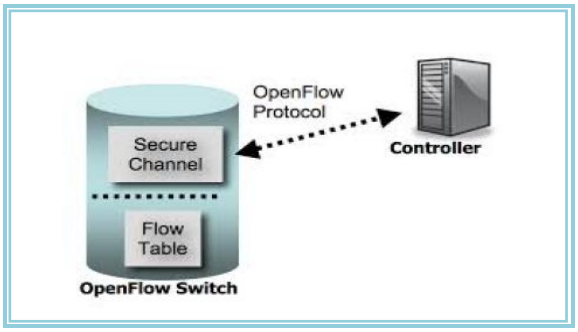


Figure I- 6 : Le protocole Openflow (20)

Le tableau 1 ci-dessous résume quelques-unes des améliorations significatives des versions OF :

Version OpenFlow	Support Description
OF v1.1	MPLS, VLAN, Multipath, Tables grouper, Tables Multi flux et Files d'attentes.
OF v1.2	In-Match, Packet-In, En-têtes extensibles, Flux Flexible match et IPV6.
OF v1.3	Tunneling, Tables de compteurs et Champs supplémentaires dans les tables de flux (Priorité, Timeouts, Cookie).
OF v1.4	Gestion optique des ports, tables synchronisées, surveillance des flux.
OF v1.5	Tables en sortie, statistiques d'entrée des flux, inclusion des drapeaux TCP en fonction du type de paquet et du type de canalisation.

Tableau I- 1 : L'évolution du protocole OpenFlow (16)

I.9.1 Commutateur Openflow :

Le commutateur OF est un élément de transfert de base, accessible via au contrôleur le protocole OF. Ces derniers sont disponibles en deux versions :

- ✓ **Hybrides (OpenFlow activé) :** Les commutateurs hybrides prennent en charge OF en plus du fonctionnement et des protocoles traditionnels (commutation L2 / L3). La plupart des commutateurs actuellement disponibles et commerciaux sont hybrides (10).
- ✓ **Pure (OpenFlow uniquement):** Les commutateurs Pure OF n'ont pas de fonctionnalités héritées ni de contrôle intégré et dépendent entièrement d'un contrôleur pour les décisions de transfert (10).

Un commutateur OpenFlow contient des tables de flux. Chaque table de flux contient un ensemble d'entrées de flux qui se compose de:

- ✓ **L'en-tête de paquet :** celui-ci permet d'identifier le flux de données et contient les informations nécessaires pour déterminer le paquet auquel cette règle sera appliquée. L'en-tête de paquet peut identifier différents protocoles tel qu'Ethernet, IPv4, IPv6 ou MPLS, cela dépend de la spécification d'OF déployée (9).
- ✓ **L'Action :** spécifie comment les paquets d'un flux seront traités. Une action peut être l'une des suivantes : a) transférer le paquet sur un ou plusieurs ports, b) supprimer le paquet, c) transférer le paquet vers le contrôleur, d) ou modifier un champ de l'entête du paquet (9).
- ✓ **Les Compteurs :** sont réservés à la collecte des statistiques de flux. Ils enregistrent le nombre de paquets et d'octets reçus de chaque flux, et le temps écoulé depuis le dernier transfert de flux (9).

La figure 5 représente un schématise des composants de la table de flux SDN.

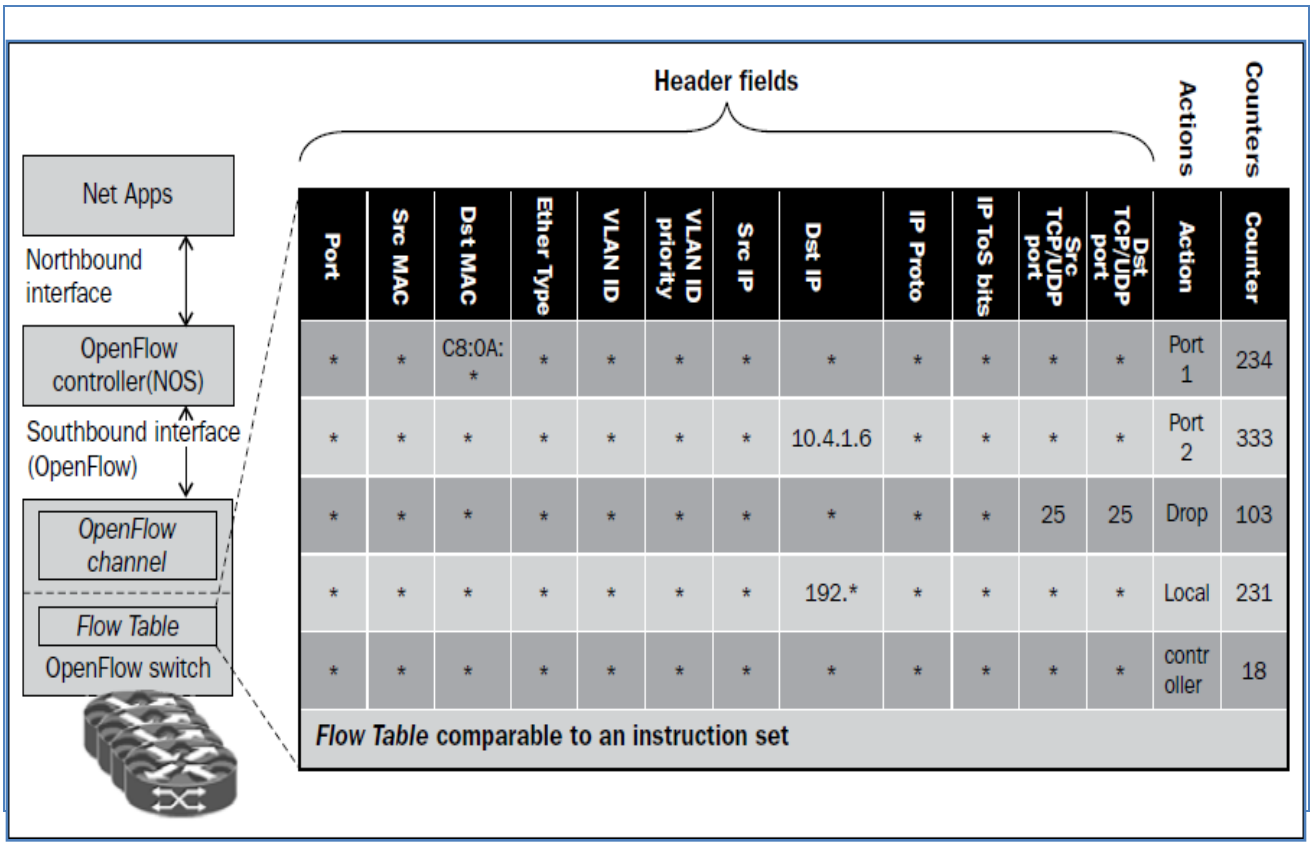


Figure I- 7 : Table de flux (3)

I.9.2 **Messages Openflow**

Lorsqu'un paquet arrive à un commutateur, celui-ci vérifie s'il y a une entrée dans la table de flux qui correspond à l'en-tête de paquet. Si c'est le cas, le commutateur exécute l'action correspondante dans la table de flux. Dans le cas contraire, s'il n'y a pas une entrée correspondante « 1 », le commutateur génère un message asynchrone vers le contrôleur « 2 » sous la forme d'un **Packet_in**. Par la suite, le contrôleur décide selon sa configuration d'une action pour ce paquet, et envoie une nouvelle règle de transmission sous la forme d'un **Packet_out** et **Flow-mod** au commutateur « 3 ». Finalement, la table de flux du commutateur est actualisée pour prendre en compte la nouvelle règle installée par le contrôleur « 4 » (9).

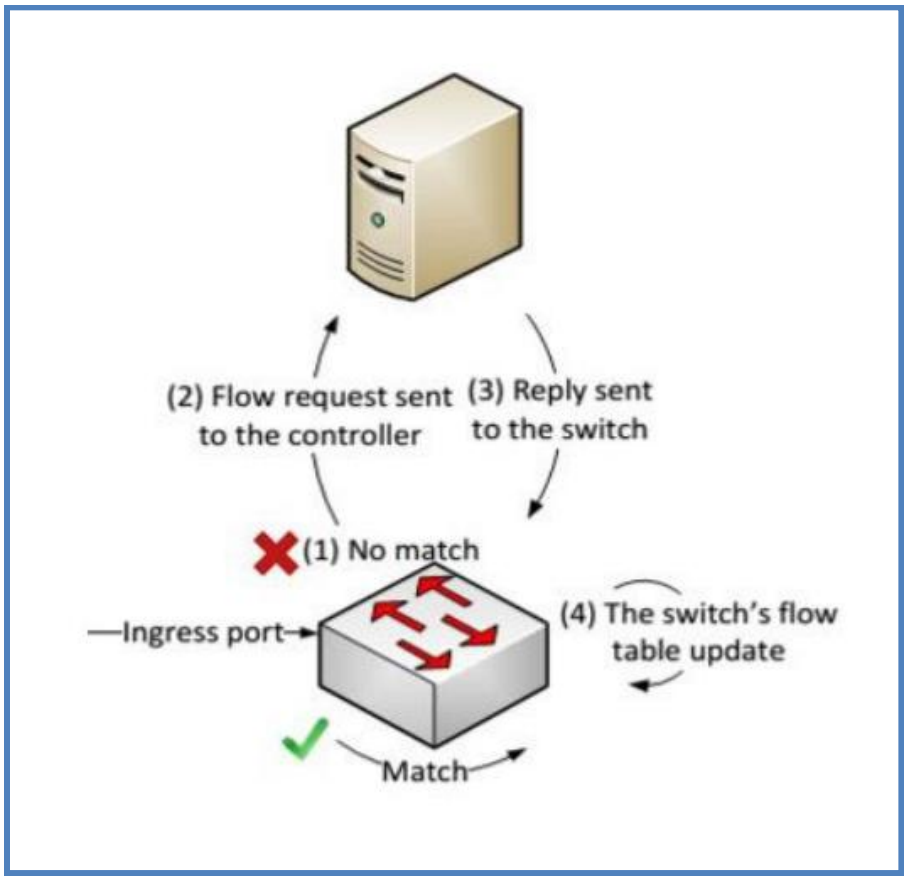


Figure I- 8 : Le processus de transmission d'un paquet avec Openflow (9)

La communication entre le contrôleur et le commutateur se fait à l'aide du protocole OpenFlow, où un ensemble de messages définis sont échangés entre ces entités sur un canal sécurisé implémentée via une connexion TLS sur TCP. Le commutateur initie la connexion TLS avec le contrôleur avec l'adresse IP de celui-ci préalablement connue. Chaque message entre le commutateur et le contrôleur commence avec l'en-tête OpenFlow. Cet en-tête précise le numéro de version OpenFlow suppoté, le type de message, la longueur de message, et l'identificateur de transaction du message (10).

Le protocole OpenFlow définit trois catégories de messages, chacun ayant plusieurs sous-types :

- ✓ Contrôleur-à-commutateur.
- ✓ Symétrique.
- ✓ Asynchrone.

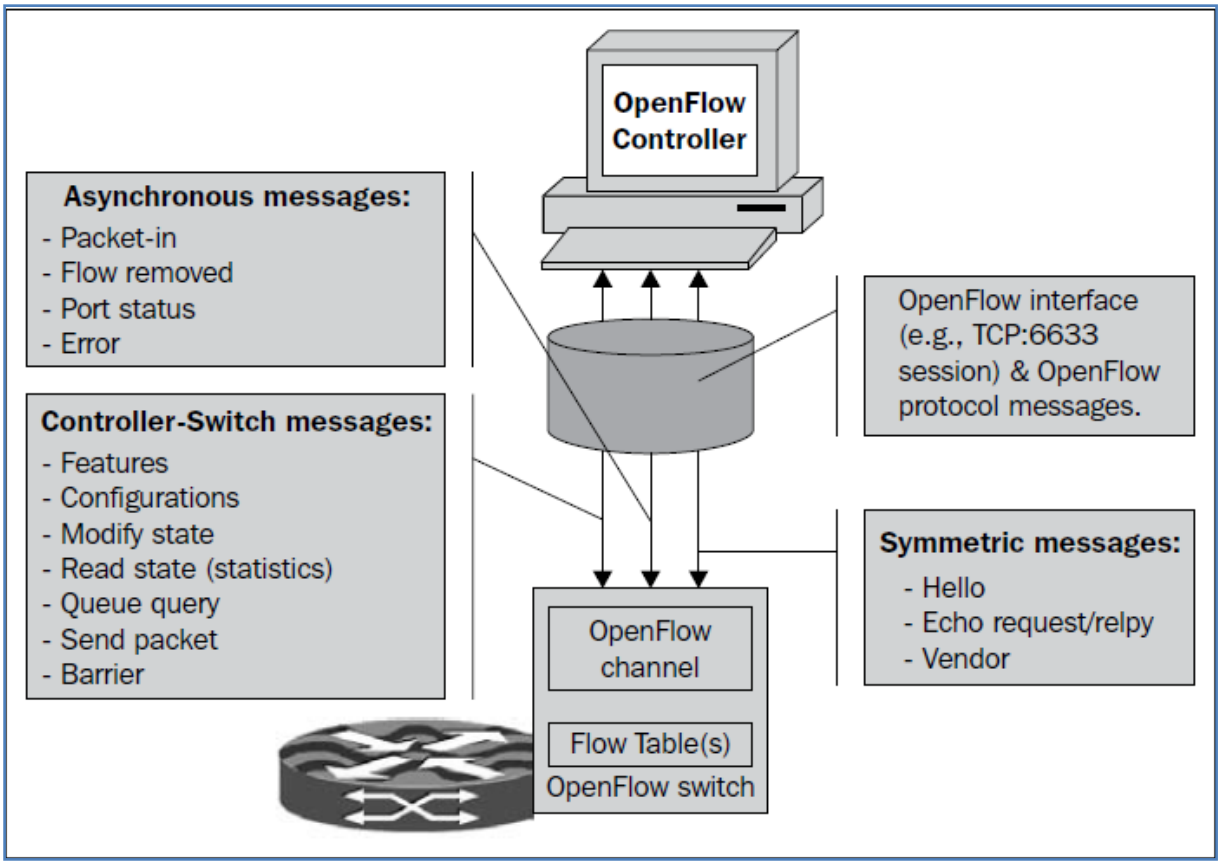


Figure I- 9 : Interface OpenFlow et protocole de messagerie (3)

I.9.3 Les messages Contrôleur-à-commutateur :

Les messages de contrôleur-à-commutateur sont initiés par le contrôleur et utilisés pour gérer ou inspecter directement l'état du commutateur. Ce type de messages peut nécessiter dans certaines situations une réponse de la part du commutateur. Les sous-types suivants sont classés sous cette catégorie :

- **Features :**

  Lors de l'établissement de la session TLS (par exemple, session TCP TLS sur le port 6633), le contrôleur envoie un message **OFPT_FEATURES_REQUEST** au commutateur et le commutateur Openflow répond via le message **OFPT_FEATURES_REPLY**.

  L'identifiant du chemin de données (datapath_id), les capacités du commutateur, les actions supportées et la définition des ports sont les fonctionnalités importantes signalées au contrôleur (10).

- **Configuration :**

  Le contrôleur peut définir et interroger les paramètres de configuration dans le commutateur avec les messages **OFPT_SET_CONFIG** et **OFPT_GET_CONFIG_REQUEST** respectivement. Le commutateur répond à une demande de configuration avec un message **OFPT_GET_CONFIG_REPLY**. Par contre, aucune réponse n'est nécessaire pour l'**OFPT_SET_CONFIG** (10).

- **Modify-State :**

  Le contrôleur est capable de modifier la table de flux avec le message **OFPT_FLOW_MOD.** En outre, le contrôleur utilise le message **OFPT_PORT_MOD** pour modifier le comportement des ports physiques. (10).

- **Read-State :**

  Le contrôleur peut interroger l'état du commutateur en utilisant le message **OFPT_STAT_REQUEST**. Le commutateur répond par un ou plusieurs messages **OFPT_STATS_REPLY**. La réponse du commutateur contient potentiellement plusieurs types d'information. Un champ type est utilisé pour préciser le type d'information qui est échangé (par exemple statistiques de file d'attente pour un port) et déterminer comment l' information doit être interprétée (10).

- **Queue query :**

  Un commutateur OF fournit un support de QoS limité grâce à un mécanisme de file d'attente simple. Une (ou plusieurs) file(s) d'attente peut être attachée à un port et peut être utilisé pour placer les flux. Les flux, qui sont mappés sur une file d'attente spécifique, seront traités en fonction de la configuration de cette file.

  Il 'est important de noter que la configuration de la file d'attente se fait en dehors du protocole OF (par exemple, via l'interface de commande) ou un protocole de configuration externe dédié. Le contrôleur peut interroger le commutateur pour les files d'attente configurées sur un port en utilisant un message **Queue query** (10).

- **Send-Packet :**

  En utilisant le message **OFPT_PACKET_OUT**, le contrôleur peut ordonner le commutateur OF d'envoyer des paquets à partir d'un port spécifique (10).

- **Barrier :**

Le message **OFPT_BARRIER_REQUEST** est utilisé par le contrôleur pour s'assurer que tous les messages OF émis par le contrôleur et qui ont précédé cette requête ont été reçus et traités par le commutateur. La confirmation est retournée par le commutateur via la réponse **OFPT_BARRIER_REPLY** (31).

I.9.4 Messages symétriques :

Les messages symétriques sont initiés par le commutateur ou le contrôleur et envoyés sans sollicitation. Il existe trois sous-types de messages symétriques dans le protocole OpenFlow comme suit:

- **Hello :**

  Les messages **Hello** sont échangés entre le commutateur et le contrôleur lors de la configuration de la connexion (10).

- **Echo :**

  Les messages **ECHO_REQUEST /REPLY** peuvent être envoyés à partir du commutateur ou du contrôleur. Ces messages peuvent être utilisés pour mesurer la latence, la bande passante et / ou la vivacité d'une connexion contrôleur-commutateur (10).

- **Vendor :**

  Ces messages fournissent un moyen standard pour les commutateurs OF pour déclarer des fonctionnalités supplémentaires dans les futures versions d'OF (10).

I.9.5 Messages asynchrones :

Les messages asynchrones sont initiés par le commutateur et utilisés pour notifier le contrôleur des événements réseau et des modifications de l'état du commutateur (10). Afin d'accomplir ce fonctionnement, les commutateurs envoient des messages asynchrones au contrôleur pour signaler l'arrivé de paquet, un changement d'état de commutateur ou une erreur.

Il existe quatre principaux messages asynchrones comme suit:

- **Packet-in :**

Pour tous les paquets qui n'ont pas d'entrée de flux correspondante ou si un paquet correspond à une action d'envoi au contrôleur, un message **PACKET-IN** est envoyé au contrôleur. Si le commutateur dispose d'une mémoire suffisante pour tamponner les paquets qui sont envoyés au contrôleur, le message **PACKET-IN** contiendra seulement une partie de l'en-tête de paquet (par défaut, 128 octets) et l'ID du paquet dans le tampon au lieu de transmettre le contenu complet du paquet. Les commutateurs qui ne prennent pas en charge le tampon interne, ou qui ne possèdent pas suffisamment d'espace tampon interne, doivent envoyer le paquet complet au contrôleur comme une partie du message (10).

- **Flow-Removed :**

  Le commutateur peut informer le contrôleur qu'une entrée de flux a été supprimée de sa table via **FLOW_REMOVED**. Cela survient lorsque aucun paquet entrant ne correspond avec cette entrée pendant une période de temps spécifiée par le contrôleur lors de la création de cette entrée au niveau de la table de flux du commutateur (31).

- **Port-status :**

  Le message **PORT_STATUS** est utilisé afin de communiquer un changement d'état du port (le lien est hors service) (31).

- **Error :**

  En cas de problème, le commutateur est capable d'informer le contrôleur en utilisant des messages d'erreur (10).

I.10 Contrôleur :

Dans les environnements SDN, les premiers déploiements où un seul contrôleur physique gouverne, ce dernier reste la pièce fondamentale de l'architecture et le point critique de son succès. (29) Car celui-ci prend en charge toutes les décisions concernant le fonctionnement du réseau.

De ce fait, la définition d'une politique de gestion d'un réseau revient à écrire des programmes et les déployer dans les contrôleurs. Dans la plupart des cas, ces programmes seront compilés, en tenant compte de la topologie et des ressources disponibles, afin de générer les

configurations nécessaires à chaque équipement du réseau pour mettre en œuvre les politiques désirées.(8)

Plusieurs contrôleurs SDN ont été développés, commerciales ainsi qu'open source. Ces derniers sont différents par leur architecture Centralisé ou Distribuée, leurs langages de programmation, les API qu'ils supportent est techniques utilisés (32). Le tableau 2 ci-dessous effectue une comparaison entre les propriétés des différents contrôleurs SDN.

Contrôleur	Langage de conception	Open source	Editeur	Langage supporté	Plateforme	Distribué	Aperçu
NOX	Python/C++	OUI	Nicira	C++	Linux	NON	Le premier contrôleur Openflow écrit en Python et C++.
POX	Python	OUI	Nicira	Python	Linux	NON	Améliorer les performances de NOX .
BEACON	Java	OUI	Standford		Linux/ Windows	NON	Multiplateforme, modulaire, qui prend en charge les opérations basées sur les événements et les unités d'exécution.
OPEN DAYLIGHT	Java	OUI	Linux Fundation	Java	Linux/ Windows	OUI	Support le framework OSGi et le REST API.
FLOODLIGHT	Java	OUI	Big Switch	Java/ Python	Linux/ Windows/ Mac	OUI	Basé sur l'implémentation Beacon. testé avec des commutateurs OpenFlow physiques et virtuels.
RYU	Python	OUI	NTT, OSRG group		Linux	NON	Un système d'exploitation SDN qui vise à fournir un contrôle et des API logiquement centralisés pour créer de nouvelles applications de gestion et de contrôle de réseau. Ryu supporte la majorité des versions d'OpenFlow.
HPE VAN	JAVA-based	OUI	Heweltt Packar	Java	HPE VAN	OUI	Le contrôleur HPE VAN fournit un point de contrôle unifié dans un réseau compatible OpenFlow, simplifiant la gestion, l'approvisionnement et l'orchestration et permettant la livraison d'une nouvelle génération de services réseau basés sur des applications.

Tableau I- 2 : Comparaison entre les propriétés des différents contrôleurs SDN

I.11 Défis SDN :

Cette partie se concentre sur les défis liés à la gestion des réseaux SDN et du protocole OF. SDN et OF offrent un moyen de simplifier le prototypage, le déploiement et la gestion des éléments du réseau. Cependant, nous devons également prendre en considération certains aspects importants qui peuvent conduire à l'indisponibilité ou la dégradation des services réseau (33) (34), comme suit :

1. **La disponibilité** du contrôleur est le principal aspect à considérer. La forte dépendance entre les commutateurs et le contrôleur chaque fois qu'une modification des règles est nécessaire ce qui peut créer un problème. De plus, si la conception du réseau ne prend en compte qu'un seul contrôleur centralisé, celui-ci pourrait devenir un «point de défaillance unique». Une approche distribuée pourrait être mise en œuvre pour garantir la disponibilité et éviter une éventuelle défaillance indésirable. De plus, une solution de redondance ou de sauvegarde pourrait être utilisée pour renforcer la robustesse (35).
2. **La sécurité** est également importante. Dans l'architecture SDN, le contrôleur est un composant ayant une connaissance critique du réseau et cet aspect expose le contrôleur à des attaques et menaces possibles. En outre, les canaux entre le contrôleur et les commutateurs pourraient être vulnérables. Selon la spécification Openflow, il est possible d'utiliser une communication sécurisée au moyen du protocole TLS (35).
3. **La cohérence des tableaux de flux** est également un problème potentiel. Étant donné que plusieurs contrôleurs peuvent gérer les mêmes tableaux de flux. Par exemple, un contrôleur de matériel de production et d'autres contrôleurs expérimentaux peuvent coexister. Par conséquence, ces derniers seront le maillon le faible. Par conséquent, ils pourraient être soumis à des contrôles de sécurité plus faibles, conduisant les tableaux de flux dans un état incohérent. Une mise en œuvre de l'outil « flow visor » peut convenir pour éviter ces menaces potentielles (35).
4. **L'évolutivité du réseau** dépend également du contrôleur, qui peut potentiellement devenir un «goulot d'étranglement». Si trop de paquets atteignent le contrôleur, des problèmes de performances peuvent survenir sur le réseau. Dans cette optique, il est important de prendre en compte la distribution du plan de contrôle pour éviter ces problèmes indésirables (35).

5. **Les performances du réseau** peuvent également être liées au modèle de contrôle adopté. La taille de la table de flux étant limitée, la gestion d'un très grand nombre de flux reste un défi majeur. Cependant, un réseau bien conçu pourrait réduire les problèmes de performances grâce à une approche proactive. En fait, l'approche proactive atteint de meilleures performances que le mode réactif car celle-ci limite la quantité de messages échangés entre le contrôleur et les commutateurs (35).

   Les flux peuvent être mis en place de deux manières : de manière proactive ou réactive.

   - La configuration proactive des flux a lieu avant que le paquet n'arrive au commutateur OF. Ainsi, lorsque le premier paquet arrive au commutateur OF, l'action correspondante à celui-ci serait déjà quoi connue. Il en résulte un délai de configuration négligeable et aucune limite réelle sur le nombre de flux par seconde que le contrôleur peut prendre en charge. Idéalement, le contrôleur SDN pré-remplit les tableaux de flux au maximum (36).
   - La configuration de flux réactifs se produit lorsque le commutateur OF reçoit un paquet qui ne correspond pas aux entrées du tableau des flux et que le commutateur doit donc envoyer le paquet au contrôleur pour obtenir le traitement assorti. Une fois que le contrôleur décide comment traiter le flux, cette information est installée dans la table du commutateur OF après que le contrôleur SDN détermine combien de temps il faut garder en mémoire cette règle (36).

I.12 Conclusion :

Au cours de ce chapitre, on a mis en évidence l'architecture SDN qui vise à révolutionner le fonctionnement réseau. On a commencé par introduire le concept de SDN et sa migration, puis on a présenté son architecture en détail de bas vers le haut, couches et interfaces. Ensuite on a parlé brièvement des travaux qui précèdent le SDN. Subséquemment, on a traité les composants essentiels d'OF et son fonctionnement. Et pour conclure ce chapitre, on a présenté les contrôleurs SDN les plus répandus ainsi que les défis de cette architecture. Dans le chapitre suivant on explore les méthodes de la qualité de service et les travaux connexes.

Chapitre II : La QoS dans le réseau SDN

II.1 Introduction :

Les réseaux de communication d'aujourd'hui nécessitent une meilleure gestion et la fourniture de QoS en raison de l'augmentation du nombre d'appareils et d'applications tels que la voix sur IP (VOIP), streaming vidéo et jeux vidéo Online. Afin de fournir de meilleures performances aux utilisateurs, la gestion de QoS est mise en place et doit être appliquée pour allouer efficacement les ressources du réseau et fournir une bonne expérience à l'utilisateur final (37).

Dans ce chapitre on présente la QoS en détaillant ses techniques et méthodes appliquées dans les réseaux IP traditionnelles. Par la suite, on donne un aperçu des principaux travaux liés au SDN et à OF. Plus précisément, on présente plusieurs contributions relatives à la QoS.

II.2 Définition de QoS :

QoS en tant que terme est une description générale des performances d'une connexion réseau. Ce terme est traité soit comme une évaluation *qualitative* des performances de connexion par un utilisateur, soit comme un ensemble de paramètres *quantitatifs* objectifs caractérisant celui-ci. L'évaluation *qualitative* de la QoS est définie comme le degré de satisfaction d'un utilisateur par la qualité de la communication. Par exemple, dans Skype la qualité du son, la gigue et la qualité de l'image sont des critères permettant de mesurer la QoS (38).

II.3 Paramètres de QoS:

Les organisations peuvent mesurer la QoS de manière quantitative en utilisant plusieurs paramètres, dont les suivants :

- **Bande passante :** La bande passante est la capacité maximale de transmission de données sur une liaison réseau dans un temps donné. (39).
- **Délai :** Le délai dans un réseau représente le temps nécessaire pour qu'un bit de données traverse le chemin entre la source et destination. Ce critère de performance est généralement mesuré en fractions de secondes (milliseconde). Le délai peut varier légèrement en fonction de l'emplacement de la paire de nœuds qui communiquent (39).
- **Jitter (La gigue):** La gigue est aussi appelée techniquement variation du retard des paquets. Il s'agit de la variance du délai en millisecondes (ms) entre les paquets de données sur un

réseau(40). Ainsi, un jitter trop instable peut dégrader la qualité de la communication vocale et vidéo.

- **Packet loss :** C'est un phénomène qui se produit lorsque les liaisons réseau deviennent encombrées et que les routeurs et les commutateurs commencent à rejeter des paquets. Lorsque la quantité de paquets perdus augmente pendant une communication en temps réel, comme des appels vocaux ou vidéo, la dégradation de la QoS devient apercevable (39).

Chaque application possède des besoins spécifiques en terme de QoS afin d'assurer son bon fonctionnement. Par exemple, une application multimédia nécessite « un débit élevé », la vidéoconférence et les jeux vidéos online nécessitent « une petite gigue et délai de bout en bout ». De son coté la télémédecine (chirurgie à distance) nécessite « un débit élevé et un faible taux d'erreur » (38). Pour répondre à ces exigences, des mécanismes de QoS bien définis doivent être appliquées dans le réseau.

II.4 Modèles de QoS :

L'Internet Engineering Task Force (IETF) a défini différents types d'architectures pour prendre en charge l'approvisionnement QoS. Trois modèles existent pour implémenter la QoS: a) Best Effort, b) Integrated Services (IntServ), c) Differentiated Services(DiffServ).

- ✓ **Best Effort** : est un modèle de qualité de service dans lequel tous les paquets reçoivent la même priorité et il n'y a pas de garantie de livraison des paquets. Le modèle Best Effort est appliqué lorsque le réseau n'est pas configuré pour appliquer des politiques de QoS ou lorsque l'infrastructure ne prend pas en charge la QoS (39) .
- ✓ **IntServ :**

  Le modèle Intserv (41) traite les trafiques par flux. Celui-ci utilise le protocole de réservation des ressources « RSVP » (42) pour fournir la qualité de service aux utilisateurs finaux. Ce modèle consiste à assurer les ressources requis pour un flux entre source et destination par la réservation explicite des ressources (bande passante et délai). Avant de débuter la transmission des données, l'application commence par demander la réservation des ressources nécessaires avant de commencer la transmission. Par la suite, selon les ressources présentes dans le réseau, la demande de l'application peut être satisfaite ou rejetée par les routeurs intermédiaires. Une fois que la demande de l'application est acceptée, l'application doit fonctionner en respectant les ressources

initialement requises. Par conséquent, les routeurs intermédiaires gardent à jour une table des états des flux qui les traversent(43). La Figure III-1 présente le protocole RSVP.

Bien qu'IntServ soit un moyen de fournir une garantie Hard QoS, cette stratégie n'a jamais décollé comme une solution tout-en-un pour les exigences QoS du réseau.(44) Cette situation s'explique principalement par les inconvénients suivants d'IntServ :

- Tous les routeurs constituant le chemin doivent supporter IntServ.
- Difficulté d'implémentation et complexité de gestion des flux.
- Le passage à l'échelle avec IntServ est difficile et ainsi, ce modèle est applicable sauf dans les petits réseaux. (39)

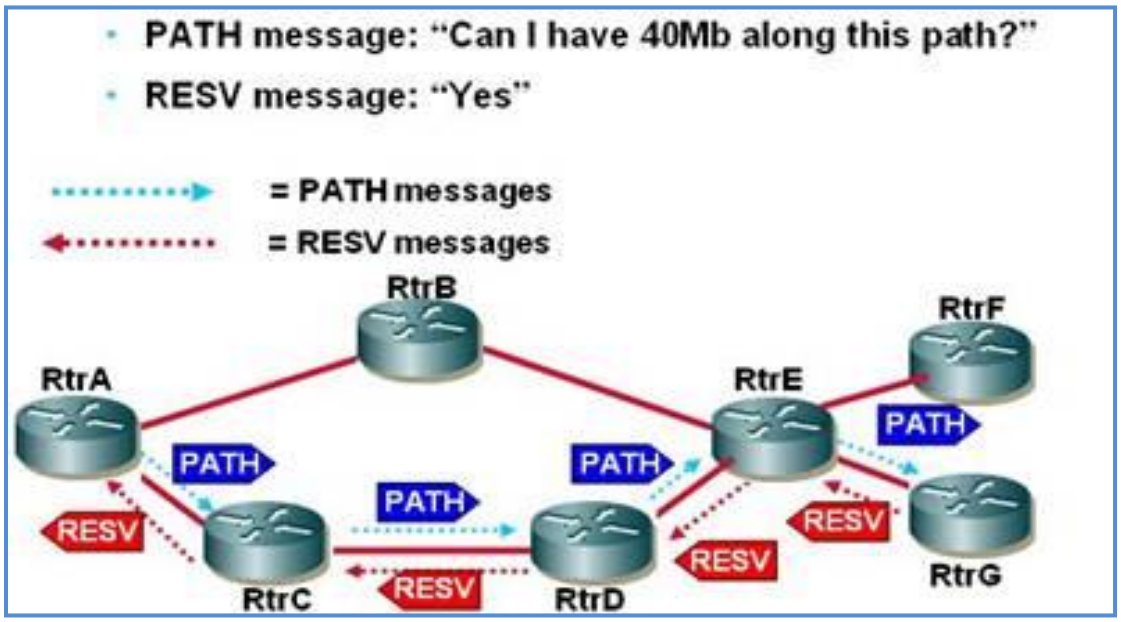


Figure II- 1 : Le protocole RSVP (45)

✓ **DiffServ :**

Le passage à l'échelle était le principal obstacle à la croissance d'IntServ en tant que mécanisme de QoS (46). Ainsi, DiffServ (47) a été introduit pour résoudre ce problème. Alors qu'IntServ traitait le trafic de bout en bout, DiffServ est un modèle qui s'applique par chaque saut. C'est ce qu'on appelle le « Per-Hop Behavior » (PHB). Contrairement à IntServ qui fonctionnait par flux, DiffServ fonctionne sur des flux agrégés (appartenant à la même classe). Les paquets sont classés selon les bits DSCP (Differentiated Services Code Points) qui occupent les six premiers bits de l'octet ToS

IPv4 (Traffic Class dans IPv6) contenus dans l'en-tête du paquet. Tous les paquets avec la même valeur DSCP reçoivent le même traitement, quels que soient les flux dont ils font partie. Contrairement à IntServ, DiffServ devient ainsi plus facile à mettre en œuvre et à maintenir sans les frais de transmission de messages de réservation.

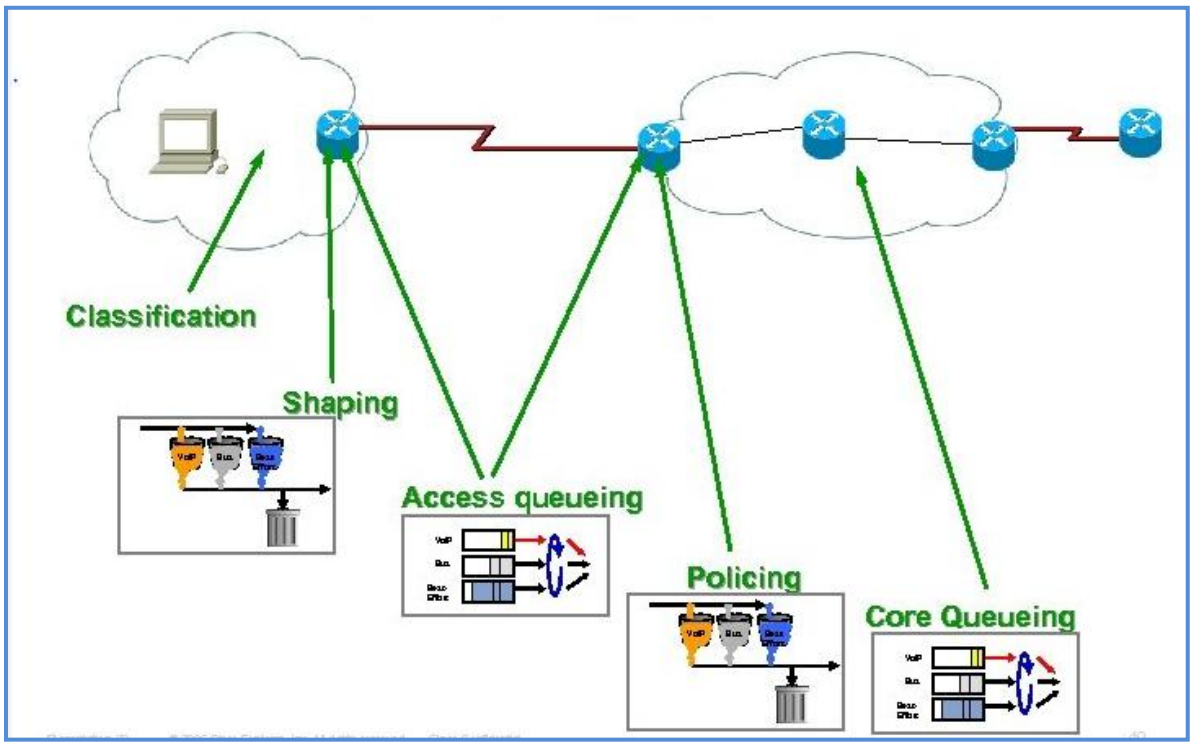


Figure II- 2 : Le protocol DiffServ (48)

En outre, le **Multi protocol Label Switching (MPLS) (26)** est un protocole qui est utilisée pour réduire les recherches complexes dans les tables de routage par des techniques d'étiquetage.

Tous ces avantages et inconvénients montrent que les architectures actuelles de qualité de service ne sont pas vraiment efficaces pour soutenir la qualité de service des fournisseurs de services, des entreprises et/ou des utilisateurs finaux.

II.5 Mécanismes de la QoS :

Les mécanismes de QoS peuvent gérer la performance du trafic de données et maintenir les exigences de l'application spécifiées dans les SLA (Service-Level Agreement). Les mécanismes de QoS entrent dans des catégories spécifiques selon le rôle qu'ils jouent dans la gestion du réseau.(44)

- ✓ **Traffic Classification et Traffic Marking :** la classification est le processus qui consiste à examiner les champs d'en-têtes afin de placer le paquet dans la file d'attente appropriée. Le marquage est le processus qui consiste à modifier les valeurs des champs « ToS » ou « Traffic Class ». (49)

  La classification et marquage différencient et trient les paquets en différents types de trafic. Le marquage marquera chaque paquet comme un membre d'une classe de flux, ce qui permet aux périphériques du réseau de reconnaître la classe du paquet instantanément. Ces techniques sont mis en œuvre sur les périphériques réseau tels que les routeurs, les commutateurs et les points d'accès (39).

- ✓ **Congestion Management (Queuing) :** La gestion de la congestion fait référence aux outils mis en œuvre sur les interfaces qui placent les paquets dans des files d'attente, sortes de tampons, lorsque l'interface est occupée par la transmission d'un autre paquet « soit de la congestion » (49). Les techniques de gestion de files d'attente utilisent marquage et la classification. Les techniques suivantes sont les plus utilisées :
  - ***FIFO queuing*** : la méthode "First In First Out" est la méthode par défaut sur les interfaces ;
  - ***Priority Queuing (PQ)*** : avec le *PQ*, le trafic est placé dans quatre files d'attente : high, medium, normal, et low ;
  - ***WRRQ, CBWFQ et CBWFQ-LLQ queuing* :** les méthodes de mise en file d'attente suivantes sont basées sur la priorité du trafic selon sa classe:
    - *Weighted round robin queuing (WRRQ)* utilise des poids configurés pour chaque file d'attente sortante.
    - *Class-Based Weighted Fair Queueing (CBWFQ)* se base sur les classes pour leur accorder un poids dans une file d'attente.
    - *Class Based Weighted Fair Queueing with Low Latency Queueing (CBWFQ-LLQ)* offre la possibilité de configurer une file d'attente hautement prioritaire à faible latence.

- ✓ **Traffic Shaping** : Cette technique mesure le taux de trafic et mets en tampon le trafic excessif de telle sorte que l'application ne dépasse pas le taux limite prédéfinie. Le taux

limite est désigné par le Committed information Rate (CIR), soit le taux minimum de transfert garanti pour l'application(49).

- ✓ **Traffic Policing :** Cette technique prend une action spécifique sur un trafic qui dépasse une limite de taux de trafic. Contrairement à la technique précédente, ce mécanisme ne met pas en tampon ou en file d'attente les paquets du trafic correspondant (49)**.**

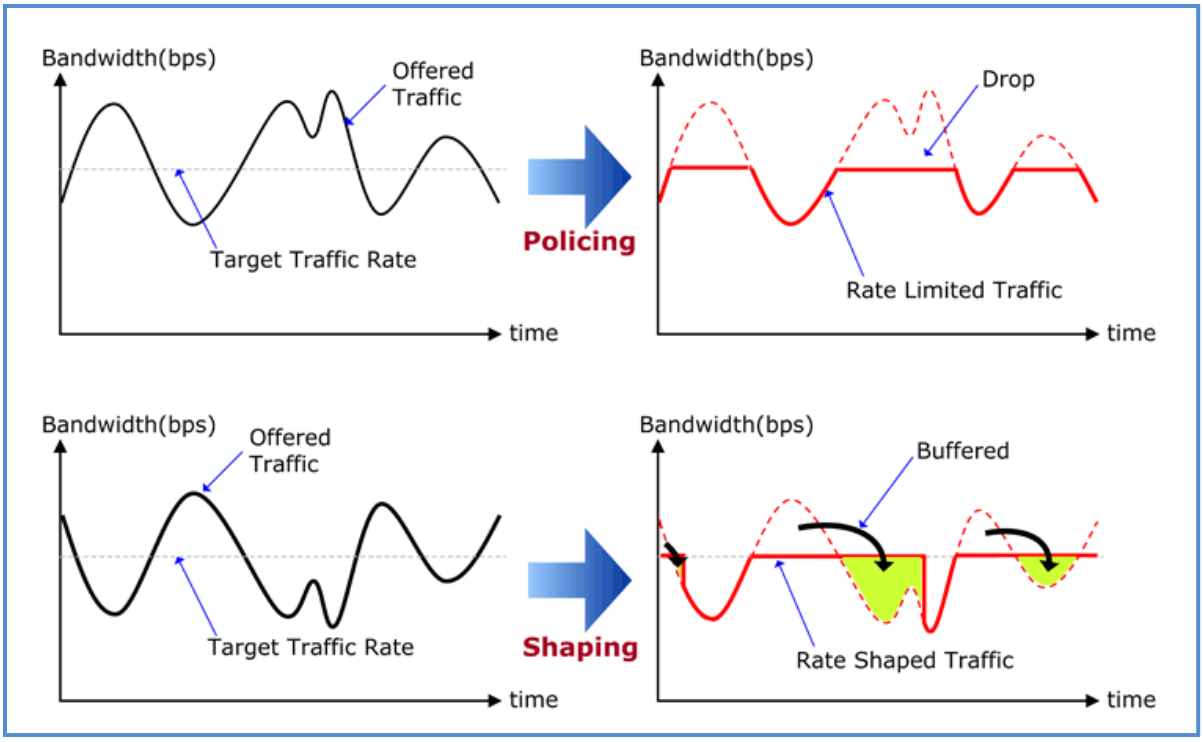


Figure II- 3 : Traffic Shaping VS Traffic Policing (50)

- ✓ **Congestion avoidance** : Les techniques d'évitement de congestion surveillent la charge du trafic sur les interfaces afin d'anticiper les étranglements (49).

  Deux algorithmes d'évitement de congestion sont utilisés sur les commutateurs Cisco :

  - **Tail Drop** (par défaut)
  - **Weighted Random Early Detection (WRED)**

II.6 L'état de l'art de la QoS dans les SDN

BaProbSDN (51) est un algorithme de routage QoS développé pour SDN, qui utilise la mesure de disponibilité de la bande passante calculée par le commutateur pour la livraison de données unicast. BaProbSDN utilise un réseau bayésien représenté par un graphe orienté

acyclique et le théorème de Bayes pour déterminer la probabilité qu'une liaison peut assurer la bande passante nécessaire pour un nouveau trafic afin de sélectionner la route appropriée. Lors de l'exécution de cet algorithme, les commutateurs doivent périodiquement transmettre au contrôleur la bande passante disponible de chaque lien. Afin de réduire les surcoûts associés à cette opération dans le réseau SDN, BaProbSDN utilise une politique de mise à jour des liaisons. Avec cette technique, les commutateurs ne diffusent les bandes passantes de chaque lien que lorsqu'un changement significatif survient. Les résultats produits par cet algorithme montrent que la surcharge peut être considérablement réduite avec un impact moins significatif sur les performances en termes de congestion réseau. A la fin, les résultats de comparaison montrent que BaProbSDN peut réduire jusqu'à 7,41% le taux de congestion de la bande passante par rapport au WSR widest-shortest path routing en présence d'une politique de mise à jour de l'état de liaison.

HiQoS (52) propose une solution multi-chemin, exploite un algorithme ECMP (Equal Cost Multipath Routing) basé sur le SDN, présenté dans (53) . Hiqos est principalement divisé en deux composantes :

- ✓ Une composante des services différenciés, qui consiste à classifier le trafic en: a) flux vidéo, b) audio/vidéo interactive, c) flux de données normal. Chaque type de flux est placé dans sa file d'attente spécifique avant d'être transmis avec la priorité accordée à celle-ci.
- ✓ La composante de routage multi-chemin qui trouve plusieurs chemins répondant à certaines contraintes QoS entre lsource et destination. Par la suite, le chemin optimal pour chaque flux est choisi et les autres chemins découverts sont préservés. Par la surveillance en temps réel de l'état du réseau, le contrôleur peut utiliser les chemins alternatifs.

Les résultats montrent que HiQoS peut réduire les délais et augmente le débit transmission pour garantir la QoS. En outre, HiQoS se remet très rapidement des défaillances des liaisons en redirigeant le trafic d'un chemin défaillant vers un autre chemin découvert lors de la phase de création multi-chemin.

Une approche de garantie de QoS est proposée dans cet article (54) pour l'allocation de bande passante qui satisfait les exigences de QoS pour tous les utilisateurs de cloud prioritaires en utilisant Open vSwitch et un contrôleur SDN afin de surveiller les performances de bout en bout des services utilisateur. La contribution proposée dans cette article comprend les deux composants suivants: a) introduction d'une nouvelle métrique basée sur la bande passante, la longueur du chemin ainsi que le RTT (Round Trip Time), b) politiques de gestion de la file d'attente multi-utilisateurs du cloud.. Les résultats expérimentaux montrent l'efficacité de la méthode proposée. Par conséquent, les solutions résultantes de l'approche sont applicables à plusieurs types d'applications et services cloud, y compris les applications multimédias.

L'article (55) propose un système pour permettre le contrôle et routage QoS dans SDN. Au début, les développeurs de ce protocole ont commencé par le contrôleur Floodlight car celui-ci implémente nativement l'algorithme Shortest Path (Dijkstra), qu'ils ont utilisé pour développer ce système. Le contrôleur modifié fournit un routage plus efficace par deux méthodes : a) celui-ci prend en considération l'état du réseau comme une mesure de la QoS pour surveiller la charge des commutateurs ou des liens et connaître l'état de la transmission. De plus, ce mécanisme utilise plusieurs files d'attente sur le port des commutateurs afin de différencier le trafic ordinaire (Best Effort) du trafic privilégié (EF). Au moment d'installation d'une règle pour un flux par le contrôleur, celui-ci s'occupe aussi de la définition de la file d'attente correspondante au nouveau flux. Les résultats expérimentaux montrent que ce système se comporte comme prévu, gère de manière plus efficace les ressources du réseau et donne des garanties sur le traitement du trafic.

La proposition dans (56) est consacrée au développement du modèle de flux multi-produits étendu afin de fournir un routage optimal qui permet d'équilibrer la charge à la fois sur les critères de charge maximale/minimale des canaux réseau et la QoS pour chaque flux. Au cours de ce travail, les auteurs catégorisent les flux en trois classes. En outre, les auteurs ont introduit un nouveau paramètre, appelé priorité relative, qui est utilisé pour identifier de manière unique le flux en fonction de ses exigences QoS ainsi que deux paramètres supplémentaires qui prennent en compte la priorité des utilisateurs et la sensibilité du flux aux redirections. Par conséquent, après avoir calculé les priorités relatives et l'application du modèle sur un scénario où le réseau surchargé, ils ont obtenues une amélioration significative des indicateur QoS.

Martin Reisslein et al (57) ont introduit une nouvelle méthode de garantie de QoS qui modélise le plan de contrôle afin de prendre en charge le routage en ligne. La méthode proposée assure le contrôle d'admission dans un réseau modélisé comme un graphe de liens de files d'attente où chaque lien physique héberge plusieurs liens de file d'attente avec différents niveaux de QoS. En effet, l'algorithme de routage à base de contrainte de délai DCLC (*Delay Constrained Least-Cost*) sélectionne l'itinéraire qui détermine à la fois le chemin emprunté par un flux admis à travers les liaisons physiques ainsi que les files d'attente QoS que le flux traverse. En outre, cette proposition prend des décisions précises de routage et d'admission avec une complexité algorithmique réduite par rapport au modèle MIP (Mixed Integer Programming) qui a besoin de quelque centaines de secondes pour produire une route. Aditionellement, cette approche atteint jusqu'à environ 93% d'utilisation moyenne des liaisons dans un scénario de communication industrielle.

MPRSDN (MultiPath Routing SDN) (38) propose une nouvelle approche pour gérer la QoS des connexions dans les réseaux SDN basé sur le routage à trajets multiple. En effet, le MPR permet à cette approche d'augmenter considérablement l'espace de recherche pour les ressources inactives et d'aboutir à une meilleure allocation de celles-ci. Contrairement à la majorité des propositions QoS qui utilisent Diffserv ou IntServ, cette proposition divise un flux en plusieurs sous-flux et les traite avec le modèle Best Effort. En ce qui concerne l'infrastructure réseau, ce protocole améliore considérablement le taux d'utilisation des ressources inactives dans l'espace de recherche..

La proposition (58) décrit l'architecture de **FlowQoS** qui permet aux utilisateurs de spécifier la bande passante maximale autorisée pour des applications spécifiques dans un réseau domestique à l'aide d'un outil de configuration Web. Cet outil comporte deux composants principaux :

- ✓ Le classificateur de flux : Ce composant utilise deux modules pour effectuer la classification du trafic. Le premier module effectue l'identification précoce de l'application du trafic **HTTP** et **HTTPS** grâce aux requêtes DNS, et le second classificateur effectue l'identification de l'application pour les autres flux (non **HTTP** et **HTTPS**)

- ✓ Le contrôleur de débit : Sur la base des résultats de la classification, le contrôleur installe des règles dans le commutateur pour ce flux. Une fois que l'association entre les flux et les applications est reconnue, le contrôleur de débit SDN de FlowQoS attribue à chaque flux le débit approprié.

Ensuite, cette proposition introduit deux commutateurs "virtuels" à l'intérieur du routeur domestique. Avec ceci, les connexions virtuelles entre ces deux commutateurs sont ensuite configurées via l'utilitaire TC (Traffic Control) de Linux et affectées à différents taux spécifiés par le contrôleur qui sont préalablement prédéfinis par l'utilisateur. Avec cette configuration, chaque type de trafic occupe une liaison virtuelle. Les résultats préliminaires montrent que FlowQoS améliore les performances du streaming vidéo et de la VoIP en présence de trafic concurrent actif.

SAQR (Simulated Annealing Based QoS-aware Routing) (59) est un algorithme de routage compatible avec SDN hybride qui comporte des switchs SDN et des switchs legacy coexistant. SAQR peut adaptativement régler les poids des contraintes requises (délai, bande passante et taux de perte) dans la fonction de coût pour trouver le meilleur chemin. En effet, le système de routage contient deux sous-modules:

- ✓ Topology discovery module : ce module reçoit les Paquets-IN et crée une topologie virtuelle dans le contrôleur SDN. En ce qui concerne les switchs legacy L2, ce système les détecte grâce au protocole STP (Spanning Tree Protocol) comme étant un seul switch virtuel. De même, ce protocole détecte les machines reliées grâce aux méssages ARP.
- ✓ Flow scheduling module : ce module crée le chemin le mieux adapté selon : a) un SLA prédéfini, b) la topologie virtuelle, c) l'état du réseau.

Les résultats de la simulation ont montré que le SAQR donne de meilleurs résultats que le MINA (60), en ce qui concerne en terme de délai, taux de perte et bande passante, avec 88 %, 90,8 % et 86,5 % des flux hébergé selon leurs exigences QoS respectives.

Dans la proposition (61), deux nouveaux algorithmes de routage QoS sont développés qui exploitent les avantages que SDN apporte pour améliorer les performances du réseau.

- ✓ La première solution concernant un contrôleur SDN centralisé, qui propose l'algorithme QVR (QoS-aware Virtualization-enabled Routing) comme protocole de routage pour la

gestion de multi-tenance, afin d'acheminer les flux à travers des chemins qui remplissent les exigences QoS correspondantes. En effet, cette approche simplifie la gestion des flux complexes liés à des applications spécifiques et définit les files d'attente prioritaires autour d'une application spécifique. Cependant, des problèmes avec la mise à l'échelle du modèle centralisé surviennent en particulier dans les réseaux à grande échelle.

- ✓ Le deuxième algorithme de routage QRLR (QoS-aware Reinforcement Learning Routing) est proposé pour un contrôleur SDN distribué basé sur l'apprentissage par renforcement. où chaque contrôleur joue le rôle d'un agent de décision qui apprend les comportements des flux entrants et prend des mesures pour améliorer le chemin à traverser dans le réseau. Grâce à ce déploiement distribué, cette solution permet de personnaliser les différent exigences pour chaque type de trafic, assure une convergence rapide et contrairement à la solution centralisée, fonctionne mieux dans des réseaux à grande échelle.

En partant du fait que l'adresse MAC des éléments du réseau est unique, une technique de filtrage d'adresses MAC est pris en compte pour le fonctionnement du pare-feu au niveau des commutateurs SDN dans cette proposition(62). Ensuite, pour la gestion de la QoS les auteurs ont comparé les performances en terme de délai moyennes pour les techniques de filtrage IP et de filtrage MAC. Les résultats illustrent que la prise en charge de la sécurité (autorisation / refus de paquets) était à peu près la même dans les deux types de technique de pare-feu, tandis que le pare-feu avec filtrage MAC est plus performant du pare-feu de filtrage IP en terme de délai des paquets.

Dans (63), les auteurs proposent une procédure en deux phases pour obtenir des chemins disjoints de QoS k-max min combinés avec le processus de hiérarchie analytique dans un environnement SDN. La première phase utilise le processus de hiérarchie analytique pour saisir les diverses exigences de QoS et les réduire à une nouvelle fonction de coût (MCC : Multi_CriterionCost) qui est utilisée pour attribuer des pondérations de lien. La deuxième phase permet d'obtenir des parcours k-max min QoS disjoints pour l'équilibrer la charge du réseau central efficace, réduire la congestion et améliore la fiabilité. Les auteurs utilisent deux Algorithmes pour le calcul du chemin : a) l'algorithme de Dijkstra modifié pour obtenir un ensemble de chemins candidats entre source et destination, b) l'algorithme qui sélectionne les chemins k max min QoS disjoint à partir des chemins candidats produit par l'étape précédente à

l'aide de MCC restant minimum. La proposition est concevable dans Mininet et OpenFlow peut être utilisé pour gérer tous les flux tandis que la prise de décision et la configuration des règles seraient gérés par le contrôleur SDN.

Titre	Méthode	Objectifs	Observations
BaProbSDN	Calcule de la probabilité de la disponibilité de la bande passante.	Minimise les surcoûts associés au routage dans le réseau SDN tout en s'assurant que les garanties de QoS sont satisfaites.	▪ Les performances de l'algorithme ont été testées dans un environnement de simulation et comparées à l'algorithme de routage (WSR).
HiQoS	Une solution Hiqos basée sur la technologie Openflow. Crée et maintien plusieurs chemins pour une seule transmission et applique la différentiation des flux.	Hiqos vise à atteindre une qualité de service de bout en bout pour différents types d'applications telles que le streaming multimédia.	▪ Contrôleur : Floodlight. ▪ L'analyse des performances de la HiQoS est comparée à la technique de routage à voie unique (LiQoS) et au routage à voie unique avec les services différenciés (MiQoS).
QoS Guaranteed Network Resource Allocation via SDN	Un algorithme de routage QoS efficace en prenant en considération le niveau de congestion pour l'ensemble du chemin.	Assure la garantie de QoS de bout en bout de chaque service utilisateur dans le cloud.	▪ Utilise les générateurs de trafic basés sur Linux Iperf, et Netperf pour écrire un script Perl qui permet de capturer les caractéristiques de performance du trafic dans les expériences. ▪ Les résultats de cette approche sont comparés avec celles produits sans utiliser aucun mécanisme.
A Network Control Application enabling Software-Defined Quality of Service	Une application de contrôle des réseaux pour le provisionnement de la QoS.	Introduit un routage QoS centralisé et configure une procédure de planification à chaque port de commutateur pour un traitement du trafic	▪ Configure deux files d'attente avec une bande passante différente sur chaque port de tous les commutateurs:

		différencié.	▪ queue0: BE traffic, 20 Mbps; ▪ queue1: privileged traffic (EF), 10 Mbps. ▪ Cet article propose trois études de cas qui montrent le comportement de cette application de contrôle dans différentes situations.
Enhanced Multi-commodity Flow Model for QoS-aware Routing in SDN	Routage optimal basé sur des critères QoS et d'utilisation maximale/minimale des canaux.	Trouver l'ensemble optimal de routes à travers le réseau pour tous les flux avec un coût total minimal et en respectant certaines contraintes (délai maximum acceptable, la perte de paquets et bande passante disponible.*)*	▪ Utilise la programmation linéaire ▪ La topologie se compose de 7 commutateurs, qui sont interconnectés avec des liaisons de 1 Gbit /s de bande passante.
Model-Based Control Plane for Fast Routing in Industrial QoS Network	Un nouveau paradigme de contrôle de QoS maintient un graphe entre files d'attentes.	Garantie les contraintes QoS nécessaires à un flux	▪ MIP est une méthode basée sur la programmation linéaire. ▪ L'exécution de l'algorithme pour obtenir un itinéraire d'un nouvel flux est plus rapide que celle de MIP.
On QoS Management in SDN by Multipath Routing	Propose une méthode basée sur le routage à trajets multiples en décompose un flux sur multiple sous-flux.	Maximise l'utilisation des liens.	▪ Le schéma proposé fournit des services de réseau seuls au machines avec l'agent multi chemin préinstallé.
FlowQoS : QoS for the rest of us	Utilise la technologie SDN afin de proposer une QoS différenciée au flux au niveau du réseau domestique.	Effectue l'identification de l'application et utilise les règles de table de flux pour transmettre le trafic aux switchs virtuels installés dans le routeur du réseau domestique.	▪ Open vSwitch intégré dans le routeur doit supporter les fonctions de QoS basées sur le flux décrites dans OpenFlow 1.3.
A QoS-aware routing	SAQR est basé sur la collecte	Un routage QoS qui prend en	▪ Les résultats de comparaison

in SDN hybrid networks	de l'état du réseau afin de trouver le meilleur chemin qui répond à plusieurs exigences de QoS.	compte plusieurs contraintes QoS dans un réseau SDN hybride.	montrent que SAQR est plus performant que LARAC (qui ne considère qu'une seule contrainte) et MINA en matière de délai.
Virtualization-enabled Adaptive Routing for QoS-aware Software-Defined Networks	Met en œuvre une méthodologie de routage en temps réel qui prend en compte les différentes exigences de QoS.	QVR : acheminer les flux à travers des chemins qui répondent aux exigences de QoS correspondantes dans un fonctionnement centralisé. QRLR: le contrôleur est considéré comme un agent de décision qui apprend les comportements des flux entrants et prend des mesures pour améliorer son chemin à travers le réseau.	▪ Contrôleur SDN distribué fonctionne d'une manière plus efficace que le mode centralisé.
Investigation of Security and QoS on SDN Firewall Using MAC Filtering	Proposition d'une implémentation de pare-feu pour SDN utilisant le filtrage MAC.	Réduit la charge de travail du contrôleur centralisé ainsi que du réseau.	▪ Contrôleur : Floodlight. ▪ Utilise Putty et Xming ▪ Dans cette expérience, quatre hôtes et trois commutateurs SDN sont connectés physiquement au réseau.
Multi-Constraint QoS Disjoint Multipath Routing in SDN	Basé sur le routage à trajets multiples. Obtient K-max min QoS qui permet un équilibrage de charge efficace, réduit la latence, et améliore la fiabilité.	Permettre au réseau de répondre aux critères QoS avec les optimisations de performances des trajectoires.	▪ La proposition intègre deux techniques : AHP et Kmax min chemins disjoints comme des solutions prometteuses qui ont amélioré la performance globale du réseau.

Tableau II- 1 : Tableau comparatif entre les solutions citées

II.7 Conclusion

Ce chapitre décrit dans un premier temps les fondements de QoS et les différents types d'architectures pour prendre en charge l'approvisionnement QoS. Ensuite, on a défini les mécanismes de QoS qui gèrent le trafic sur le réseau. Enfin on a présenté travaux connexes sur le DiffServ et le traffic shaping dans le SDN, ils nous ont renseignés sur les divers types de techniques qui peuvent être utilisées pour mettre en oeuvre les systèmes de QoS dans le SDN, sur les divers défis auxquels fait face la communauté du SDN et nous a suggéré des solutions plausibles à certains des problèmes. Dans le chapitre suivant on va explorer les outils SDN utilisés pour l'implémentation et l'étude de performance de la QoS.

Chapitre III : Etude de performances de la gestion QoS dans un réseau SDN

III.1 Introduction

Les algorithmes de routage sont généralement classés en deux catégories : les algorithmes de vecteur de distance et les algorithmes d'état de liaison. La principale métrique utilisée par ces algorithmes pour calculer le chemin le plus court est le nombre de sauts ou le coût de la liaison définie en fonction de la bande passante (63).

Au début de ce chapitre, on présente les outils que nous avons utilisé pour cette implémentation ainsi que leurs utilités. En outre, nous introduisons les paramètres de simulations pour quantifier la différence des performances avec et sans prise en charge de la gestion de QoS à l'aide d'Open Flow. La méthode de gestion de QoS qu'on a utilisé pour mettre en évidence l'efficacité de la technologie SDN a été publiée dans (63). Enfin, les résultats obtenus sont interprétés.

III.2 Outils logiciels

- **Open vSwitch**

Open vSwitch est un commutateur virtuel multicouche de qualité production, sous licence open source Apache 2.0. Celui-ci peut être contrôlé via OF et qui est actuellement très populaire dans les environnements de serveurs virtuels et même comme élément de commutateurs matériels. Ce composant est conçu pour permettre l'automatisation massive du réseau via l'interface programmable, tout en prenant en charge les interfaces et protocoles de gestion standard. Si le transfert de données peut être contrôlé par OpenFlow, OVS présente aussi d'autres aspects qui ne relèvent pas du champ d'application d'OpenFlow. La configuration et les états/propriétés d'OVS sont conservés dans la base de données Open vSwitch (OVSDB), qui peut être interrogée et manipulée via le protocole OVSDB basé sur JSON (64). POX supporte le protocole OVSDB, qui permet de se connecter, de configurer et d'interroger des instances OVS.

- **Mininet**

Mininet (65) est un émulateur de réseau capable de créer un réseau d'hôtes, de commutateurs, de contrôleurs et de liens virtuels sur une seule machine. Les hôtes Mininet exécutent un logiciel de réseau Linux standard et ses commutateurs prennent en charge OF pour un routage personnalisé hautement flexible et une mise en réseau définie par logiciel. Mininet est caractérisé par sa flexibilité, interactivité et évolutivité. Principalement cet émulateur permet (65):

- ✓ Prototypage rapide de grands réseaux sur un seul ordinateur.

- ✓ Tests de topologie complexes sans avoir besoin de câbler un réseau physique.
- ✓ Collaboration et développement parallèle sur la même topologie.

Mininet fournit une API Python pour la création et l'expérimentation de réseaux. Celui-ci est publié sous une licence Open Source BSD (Berkeley Software Distribution) permissive, activement en développement et bénéficie du soutien de la communauté SDN (66).

Il fournit la capacité de créer des hôtes, les commutateurs et contrôleurs via :

- ✓ Ligne de commande.
- ✓ Interface interactive (Miniedit).
- ✓ Script Python.

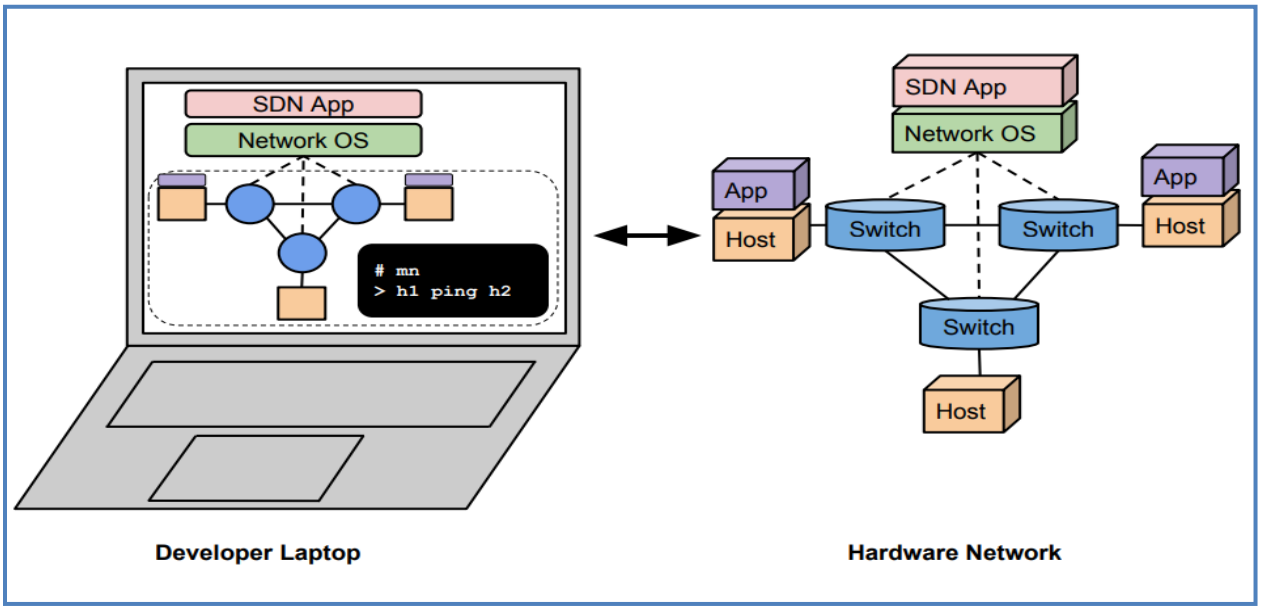


Figure III- 1 : Emulateur Mininet (66)

- **POX:**

POX est une plateforme pour le développement et le prototypage rapide de logiciels de contrôle de réseau en Python (67). Ce contrôleur fournit diverses API OpenFlow qui peuvent être utilisées pour interagir avec les commutateurs, extraire des informations statistiques des commutateurs et écrire des informations de contrôle sur les commutateurs (63). L'un des avantages de POX est sa facilité d'installation et d'utilisation. POX peut être utilisé avec l'interpréteur Python «standard» (CPython), mais prend également en charge PyPy (64). En outre, POX offre des implémentations prédéfinies pour des stratégies standards comme openflow.discovery, forwarding.l2_multi (66).

D-ITG:

Le Distributed Internet Traffic Generator (D-ITG) (68) est une plateforme capable de produire du trafic qui respecte précisément les paramètres introduits par l'utilisateur. De plus, le D-ITG intègre certains modèles pour émuler divers protocoles : TCP, UDP, ICMP, DNS, Telnet et VoIP. Le D-ITG peut mesurer pour chaque transmission générée par ce module : le délai unidirectionnel (OWD one-way-delay), temps d'aller-retour (RTT round-trip-time), la perte de paquets, la gigue et débit. De plus, le D-ITG permet de définir les valeurs des champs comme le TOS et TTL du paquet. Figure 2 montre un aperçu graphique de la relation entre les principales composants du module DITG.

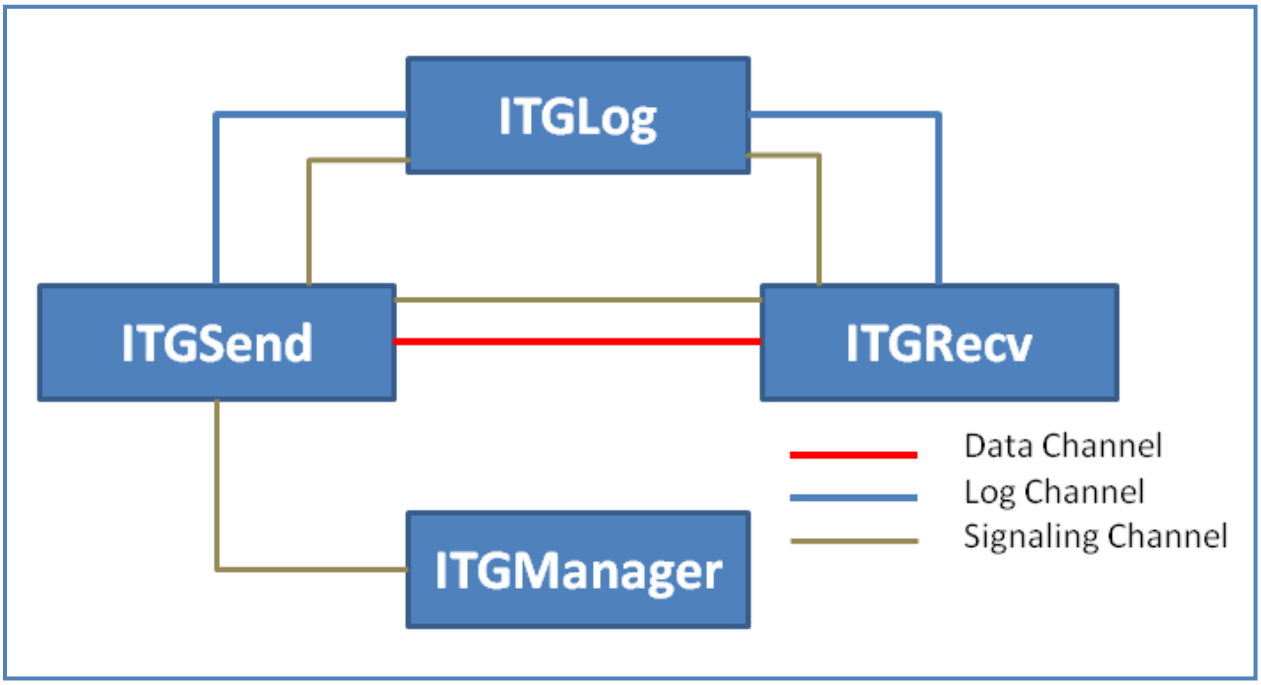


Figure III- 2 : Architecture de D-ITG (69)

La communication entre l'émetteur et le récepteur se fait en utilisant un canal de signalisation distinct et se contrôle par le protocole de configuration de l'expérience TSP (Traffic Specification Protocol).

Les trois principaux composants de l'architecture D-ITG est:

- ✓ **ITGSend** est le composant émetteur de la plate-forme de génération de trafic D-ITG.
- ✓ **ITGRecv** fonctionne toujours comme un démon concurrent, à l'écoute des nouvelles connexions à la couche traport.

- ✓ **ITGLog** est un serveur de journalisation, fonctionnant sur un hôte différent de ITGSend et ITGRecv, qui reçoit et stocke les informations de journalisation de plusieurs expéditeurs et récepteurs.

- **Xming :**

Xming offre la possibilité d'installer un serveur X sous Windows léger et performant. Le programme Xming est mieux connu sous le nom de serveur d'affichage qui permet notamment d'utiliser des applications X-Windows sur PC et Windows. En outre, Xming supporte un mode multi-fenêtres (70).

- **Putty :**

Putty est un programme permettant de se connecter à distance à des serveurs en utilisant les protocoles SSH(Secure Shell), Telnet ou Rlogin (71).

VMware Workstation:

VMware Workstation propose un outil de virtualisation de système d'exploitation (OS) sur une machine hôte. L'application s'appelle ici un hyperviseur et peut créer des machines virtuelles (VM) complètes avec gestion du son, de la vidéo, du réseau, de la quantité de RAM, des disques durs, des processeurs, etc. VMware Workstation peut capturer les périphériques USB et les intégrer à l'OS invité. Le logiciel autorise de lancer plusieurs VM simultanément, les mettre en pause pour les relancer plus tard depuis leurs l'état de suspension, etc. Enfin, VMware Workstation met à disposition des utilisateurs une connexion VNC (logiciel de partage d'écran) pour l'accès au bureau à distance du système invité et faciliter ainsi les démonstrations logicielles sur un OS propre (7).

- **Python :**

Python est un langage de programmation de haut niveau qui est largement utilisé actuellement. Sa philosophie de conception met l'accent sur la lisibilité du code, et sa syntaxe permet aux programmeurs d'exprimer des concepts dans moins de lignes de code par rapport à autres langages tels que C++ ou Java (72).

III.3 Expérimentations :

III.3.1 Topologie réseau :

La topologie a été implémentée sous l'émulateur Mininet en utilisant le scripte Python suivant :

```python
from mininet.topo import Topo
from mininet.link import TCLink

class MyTopo( Topo ):

    def __init__( self ):
        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        h1Host = self.addHost( 'h1' )
        h2Host = self.addHost( 'h2' )
        h3Host = self.addHost( 'h3' )
        h4Host = self.addHost( 'h4' )
        s1Switch = self.addSwitch( 's1' )
        s2Switch = self.addSwitch( 's2' )
        s3Switch = self.addSwitch( 's3' )
        s4Switch = self.addSwitch( 's4' )
        s5Switch = self.addSwitch( 's5' )

        # Add links
        self.addLink( h1Host, s1Switch )
        self.addLink( h2Host, s2Switch )
        self.addLink( h3Host, s3Switch )
        self.addLink( h4Host, s4Switch )
        self.addLink( s1Switch, s2Switch )
        self.addLink( s1Switch, s4Switch )
        self.addLink( s2Switch, s3Switch )
        self.addLink( s3Switch, s4Switch )
        self.addLink( s1Switch, s5Switch )
        self.addLink( s2Switch, s5Switch )
        self.addLink( s3Switch, s5Switch )
        self.addLink( s4Switch, s5Switch )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Figure III- 3 : Code source de la topologie de test

La figure 4 illustre la topologie réseau sur la quelle on souhaite mesurer les paramètres QoS.

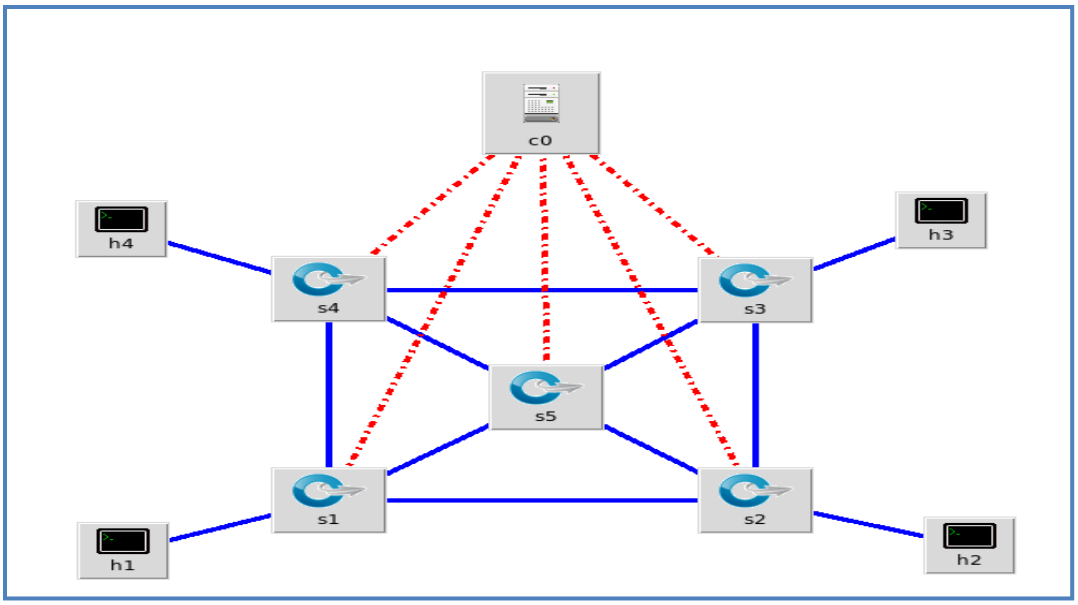


Figure III- 4 : Diagramme de topologie

La commande à exécuter sur Mininet pour créer cette topologie à partir du scripte Python et la suivante :

sudo mn --custom customtopo.py --topo mytopo --link tc , bw=1000 --controller remote, ip=192.168.1.6

- L'option --custom customtopo.py --topo mytopo permet d'utiliser la topologie personnalisée définie dans customtopo.py avec le classe mytopo.
- L'option --link=tc permet de spécifier les caractéristiques des liens (bandwidth, delay, loss). Dans notre cas, on a défini la bande passante de tous les liens à 1Gb/s « --link tc,bw=100 »
- L'option --controller remote permet à Mininet de connecter les commutateurs avec un contrôleur externe. Par défaut, l'adresse et le numéro de port du contrôleur prennent les valeurs {172.0.0.1 :6633}.

Par ailleurs, le contrôleur POX doit toujours être lancé sur une machine virtuelle avant mininet pour qu'il puisse fonctionner. Le tableau 1 montre les DPID (Data Path) des commutateurs ainsi que les adresses IP des hôtes.

Switch	DPID
Switch_1	00-00-00-00-00-01
Switch_2	00-00-00-00-00-02
Switch_3	00-00-00-00-00-03
Switch_4	00-00-00-00-00-04
Switch_5	00-00-00-00-00-05

Hôte	Adresse IP
Host_1	10.0.0.1
Host_2	10.0.0.2
Host_3	10.0.0.3
Host_4	10.0.0.4

Tableau III- 1 : Switches et hôtes

III.3.2 Paramètres des transmissions

Dans ce qui suit, on va comparer deux approches de découverte de route optimale. La première méthode permet de trouver le plus court chemin entre une source et une destination sans prendre en considération les exigences de QoS. Par contre, la deuxième approche tient compte des paramètres de QoS lors du choix de la route. On mesure pendant les simulations deux métriques de QoS : le délai et le débit. En outre, l'étude est répartie en deux scénarios. Pour chaque scéario, deux transmissions sont lancées parallèlement dans le réseau avec un taux de transmission ordinaire (allons de 100 à 280 pkts/s) et dans un deuxième scénario avec un trafic intense (allons de 500 à 5000 pkts/s).

Pour l'expérimentation sans prise en charge de la QoS, on lance le contrôleur POX par le composant de base l2_multi. Ce composant (64) permet aux commutateurs OVS d'apprendre la topologie comme s'ils sont des commutateurs L2 autonomes qui apprennent les adresses MAC et choisissent les chemins les plus courts grâce à l'algorithme Floyd-Warshall. De plus, ce composant fonctionne avec le composant openflow.spanning_tree (73).

Dans l'approche SDN de gestion de la QoS (63), un schéma d'acheminement est appliqué basé sur les exigences de QoS du trafic. Cette méthode prend en considération la latence des liens, la bande passante et les pertes des paquets sur les files d'attentes comme paramètres de QoS pour sélectionner le meilleur chemin en fonction de la classe du service. La figure 5 illustre le digramme de flux dans cette approche.

III.3.3 Mécanisme de gestion de QoS avec SDN:

La latence du réseau représente le temps nécessaire à un paquet pour passer de la source à la destination. A la base de la méthode décrite dans le document "Monitoring latency with OpenFlow" (74) , la latence des liens est calculée. A noté que le retard subis par un paquet est principalement la conséquence du placement dans les files d'attentes des commutateurs.

Pour calculer la latence de chaque lien, le contrôleur génère un paquet spécial identifiable par une valeur (0x07c3) comme étant de type Ethernet et avec une adresse destination multicast (le protocole NDP Neighbor Discovery Protocol). La charge utile (payload) comprend l'adresse MAC du commutateur, le port d'où il doit être envoyé et l'instant de sa création (Time Stamp). Initialement, lorsque les commutateurs sont allumés, le contrôleur écrit des entrées dans les tables de flux commutateurs pour renvoyer les paquets de type Ethernet 0x07c3 au contrôleur via un message PACKET OUT. Par la suite, ce paquet spécial est envoyé depuis un contrôleur et inversement tout en mesurant le temps nécessaire pour la transition (63). En outre, le RTT entre le contrôleur et les commutateurs est calculé de la même façon. Le calcul de la latence de lien se fait comme suit :

$$T_{link(s1,s2)} = T_{total} - [(T_{s1} + T_{s2})/2] \quad \text{(Équation 1)}$$

a. T_{total} - le temps total
b. T_{s1}- le RTT entre le contrôleur et s1
c. T_{s2}- le RTT entre le contrôleur et s2

Afin de gérer la QoS proposée aux applications, le contrôleur SDN utilise une unité de sondage des liens qui interroge les commutateurs et enregistre les paramètres de liaison tels que

la gigue et les pertes de paquets dans la base de données d'état des liens (LSDB). Cette unité interroge le réseau périodiquement ou en cas d'interruption par le changement d'état de la liaison ou du nœud. Par la suite un deuxième module est employé pour calculer l'indice de QoS. Celui-ci utilise une fonction de coût basée sur les paramètres de liaison sauvegardés dans la LSDB pour calculer l'indice de QoS pour une liaison. Ces informations sont transmises au module Path Finder pour déterminer le chemin optimal.

Calculateur de l'indice de QoS = K1 * Utilisation de la bande passante + N (K2 * latence + K3 * perte dans file d'attente) Équation 2

Le calculateur de l'indice de QoS calcule le coût de chaque liaison en donnant des poids différents K1, K2 et K3 selon la classe du trafic. Pour la valeur de N elle est égale à 0 dans le cas où le trafic Best Effort et égale à 1 dans le cas d'autres types de trafic, K1 est fixé égale 100.

- **Trafic vocal** : Le trafic vocal est sensible à la latence et moins tolérant à la perte de données. Ainsi, lors du calcule d'un chemin pour un trafic de cette classe, le coéfficient K2 a une valeur plus élevée alors que K3 a un poids moins élevé.
- **Trafic vidéo** : Le trafic vidéo n'est pas très sensible à la latence ou à la perte de données. Ainsi, le K2 et le K3 auront un poids moyen.
- **Trafic critique** : Le trafic commercial ne tolère pas le délai mais est très sensible à la perte de données. Ainsi ce trafic est caractérisé par une valeur de K3 élevée et un poids moins élevé pour la latence.

Finalement, le module Path Finder recherche un chemin optimal en exécutant l'algorithme Floyd-Warshall pour chaque classe de trafic en se basant sur les entrés du calculateur de l'indice de QoS. La route optimale est calculée de manière périodique.

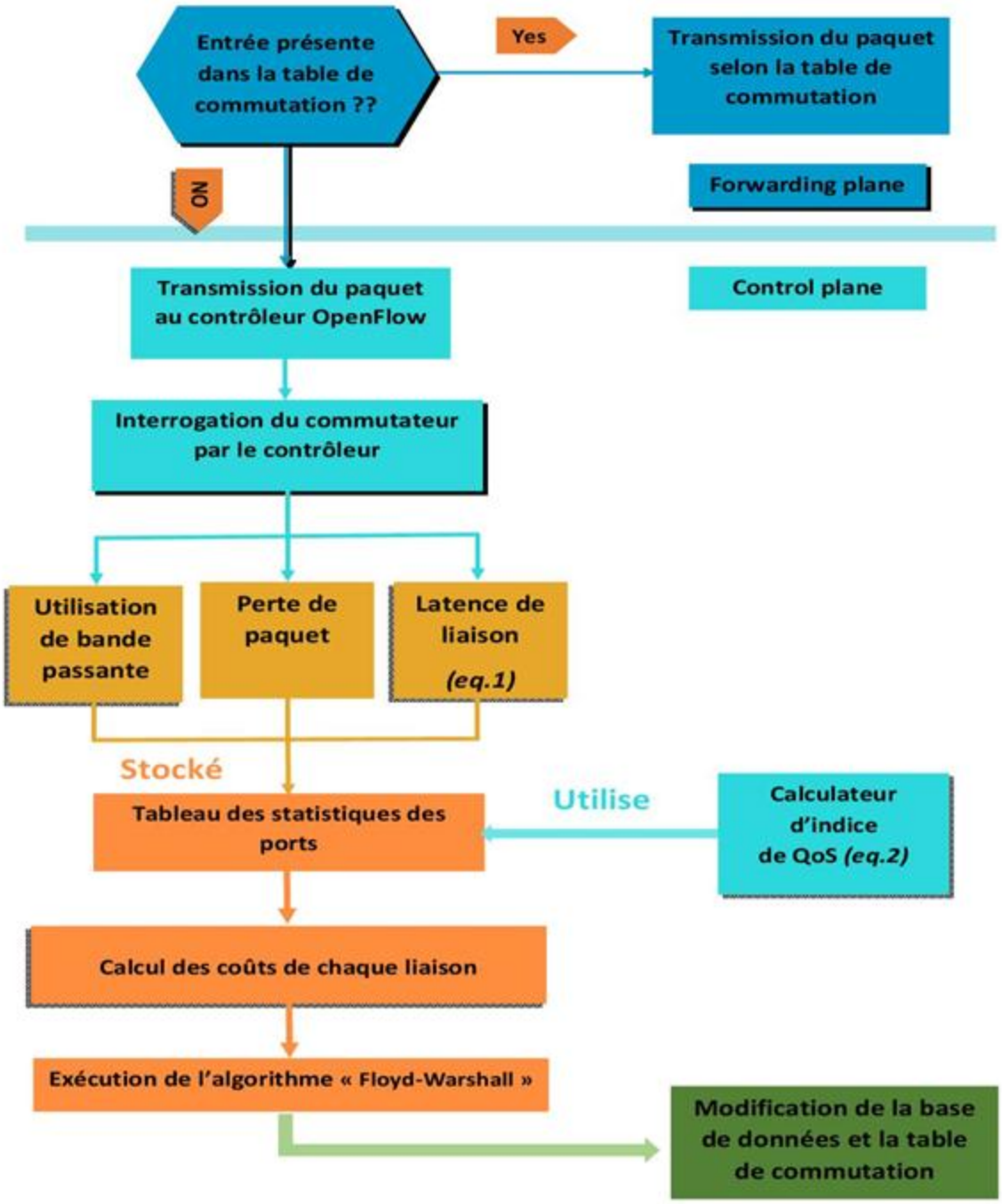


Figure III- 5 : Démarche de découverte de la route optimale avec prise en charge de la QoS

III.3.4 Mise en œuvre de l'expérimentation

Pour comparer les deux approches de découverte de route optimale avec et sans politique de QoS, on a utilisé le D-ITG générateur des flux de trafic pour lancer des transmissions simultanément entre les hôtes dans le réseau avec deux scénarios :

- ✓ **Scénario° 1 :** trafic ordinaire avec 100 à 280 paquets par seconde ;
- ✓ **Scénario° 2 :** trafic intense avec 500 à 5000 paquets par seconde ;

Afin de générer du trafic avec D-ITG, il suffit d'exécuter une instance de ITGRecv dans chaque récepteur et une instance ITGSend pour chaque transmetteur. Les étapes suivantes permettent de lancer la simulation sans gestion de QoS :

1. On a démarré le contrôleur POX avec **l2_multi** avec la commande suivante:

./pox.py forwarding.l2_multi openflow.discovery --eat-early-packets openflow.spanning_tree --no-flood --hold-down

- --eat-early-packets l'idée est de s'assurer qu'il n'effectue aucune transmission tant que la découverte de la topologie n'est pas accomplie.
- --no-flood désactive l'inondation sur tous les ports dès qu'un commutateur se connecte ; sur certains ports, il sera activé plus tard.
- --hold-down empêche l'altération du contrôle des inondations jusqu'à ce qu'un cycle complet de découverte soit terminé.

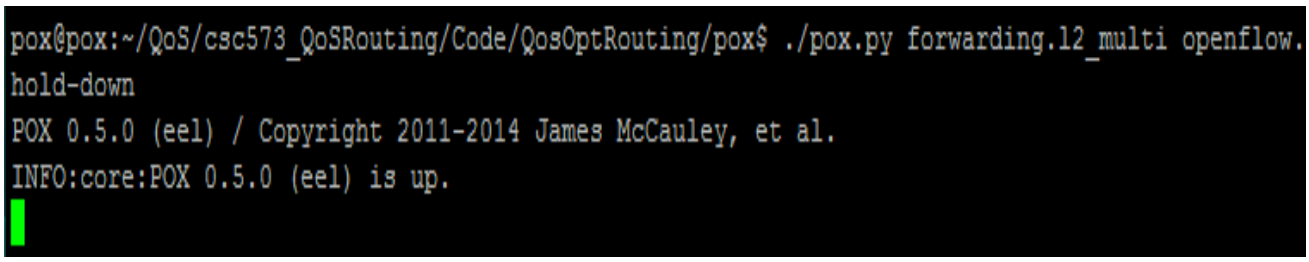


Figure III- 6 : Le résultat de l'exécution *de la commande pour lancé le composant l2_multi*

2. On a démarré Mininet dans un autre terminal à l'aide de la commande :

sudo mn --custom customtopo.py --topo mytopo --link tc , bw=1000 --controller remote, ip=192.168.1.6

```
pox@pox:~/QoS/csc573_QoSRouting/Code/QosOptRouting$ sudo mn --custom  customtopo.py --top
o mytopo --link tc,bw=1000  --controller remote,ip=192.168.1.8
*** Creating network
*** Adding controller
Unable to contact the remote controller at 192.168.1.8:6653
Connecting to remote controller at 192.168.1.8:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(1000.00Mbit) (1000.00Mbit) (h1, s1) (1000.00Mbit) (1000.00Mbit) (h2, s2) (1000.00Mbit) (
1000.00Mbit) (h3, s3) (1000.00Mbit) (1000.00Mbit) (h4, s4) (1000.00Mbit) (1000.00Mbit) (s
1, s2) (1000.00Mbit) (1000.00Mbit) (s1, s4) (1000.00Mbit) (1000.00Mbit) (s1, s5) (1000.00
Mbit) (1000.00Mbit) (s2, s3) (1000.00Mbit) (1000.00Mbit) (s2, s5) (1000.00Mbit) (1000.00M
bit) (s3, s4) (1000.00Mbit) (1000.00Mbit) (s3, s5) (1000.00Mbit) (1000.00Mbit) (s4, s5)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 5 switches
s1 s2 s3 s4 s5 ...(1000.00Mbit) (1000.00Mbit) (1000.00Mbit) (1000.00Mbit) (1000.00Mbit) (
1000.00Mbit) (1000.00Mbit) (1000.00Mbit) (1000.00Mbit) (1000.00Mbit) (1000.00Mbit) (1000.
00Mbit) (1000.00Mbit) (1000.00Mbit) (1000.00Mbit) (1000.00Mbit) (1000.00Mbit) (1000.00Mbi
t) (1000.00Mbit) (1000.00Mbit)
*** Starting CLI:
mininet>
```

Figure III- 7 : Le résultat de l'exécution de la commande pour lancé la topologie dans Mininet

```
pox@pox:~/QoS/csc573_QoSRouting/Code/QosOptRouting/pox$ ./pox.py forwarding.l2_multi
hold-down
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-05 5] connected
INFO:openflow.of_01:[00-00-00-00-00-02 6] connected
INFO:openflow.of_01:[00-00-00-00-00-03 4] connected
INFO:openflow.of_01:[00-00-00-00-00-01 3] connected
INFO:openflow.of_01:[00-00-00-00-00-04 2] connected
INFO:openflow.discovery:link detected: 00-00-00-00-00-05.4 -> 00-00-00-00-00-04.4
INFO:openflow.discovery:link detected: 00-00-00-00-00-05.1 -> 00-00-00-00-00-01.4
INFO:openflow.discovery:link detected: 00-00-00-00-00-05.2 -> 00-00-00-00-00-02.4
INFO:openflow.discovery:link detected: 00-00-00-00-00-05.3 -> 00-00-00-00-00-03.4
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.4 -> 00-00-00-00-00-05.2
```

Figure III- 8 : Le résultat dans le POX après le lancement de la topologie dans Mininet

3. Dans la console de Mininet, on a lancé la commande suivante pour ouvrir des terminaux sur les hôtes :

   xterm h1 h2 h3 h4

4. On a lancé le récepteur sur les deux hôtes de destinations h2 h3 (10.0.0.2 / 10.0.0.3) à l'aide de la commande suivante :

   ITGRecv

5. On lance l'émetteur dans les deux hôtes sources h1 h4 (10.0.0.1 / 10.0.0.4) simultanément à l'aide de deux commandes suivantes :

Terminal de h1 : ITGSend –T UDP -a 10.0.0.3 -rp 10001 -C 100 –c 1400 -b 160 -x recv_log_file

Figure III- 9 : Commande ITGSend dans h1

Terminal de h4 : ITGSend –T UDP-a 10.0.0.2 -rp 10001 -C 100 –c 1400 -b 160 -x recv_log_file

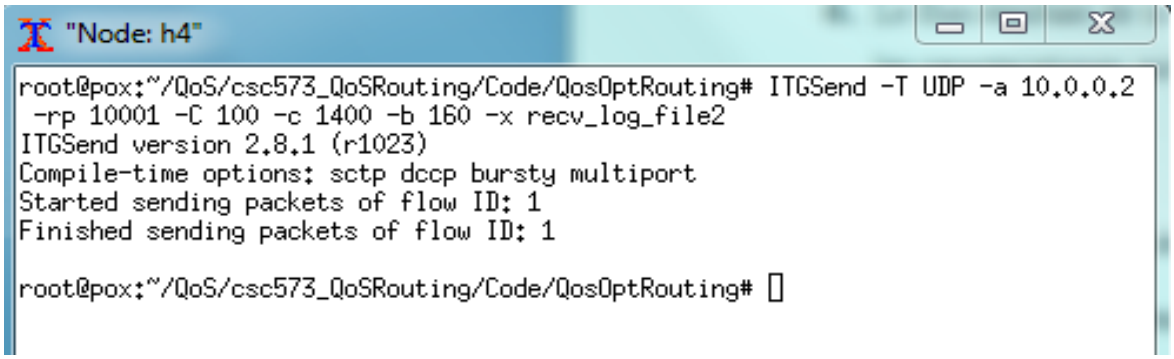


Figure III- 10 : Commande ITGSend dans h4

6. Le flux résultant de h1 à h3 (10.0.0.1 à 10.0.0.3) et h4 à h2 (10.0.0.4 à 10.0.0.2) présente les caractéristiques suivantes :

   - Le trafic est encapsulé avec UDP (User Datagram Protocole) ;
   - Le port émetteur est choisi par défaut ;
   - Le port de destination est le 10001;
   - 100 paquets par seconde sont envoyés avec un temps constant entre chaque deux paquets successives ;
   - La taille de chaque paquet est égale à 1400 octets ;
   - La durée de la simulation est de 10 secondes (10 000 millisecondes) ;
   - Du côté du destinataire, ITGRecv crée un fichier journal nommé recv_log_file ;
   - La classe de trafic est définie avec le champ TOS égale à 160 qui est synonyme de « trafic vocal ».

7. Le fichier recv_log_file2 est décodé pour chaque récepteur (h2 et h3). Le résultat est une liste d'indices de performance comme illustré dans figure 11.

8. On prend les valeurs du délai et débit depuis les résultats obtenus des deux transmissions et on calcule la moyenne entre les deux trafics.

```
root@pox:~/QoS/csc573_QoSRouting/Code/QosOptRouting# ITGDec recv_log_file2
ITGDec version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
/------------------------------------------------------------
Flow number: 1
From 10.0.0.4:56752
To    10.0.0.2:10001
------------------------------------------------------------
Total time                =      9.968884 s
Total packets             =           904
Minimum delay             =      0.000039 s
Maximum delay             =      0.058454 s
Average delay             =      0.000511 s
Average jitter            =      0.000550 s
Delay standard deviation  =      0.002807 s
Bytes received            =       1265600
Average bitrate           =   1015.640266 Kbit/s
Average packet rate       =     90.682167 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =      0.000000 pkt
------------------------------------------------------------
```

Figure III- 11 : Décodage de fichier recv_log_file2 sur la machine h2

```
root@pox:~/QoS/csc573_QoSRouting/Code/QosOptRouting# ITGDec recv_log_file3
ITGDec version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
------------------------------------------------------------
Flow number: 1
From 10.0.0.1:45059
To    10.0.0.3:10001
------------------------------------------------------------
Total time                =      9.983171 s
Total packets             =           903
Minimum delay             =      0.000045 s
Maximum delay             =      0.053607 s
Average delay             =      0.000438 s
Average jitter            =      0.000510 s
Delay standard deviation  =      0.002104 s
Bytes received            =       1264200
Average bitrate           =   1013.064887 Kbit/s
Average packet rate       =     90.452222 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =      0.000000 pkt
------------------------------------------------------------
```

Figure III- 12 : Décodage de fichier recv_log_file3 sur la machine h3

Pour la simulation du contrôleur prenant compte de la gestion de QoS, il suffit de suivre la même démarche précédante avec le composant *QoS Routing* **:**

./pox.py log.level --ERROR QoSOptRouting

```
pox@pox:~/QoS/csc573_QoSRouting/Code/QosOptRouting/pox$ ./pox.py log.level --ERROR QoSOptRouting
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
 Routage optimal basee sur Qos utilisant OpenFlow
```

Figure III- 13 : Le résultat de l'exécution de la commande pour lancé QoSOptRouting

III.3.5 **Analyse des résultats:**

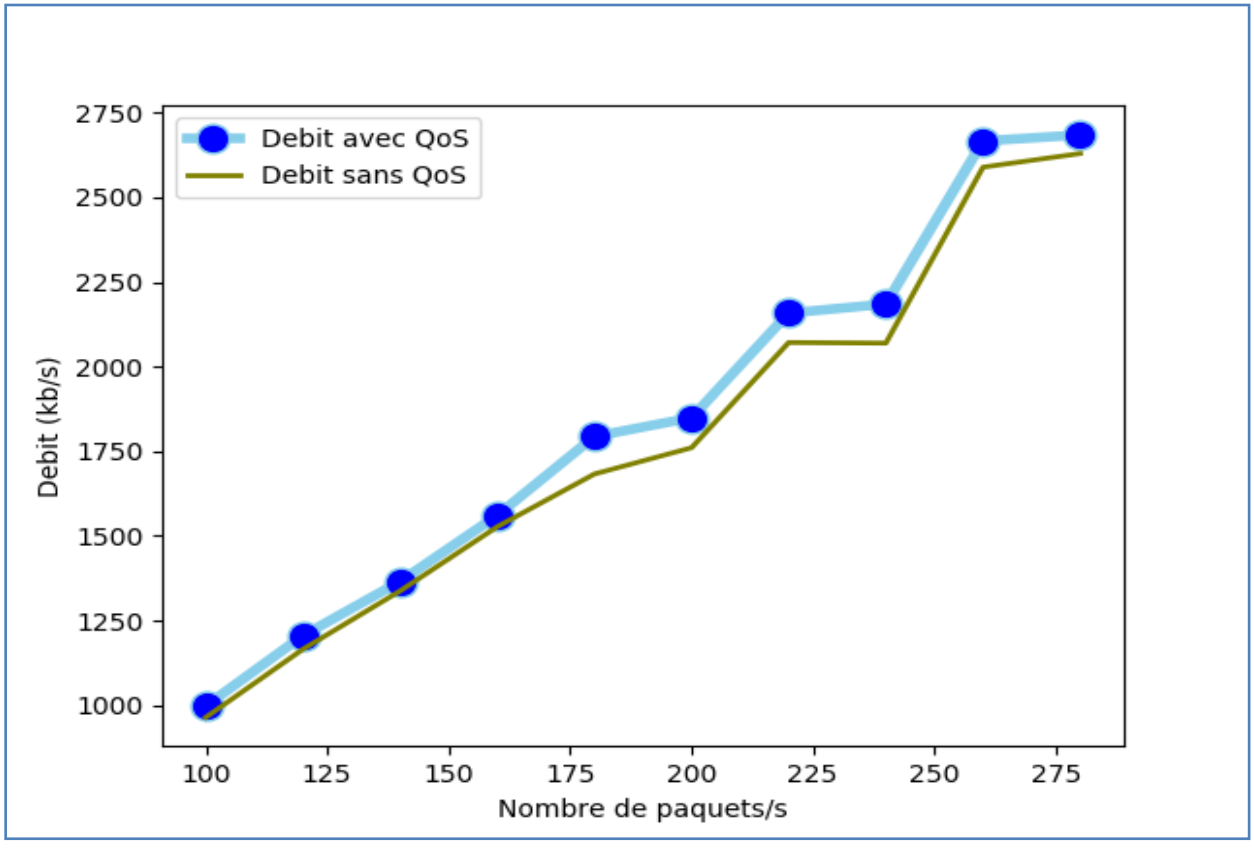


Figure III- 14 : Graphes de débit avec et sans gestion de QoS

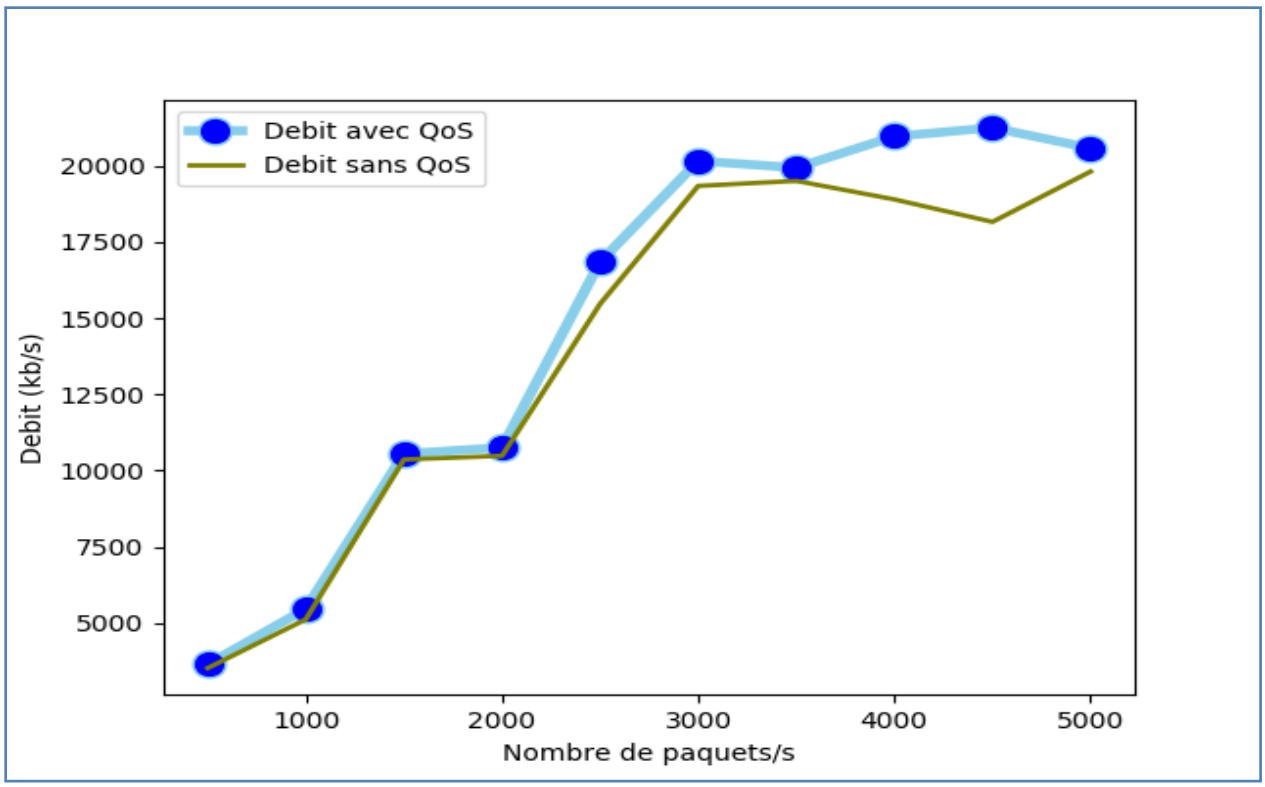


Figure III- 15 : Graphes de débit avec et sans gestion de QoS

Les résultats dans les figures 14 et 15 représentent l'évolution de débit avec et sans gestion de QoS pour les deux scénarios (trafic ordinaire et trafic intense respectivement). Il est percevable que le débit augmente puisque le taux de transmission accroît des deux machines h1 et h4. Par contre, on remarque que le débit avec gestion QoS est plus élèvé par rapport au débit sans QoS même ci cette différence est réduite entre les deux configurations. En effet, cette différence est encore plus claire à partir de 3500 paquets par seconde dans la figure 15 (trafic intense). Cela implique qu'il y a une augmentation de pertes de paquets dans l'approche de découverte de route sans QoS. La raison derrière cette perte de paquets est la saturation des files d'attente dans lesquels les paquets sont placés pendant leurs transitions.

Les résultats dans figures 16 et 17 représentent l'évolution du délai avec et sans gestion de QoS pour les deux scénarios. Dans figure 16, on remarque qu'il y a une grande différence entre de délai. Le délai avec QoS ne dépasse pas 0.27 ms alors que le délai sans QoS augmente jusqu'à 0.6 ms.

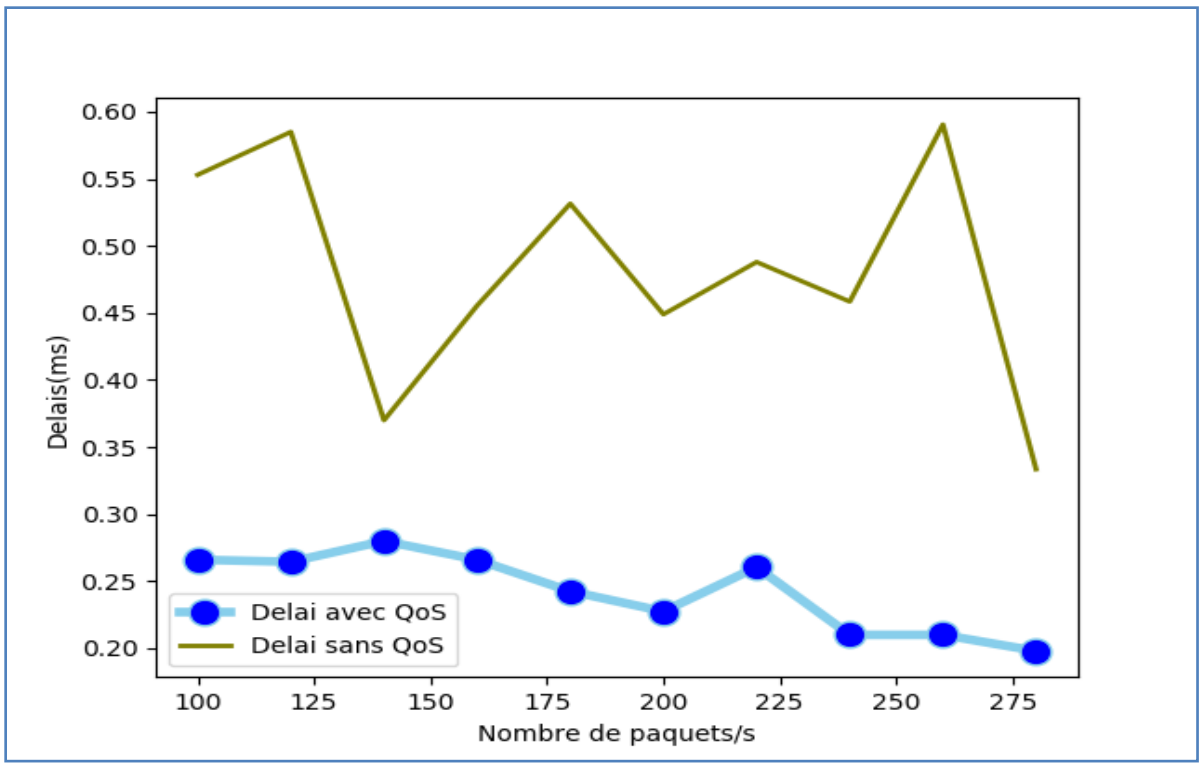


Figure III- 16 : Graphes de délai avec et sans QoS par rapport au trafic ordinaire

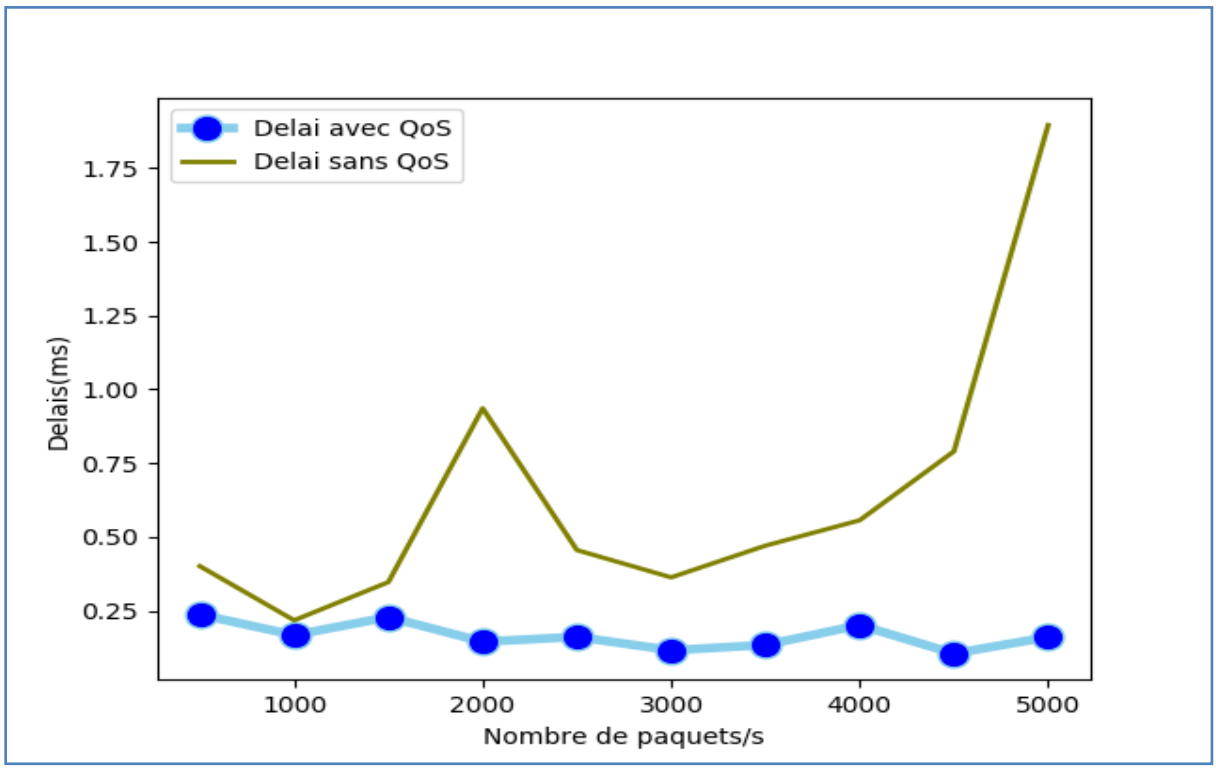


Figure III- 17 : Graphes de délai avec et sans QoS par rapport au trafic intense

Dans la figure 17, on remarque que la différence entre les délais obtenus augmente significativement. Avec la gestion de QoS, les valeurs du délai restent proches à 0.25 ms malgré l'augmentation de taux de transmission jusqu'à 5000. Par contre, le délai sans QoS varie de 0.25 ms à 1 ms dans les transmissions de 1000 jusqu'à 3000 pkts/s. En outre, pour plus de 3000 pkts/s on remarque que les valeurs de délai augmentent jusqu'à 2 ms. Cela implique que les paquets arrivent à la destination mais restent dans les files d'attente pour une plus longue durée car le trafic est transmis à travers des files d'attente qui sont remplies par d'autres paquets en attente de leurs tours.

Les résultats des simulations montrent que dans les deux scénarios l'algorithme de routage avec QoS est plus performant par rapport à l'algorithme de routage sans QoS en matière de délai et débit. De ce fait, pour que l'algorithme de routage avec QoS sélectionne un chemin optimal qui satisfait les exigences de QoS. Selon la classe de trafic, celui-ci traite l'état de chaque liaison grâce aux trois métriques de QoS pour calculer les coûts des liens. Par la suite, le meilleur chemin est choisi. Donc, cet algorithme évite des liens qui sont saturés ou bien qui sont déjà occupés par d'autres transmissions de trafic.

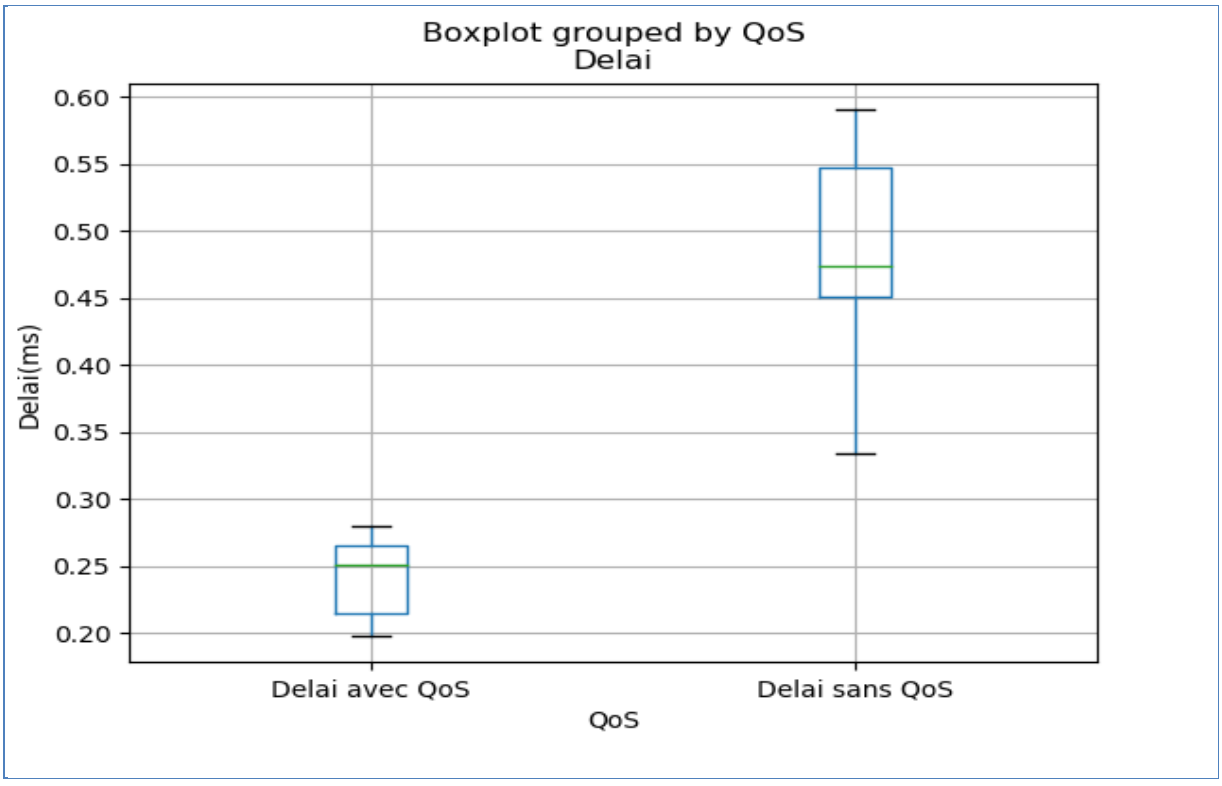


Figure III- 18 : Variation de délai dans le trafic ordinaire 100 – 280 paquets par seconde

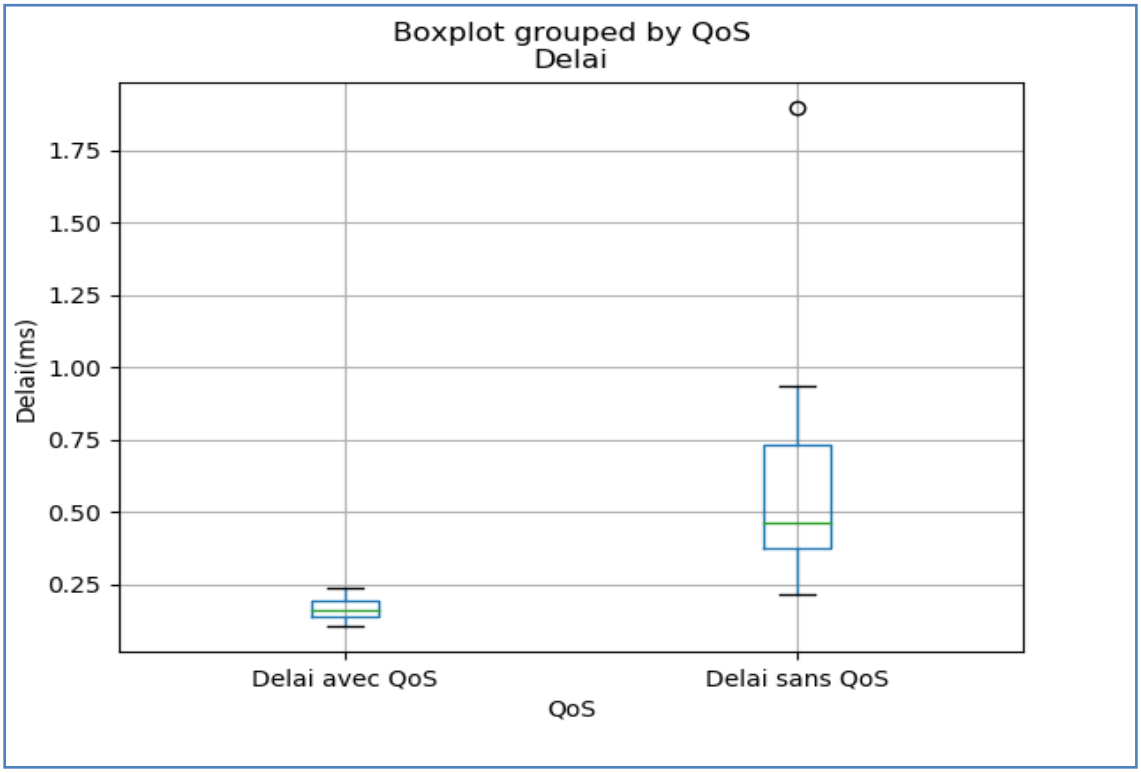


Figure III- 19 : Variation de délai dans le trafic intense 500 – 5000 paquets par seconde

Les figures 18 et 19 présentent les variations des délais produits par les scénarios. Les résultats montrent que les délais varient largement avec l'augmentation de l'intensité du trafic lorsque la gestion de QoS n'est pas appliquée. Par contre, avec la gestion de la QoS, le délai varie très peu d'un scénario à un autre.

III.4 Conclusion

Dans ce chapitre, on a commencé par présenter les outils que nous avons utilisés pour cette étude. Par la suite, on a présenté la topologie réseau simulée ainsi que les paramètres des transmissions. La comparaison des résultats illustre l'efficacité du routage avec gestion de QoS. La technique implémentée offre l'avantage d'être basée sur la technologie SDN qui rend l'approche plus flexible et adaptable.

Conclusion générale

La réalisation de ce projet de fin d'études a été motivée par l'évolution rapide du SDN, qui prétend à révolutionner l'architecture actuelle des réseaux. Ainsi, le SDN est devenu une solution valable dans divers domaines modernes notamment dans le réseau 5G, le WAN d'entreprise pour le routage intelligent des flux IP des différentes agences de l'entreprise, le Cloud Computing et Data Center modernes. En effet, la technologie SDN permet de programmer le comportement du réseau de manière centralisée grâce à des API ouvertes.

On a d'abord présenté l'entreprise Algérie Télécom, sa structure ainsi que les difficultés rencontrées par celle-ci dans la gestion du réseau traditionnel. Ensuite, on a introduit le réseau SDN en donnant ses avantages et ses défis dans le milieu d'entreprise. La mise en œuvre de la QoS dans le SDN est plus facile et moins coûteuse que sur les réseaux traditionnels. En outre, il est possible d'implémenter des techniques plus complexes de QoS comme DiffServ/Intserv alors que ces dernieres sont plus complexes à implémenter dans les réseaux IP traditionnels. Enfin les résultats obtenus par la stratégie de gestion de QoS sont très satisfaisants. Dans des futurs travaux, nous souhaitons explorer les perspectives suivantes :

- Mener une comparaison empirique entre les différents contrôleurs SDN actuellement disponibles tel que RYU.
- Implémenter une politique d'économie d'énergie basée sur la technologie SDN.

Annexe

Exécutez les commandes suivantes à partir du terminal Ubuntu afin d'installer Mininet et ensuite POX sur Ubuntu :

- ✓ **Mininet :**

  $ mkdir mininet

  $ cd mininet

  $ install git par : sudo apt-get install git

  $ git clone https://github.com/mininet/mininet .

  $ cd util/

  $ ./install.sh -a

- ✓ **POX :**

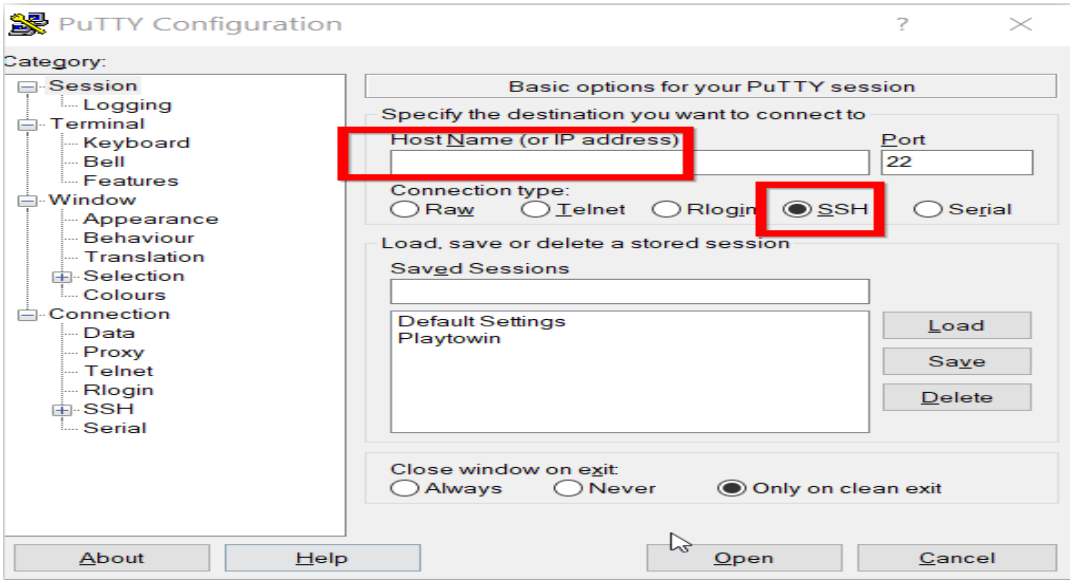
  $ mkdir pox

  $ cd pox

  $ git clone https://github.com/noxrepo/pox

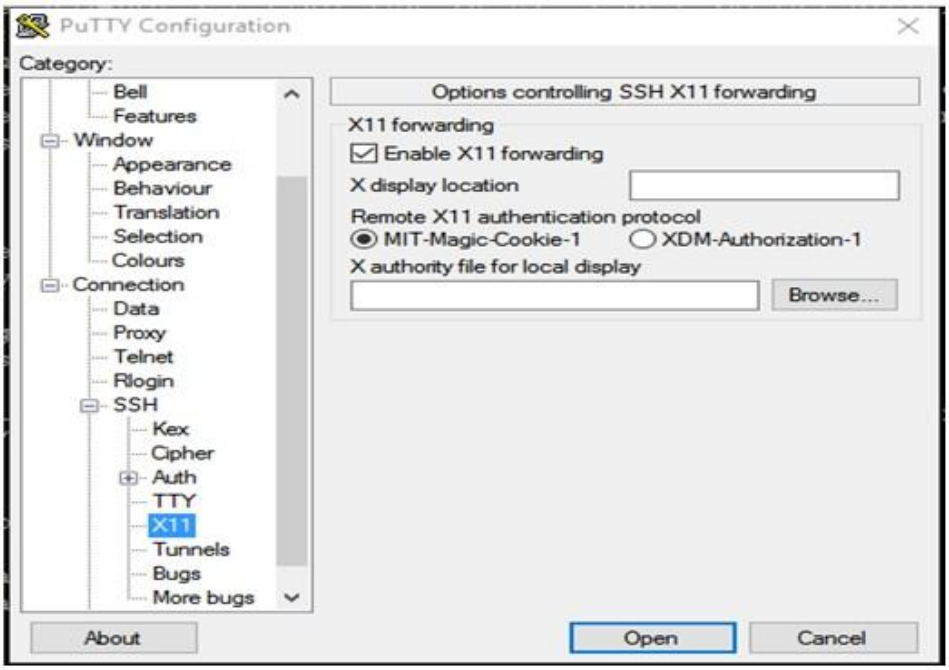
Pour installer Putty et Xming :

1. Exécutez **putty** et **xming** dans Windows.
2. Remplissez l'adresse Ip du machine virtuelle **POX** préinstalle. Le port doit être 22 et assurez-vous que SSH est sélectionné.



Configuration Putty

3. Pour utiliser des applications graphiques, activez la redirection **X11** : sous Connection->SSH->X11, cochez Enable X11 forwarding..
4. Cliquez ensuite sur open.



Configuration de l'affichage avec X11

Bibliographie

1. Kucminski, Andrew et al. QoS-based routing over software defined networks. IEEE Int Symp Broadband Multimed Syst Broadcast (BMSB)IEEE. 2017;

4. Germain M. Introduction au réseaux. Forum ATENA; 2012. 1-17 p.

12. Göransson P, Black C. Software Defined Networks A Comprehensive Approach. Mark Roger. Elliot S, éditeur. 2014.

13. Babu R. Dawadi , Danda B. Rawat SRJ. Software Defined IPv6 Network: A New Paradigm for Future Networking. J Inst Eng. 2019;15(2):1-13.

15. Cox JH, Chung J, Donovan S, Ivey J, Clark RJ, Riley G, et al. Advancing software-defined networks: A survey. IEEE Access. 2017;5:25487-526.

16. M Smith, M Dvorkin YL et al. OpFlex control protocol. IETF. 2014;

17. Bianchi, Giuseppe et al. Openstate: Programming platform-independent stateful openflow applications inside the switch. ACM SIGCOMM Comput Commun Rev. 2014;44-51.

18. Shaghaghi, Arash MAK +et al. Software-Defined Network (SDN) Data Plane Security: Issues, Solutions, and Future Directions. Clust Comput J. 2018;24.

21. Kreutz D, Ramos FMV, Verissimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: A comprehensive survey. Proc IEEE. 2014;103(1):14-76.

24. Dong X, Lin H, Tan R, Iyer RK, Kalbarczyk Z. Software-defined networking for smart grid resilience: Opportunities and challenges. Proc 1st ACM Work Cyber-Physical Syst Secur. 2015;61-8.

25. Kaivola R. Practical Aspects of Declarative Languages. Hutchison D, Kanade T, Kittler J, Kleinberg JM, éditeurs. Vol. 6539, 13th International Symposium, PADL 2011, Austin, TX, USA, January 24-25, 2011. Proceedings. Austin, TX, USA; 2011.

26. Monsanto C, Foster N, Harrison R, Walker D. A compiler and run-time system for network programming languages. In Philadelphia, PA, USA: ACM; 2012. p. 217-30.

28. Monsanto, Christopher et al. Composing Software-Defined Networks. 10th {USENIX} Symp Networked Syst Des Implement ({NSDI} 13). 2013;

29. Benamrane F, Mamoun M Ben, Redouane B. Etude des Performances des Architectures du Plan de Contrôle des Réseaux Software -Defined Networks. 2017.

31. Le Protocole OpenFlow dans l ’ Architecture SDN (Software Defined Network). 2016.

32. Anis A, Yanis A. Etude et implémentation d’une architecture SDN LAN. Mouloud MAMMERI DE TIZI- OUZOU; 2018.

33. Fernandez MP. Comparing OpenFlow controller paradigms scalability: Reactive and proactive. 2013 IEEE 27th Int Conf Adv Inf Netw Appl (AINA) IEEE. 2013;1009-16.

34. Lara, Adrian, Anisha Kolasani and BR. Network innovation using open flow: A survey. IEEE Commun Surv tutorials. 2014;16(1):493-512.

35. Ongaro F. ENHANCING QUALITY OF SERVICE IN SOFTWARE-DEFINED NETWORKS. Diss Alma Mater Stud Bol. 2014;

38. Chemeritskiy, Eugene and RS. On QoS management in SDN by multipath routing. 2014 Int Sci Technol Conf (Modern Netw Technol. 2014;

41. Shenker S. « Integrated Services in the Internet Architecture: an Overview Status of this Memo ». 1994; Disponible sur: https://dl.acm.org/doi/pdf/10.17487/RFC1633

44. Mehdi C, Adel D. Étude et implémentation de l’ approche Software Defined Network dans. 2018.

46. Villapol ME, Billington J. Internet service quality: A survey and comparison of the IETF approaches. Telecommun J Aust [Internet]. 2000;50(2):57-68. Disponible sur: http://www.ciens.ucv.ve/mvillap/Papers/TJApaperD.pdf

51. Al-Jawad A, Trestian R, Shah P, Gemikonakli O. BaProbSDN: A probabilistic-based QoS routing mechanism for Software Defined Networks. Proc 2015 1st IEEE Conf Netw Softwarization (NetSoft)IEEE. 2015;

52. Yan, Jinyao et al. HiQoS: An SDN-based multipath QoS solution. China Commun. 2015;12(5):123-33.

53. Zhang, Hailong et al. Sdn-based ecmp algorithm for data center networks. 2014 IEEE Comput Commun IT Appl Conf IEEE. 2014;

54. Akella, Anand V. and KX. Quality of service (QoS)-guaranteed network resource allocation via software defined networking (SDN). Proc - 2014 World Ubiquitous Sci Congr 2014 IEEE 12th Int Conf Dependable, Auton Secur Comput DASCIEEE. 2014;7-13.

55. Adami, Davide et al. A network control application enabling Software-Defined Quality of Service. 2015 IEEE Int Conf Commun (ICC) IEEE. 2015;

56. Seliuchenko, Marian et al. Enhanced Multi-commodity Flow Model for QoS-aware Routing in SDN. 2016 Int Conf Radio Electron Info Commun (UkrMiCo)IEEE. 2016;2-4.

57. Guck, Jochen W., Martin Reisslein and WK. Model-based control plane for fast routing in industrial QoS network. 2015 IEEE 23rd Int Symp Qual Serv (IWQoS)IEEE. 2015;65-6.

58. Seddiki, M. Said et al. FlowQoS: QoS for the rest of us. Proc third Work Hot Top Softw Defin networking. 2014;

59. Lin, Chienhung, Kuochen Wang and GD. A QoS-aware routing in SDN hybrid networks. Procedia Comput Sci [Internet]. 2017;242-9. Disponible sur: http://dx.doi.org/10.1016/j.procs.2017.06.091

60. Qin Z, Denker G, Al. E. A software defined networking architecture for the internet-of-things. 2014 IEEE Netw Oper Manag Symp. 2014;

61. Xifra Porxas A. Virtualization-enabled Adaptive Routing for QoS-aware Software-Defined Networks. 2014.

62. Rengaraju, Perumalraja, S. Senthil Kumar and C-HL. Investigation of security and QoS on SDN firewall using MAC filtering. 2017 Int Conf Comput Commun Informatics (ICCCI) IEEE. 2017;

63. D.Sonawane VR et al. QoS based Optimal Route Discovery Using OpenFlow. 2015;

68. Avallone, Stefano et al. D-ITG distributed internet traffic generator. First Int Conf Quant Eval Syst 2004 QEST 2004 Proceedings IEEE. 2004;

74. Phemius, Kevin and MB. Monitoring latency with OpenFlow. Proc 9th Int Conf Netw Serv Manag (CNSM 2013) IEEE. 2013;

Webographie

1. Kucminski, Andrew et al. QoS-based routing over software defined networks. IEEE Int Symp Broadband Multimed Syst Broadcast (BMSB)IEEE. 2017;

4. Germain M. Introduction au réseaux. Forum ATENA; 2012. 1-17 p.

12. Göransson P, Black C. Software Defined Networks A Comprehensive Approach. Mark Roger. Elliot S, éditeur. 2014.

13. Babu R. Dawadi , Danda B. Rawat SRJ. Software Defined IPv6 Network: A New Paradigm for Future Networking. J Inst Eng. 2019;15(2):1-13.

15. Cox JH, Chung J, Donovan S, Ivey J, Clark RJ, Riley G, et al. Advancing software-defined networks: A survey. IEEE Access. 2017;5:25487-526.

16. M Smith, M Dvorkin YL et al. OpFlex control protocol. IETF. 2014;

17. Bianchi, Giuseppe et al. Openstate: Programming platform-independent stateful openflow applications inside the switch. ACM SIGCOMM Comput Commun Rev. 2014;44-51.

18. Shaghaghi, Arash MAK +et al. Software-Defined Network (SDN) Data Plane Security: Issues, Solutions, and Future Directions. Clust Comput J. 2018;24.

21. Kreutz D, Ramos FMV, Verissimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: A comprehensive survey. Proc IEEE. 2014;103(1):14-76.

24. Dong X, Lin H, Tan R, Iyer RK, Kalbarczyk Z. Software-defined networking for smart grid resilience: Opportunities and challenges. Proc 1st ACM Work Cyber-Physical Syst Secur. 2015;61-8.

25. Kaivola R. Practical Aspects of Declarative Languages. Hutchison D, Kanade T, Kittler J, Kleinberg JM, éditeurs. Vol. 6539, 13th International Symposium, PADL 2011, Austin, TX, USA, January 24-25, 2011. Proceedings. Austin, TX, USA; 2011.

26. Monsanto C, Foster N, Harrison R, Walker D. A compiler and run-time system for network programming languages. In Philadelphia, PA, USA: ACM; 2012. p. 217-30.

28. Monsanto, Christopher et al. Composing Software-Defined Networks. 10th {USENIX} Symp Networked Syst Des Implement ({NSDI} 13). 2013;

29. Benamrane F, Mamoun M Ben, Redouane B. Etude des Performances des Architectures du Plan de Contrôle des Réseaux Software -Defined Networks. 2017.

31. Le Protocole OpenFlow dans l ’ Architecture SDN (Software Defined Network). 2016.

32. Anis A, Yanis A. Etude et implémentation d’une architecture SDN LAN. Mouloud MAMMERI DE TIZI- OUZOU; 2018.

33. Fernandez MP. Comparing OpenFlow controller paradigms scalability: Reactive and proactive. 2013 IEEE 27th Int Conf Adv Inf Netw Appl (AINA) IEEE. 2013;1009-16.

34. Lara, Adrian, Anisha Kolasani and BR. Network innovation using open flow: A survey. IEEE Commun Surv tutorials. 2014;16(1):493-512.

35. Ongaro F. ENHANCING QUALITY OF SERVICE IN SOFTWARE-DEFINED NETWORKS. Diss Alma Mater Stud Bol. 2014;

38. Chemeritskiy, Eugene and RS. On QoS management in SDN by multipath routing. 2014 Int Sci Technol Conf (Modern Netw Technol. 2014;

41. Shenker S. « Integrated Services in the Internet Architecture: an Overview Status of this Memo ». 1994; Disponible sur: https://dl.acm.org/doi/pdf/10.17487/RFC1633

44. Mehdi C, Adel D. Étude et implémentation de l’ approche Software Defined Network dans. 2018.

46. Villapol ME, Billington J. Internet service quality: A survey and comparison of the IETF approaches. Telecommun J Aust [Internet]. 2000;50(2):57-68. Disponible sur: http://www.ciens.ucv.ve/mvillap/Papers/TJApaperD.pdf

51. Al-Jawad A, Trestian R, Shah P, Gemikonakli O. BaProbSDN: A probabilistic-based QoS routing mechanism for Software Defined Networks. Proc 2015 1st IEEE Conf Netw Softwarization (NetSoft)IEEE. 2015;

52. Yan, Jinyao et al. HiQoS: An SDN-based multipath QoS solution. China Commun. 2015;12(5):123-33.

53. Zhang, Hailong et al. Sdn-based ecmp algorithm for data center networks. 2014 IEEE Comput Commun IT Appl Conf IEEE. 2014;

54. Akella, Anand V. and KX. Quality of service (QoS)-guaranteed network resource allocation via software defined networking (SDN). Proc - 2014 World Ubiquitous Sci Congr 2014 IEEE 12th Int Conf Dependable, Auton Secur Comput DASCIEEE. 2014;7-13.

55. Adami, Davide et al. A network control application enabling Software-Defined Quality of Service. 2015 IEEE Int Conf Commun (ICC) IEEE. 2015;

56. Seliuchenko, Marian et al. Enhanced Multi-commodity Flow Model for QoS-aware Routing in SDN. 2016 Int Conf Radio Electron Info Commun (UkrMiCo)IEEE. 2016;2-4.

57. Guck, Jochen W., Martin Reisslein and WK. Model-based control plane for fast routing in industrial QoS network. 2015 IEEE 23rd Int Symp Qual Serv (IWQoS)IEEE. 2015;65-6.

58. Seddiki, M. Said et al. FlowQoS: QoS for the rest of us. Proc third Work Hot Top Softw Defin networking. 2014;

59. Lin, Chienhung, Kuochen Wang and GD. A QoS-aware routing in SDN hybrid networks. Procedia Comput Sci [Internet]. 2017;242-9. Disponible sur: http://dx.doi.org/10.1016/j.procs.2017.06.091

60. Qin Z, Denker G, Al. E. A software defined networking architecture for the internet-of-things. 2014 IEEE Netw Oper Manag Symp. 2014;

61. Xifra Porxas A. Virtualization-enabled Adaptive Routing for QoS-aware Software-Defined Networks. 2014.

62. Rengaraju, Perumalraja, S. Senthil Kumar and C-HL. Investigation of security and QoS on SDN firewall using MAC filtering. 2017 Int Conf Comput Commun Informatics (ICCCI) IEEE. 2017;

63. D.Sonawane VR et al. QoS based Optimal Route Discovery Using OpenFlow. 2015;

68. Avallone, Stefano et al. D-ITG distributed internet traffic generator. First Int Conf Quant Eval Syst 2004 QEST 2004 Proceedings IEEE. 2004;

74. Phemius, Kevin and MB. Monitoring latency with OpenFlow. Proc 9th Int Conf Netw Serv Manag (CNSM 2013) IEEE. 2013;

Webographie

2. Présentation de l'entreprise [Internet]. [cité 15 juill 2020]. Disponible sur: https://www.algerietelecom.dz/fr/page/presentation-du-groupe-p2

3. BARA AR. NOTICE d'information ALGERIE TELECOM SPA [Internet]. Bulletin Officiel des Annonces Légales (B.O.A.L). 2005. Disponible sur: https://www.cosob.org/wp-content/uploads/2014/12/les-emetteurs-notice-algerietelecom.pdf

5. Principe du SDN dans une architecture réseau classique | Frédéric Launay [Internet]. [cité 15 juill 2020]. Disponible sur: http://blogs.univ-poitiers.fr/f-launay/2018/01/15/principe-du-sdn-dans-une-architecture-reseau-classique/?fbclid=IwAR0s674Mm851hb63aDLfqK3VFEfxzOGxMNWgrn0qmZ3ddkTg5-2478H5C1k

6. SDN Versus Traditional Networking Explained [Internet]. [cité 21 août 2020]. Disponible sur: https://www.ibm.com/services/network/sdn-versus-traditional-networking?fbclid=IwAR2r35ClihOTOZk5luJx4v48nrTfQrgYnWGigKPOcqqEhLMb98cTzpWPFF4

7. Qu'est-ce que le SDN (Software Defined Networking) ? - IONOS [Internet]. [cité 21 août 2020]. Disponible sur: https://www.ionos.fr/digitalguide/serveur/know-how/software-defined-networking/?fbclid=IwAR3r4UTJlETCtPKpY1yLb0aSYsutI3eEq6FGDEEhIBzAim2lEUwipuXxVV4

8. Chahed B. Mise en oeuvre des aspects de gestion des réseaux définis par logiciels (réseaux SDN) [Internet]. Université de Montréal; 2015. Disponible sur: https://publications.polymtl.ca/1924/

9. Choukri, Ihssane, Mohammed Ouzzif and KB. Software Defined Networking (SDN): Etat de L ' art. 2019; Disponible sur: https://hal.archives-ouvertes.fr/hal-02298874

10. Azodolmolky S. Software Defined Networking with OpenFlow [Internet]. Packt Publishing Ltd. 2013. 153 p. Disponible sur: http://medcontent.metapress.com/index/A65RM03P4874243N.pdf

11. Software-Defined Networking (SDN) Definition - Open Networking Foundation [Internet]. [cité 15 juill 2020]. Disponible sur: https://www.opennetworking.org/sdn-definition/

14. An SDN Perspective to Mitigate the Energy Consumption of Core Networks [Internet]. [cité 15 juill 2020]. Disponible sur: https://www.researchgate.net/figure/Traditional-Network-versus-SDN_fig1_319876305

19. Singla R. Contrail Architecture [Internet]. Juniper Network. 2013. Disponible sur: https://www.juniper.net/us/en/local/pdf/whitepapers/2000535-en.pdf

20. Foster N, Harrison R, Freedman MJ, Monsanto C, Rexford J, Story A, et al. Frenetic: A network programming language. ACM Sigplan Not [Internet]. 2011;46(9):279-91. Disponible sur: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.300.5643&rep=rep1&type=pdf

22. J.Dix. Clarifying the role of software-defined networking northbound APIs [Internet]. [cité 15 juill 2020]. Disponible sur: https://www.networkworld.com/article/2165901/clarifying-the-role-of-software-defined-networking-northbound-apis.html

23. Le Software Defined Networking | Coursera [Internet]. 2015 [cité 15 juill 2020]. Disponible sur: https://www.coursera.org/learn/sdn

27. Voellmy, Andreas, Hyojoon Kim and NF. Procera: A Language for High-Level Reactive Network Control. Proc first Work Hot Top Softw Defin networks [Internet]. 2012;43-8. Disponible sur: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.869.2661&rep=rep1&type=pdf

30. Software-Defined Networking-Open Flow [Internet]. [cité 15 juill 2020]. Disponible sur: http://www-igm.univ-mlv.fr/~dr/XPOSE2014/software-defined_networking/openflow.html

36. Metzler, Ashton and AM. Ten Things to Look for in an SDN Controller [Internet]. 2013. Disponible sur: https://www.necam.com/docs/?id=23865bd4-f10a-49f7-b6be-a17c61ad6fff

37. Szigeti, Tim et al. End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks. [Internet]. Cisco pres. 2014. Disponible sur: https://ptgmedia.pearsoncmg.com/images/9781587143694/samplepages/1587143690.pdf

39. What is QoS (quality of service) ? - Definition from WhatIs.com [Internet]. [cité 15 juill 2020]. Disponible sur: https://searchunifiedcommunications.techtarget.com/definition/QoS-Quality-of-Service

40. Qu'est-ce que le gigue ? [Internet]. [cité 1 sept 2020]. Disponible sur: https://www.speedcheck.org/fr/wiki/gigue/

42. L. Zhang SB et al. Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification [Internet]. [cité 15 juill 2020]. Disponible sur: https://tools.ietf.org/html/rfc2205

43. Bambrik Ilyas. Chapitre VI La Qualité de Service. [Internet]. [cité 15 juin 2020] .https://elearn.univ-tlemcen.dz/pluginfile.php/109043/mod_resource/content/3/Chapitre%20VI%20La%20Qualit%C3%A9%20de%20Service.pdf

45. Resource ReSerVation Protocol (RSVP) [Internet]. [cité 15 juill 2020]. Disponible sur: http://tom1226.blogspot.com/2010/01/resource-reservation-protocol-rsvp.html

47. E. Davies MAC et al. An Architecture for Differentiated Services [Internet]. [cité 6 juill 2020]. Disponible sur: https://tools.ietf.org/html/rfc2475

48. http dl free fr k FB 3 ljra 4 cours [Internet]. [cité 24 juill 2020]. Disponible sur: https://present5.com/http-dl-free-fr-k-fb-3-ljra-4-cours/

49. F. Goffinet. Concepts QoS Cisco [Internet]. [cité 15 juill 2020]. Disponible sur: https://cisco.goffinet.org/ccna/gestion-infrastructure/concepts-qos-cisco/#1-introduction-à-la-qualité-de-service-qos

50. Policing과 Shaping의 차이 | NETMANIAS [Internet]. [cité 24 juill 2020]. Disponible sur: https://www.netmanias.com/ko/post/blog/5379/network-protocol/policing-vs-shaping

64. POX Manual Current documentation [Internet]. [cité 18 août 2020]. Disponible sur: https://noxrepo.github.io/pox-doc/html/

65. Mininet Overview [Internet]. [cité 18 août 2020]. Disponible sur: http://mininet.org/overview/

66. MININET [Internet]. [cité 18 août 2020]. Disponible sur: https://www.opennetworking.org/mininet/

67. William E. S. Yu. CS 154: Introduction to Mininet [Internet]. [cité 18 août 2020]. Disponible sur: http://cng.ateneo.edu/cng/wyu/classes/cs154/cs154-mn-intro.pdf

69. D-ITG Software Architecture [Internet]. [cité 21 août 2020]. Disponible sur: https://www.researchgate.net/figure/D-ITG-Software-Architecture-9_fig2_303279498

70. Xming [Internet]. [cité 21 août 2020]. Disponible sur: https://xming.fr.softonic.com/

71. Télécharger PuTTY [Internet]. [cité 21 août 2020]. Disponible sur: https://www.01net.com/telecharger/windows/Internet/serveur_ftp/fiches/20166.html

72. Welcome to Python.org [Internet]. [cité 21 août 2020]. Disponible sur: https://www.python.org/

73. pox/l2_multi.py at eel • noxrepo/pox [Internet]. [cité 31 août 2020]. Disponible sur: https://github.com/noxrepo/pox/blob/eel/pox/forwarding/l2_multi.py

Résumé

Dans les dernières années, l'émergence des nouvelles générations de réseaux a provoqué l'augmentation des besoins des applications en termes de Qualité de Service (QoS). Ainsi, il est nécessaire de développer des algorithmes de routage qui tiennent compte des paramètres de QoS. Dans ce projet, nous étudions l'effet sur la performance réseau d'une approche de découverte de route basée sur la QoS à l'aide d'Open Flow et le SDN (Software Defined Network). Cette méthode prend en considération la latence des liens, la bande passante et les pertes des paquets sur les files d'attente comme paramètres de QoS pour calculer le chemin le plus court en fonction des exigences de service. En outre, pour évaluer l'efficacité de cet algorithme de routage avec QoS on va le comparer avec un algorithme de routage sans QoS. L'environnement de simulation des approches est implémenté grâce à l'émulateur Mininet. Enfin, les résultats d'analyse de performance sont obtenus à l'aide d'un générateur de trafic D-ITG pour les interpréter.

Mots clefs : SDN, QoS, OpenFlow, DSCP, D-ITG.

Abstract

The use of new generations of networks for multimedia and real-time applications is characterized by an increase in Quality of Service (QoS) requirements in terms of end-to-end delay, packet loss rates, and bandwidth utilization, etc. As a result, it is necessary to develop routing algorithms that take QoS parameters into account. In this project, a QoS-based route discovery approach using Open Flow and SDN (Software Defined Network) will be studied, taking into account link latency, bandwidth and packet losses on queues as QoS parameters to compute the best path depending on service requirements. In addition, to evaluating the efficiency of this routing algorithm with QoS management, it will be compared with a non-QoS routing algorithm based on the best effort policy. This method is simulated and tested using the Mininet emulator. Finally, performance analysis is done by running D-ITG traffic generator.

Keywords: SDN, QoS, OpenFlow, DSCP, D-ITG.

ملخص

يتميز استخدام الأجيال الجديدة من الشبكات في سياق الوسائط المتعددة وتطبيقات الوقت الفعلي بزيادة متطلبات جودة الخدمة من حيث التأخير من طرف إلى طرف (QoS). ومعدل فقدان الحزمة ، و استخدام الشريط ، إلخ لذلك ، من الضروري تطوير خوارزميات التوجيه التي تأخذ في QoS. الاعتبار معلمات

ونأخذ في الاعتبار زمن QoSالخدمة في هذا المشروع ، سنقوم بدراسة وتنفيذ نهج اكتشاف المسار استنادًا إلى جودةQoS باستخدام Open Flow انتقال الارتباط وعرض النطاق الترددي وفقدان الحزمة في قوائم الانتظار كمعلمات بالإضافة إلى ذلك ، لتقييم كفاءة خوارزمية التوجيهمع لحساب أقصر مسار وفقًا لمتطلبات الخدمة الخدمة، سنقارنها بخوارزمية توجيه بدون جودة الخدمة بناءً على أفضل خدمة (Tos / DSCP).جودة باستخدام محاكي mininet.مجهود. يتم تنفيذ بيئة محاكاة النهج

أخيرًا ، يتم الحصول على نتائج تحليل الأداء باستخدام منشئ حركة مرورD-ITG لتفسيرها .

كلمات أساسية : SDN, QoS, OpenFlow, DSCP, D-ITG