

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des sciences et de la technologie HOUARI BOUMEDIENNE



**Faculté d'électronique et d'informatique
Département Télécommunications**

Mémoire de MASTER

Domaine : Sciences et Technologie
Filière : Télécommunications

THEME

**L'évaluation de la QoS pour le
multimédia streaming dans
l'environnement SDN**

présenté par :

**HANOU Rimy
BOUHALISSA Ayoub**

Soutenu devant le jury composé de :

**Président : Pr SMARA Youcef
Examinateur : Dr DEMRI Ilyes
Promotrice : Pr MERAZKA Fatiha**

2021/2022

Remerciements

Avant tout, nous remercions Dieu le tout puissant de nous avoir donné la force, le courage, la persistance et nous a permis d'accomplir ce modeste travail. Merci de nous avoir éclairé le chemin de la réussite.

Nous tenons à remercier chaleureusement notre promotrice, Madame MERAZKA Fatihha, pour son aide, ses conseils, ses orientations et sa grande gentillesse. Merci pour avoir accepté d'encadrer ce mémoire et diriger ce travail, pour votre présence et votre disponibilité permanente.

Nous tenons à exprimer notre reconnaissance à Monsieur le professeur SAMARA Youcef, d'avoir accepté de présider le jury et de nous avoir permis une telle formation.

Nos remerciements également à Monsieur DEMRI Ilyes, pour avoir aimablement accepté d'examiner ce travail, qu'il accepte ce mémoire en gage de notre grand respect.

Enfin, on adresse nos sincères sentiments de gratitude et de reconnaissances à toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Dédicaces

Je dédie ce mémoire à mes chers parents qui ont été toujours à mes côtés et m'ont toujours soutenu tout au long de ces années d'études.

En signe de reconnaissance , qu'ils trouvent ici, l'expression de ma profonde gratitude pour tout ce qu'ils ont consenti d'efforts et de moyens pour me voir réussir dans mes études. A toute ma famille et à toutes mes amies.

A tous les gens qui me connaissent et que je connais et à tous ceux qui aiment le bon travail et ne reculent pas devant les obstacles de la vie.

- *Ayoub*

Dédicaces

La réalisation de ce travail constitue le couronnement d'une multitude d'efforts et l'aboutissement d'un long parcours ; bien trop souvent éprouvant, et n'a été rendue possible que grâce aux concours, aux sacrifices et à la patience de plusieurs personnes ; à qui je voudrais témoigner ici toute ma gratitude.

Je voudrais tout d'abord adresser toute ma reconnaissance à ma famille et à mes proches ; à mes parents pour leur patience et pour leur présence tout au long de ce parcours qui a toujours su me diriger vers les bons choix. A ma tante SAMIRA pour son inébranlable foi en moi aux moments où je n'en avais aucune, pour son indéfectible soutien et l'étendue de son amour.... À mon oncle RACHID pour sa droiture, son sens de la famille et toutes ces valeurs et l'exemple qu'elles ont pu être pour moi. A ma grand-mère FATIMA, que Dieu la garde et la préserve ; pour sa vie de sacrifices et l'infnie bonté qui remplit son cœur, pour son amour et pour toutes ces fois où elle a pu porter l'enfant que j'étais sur son dos, et pour tout ce que l'homme que je suis aujourd'hui a pu devenir.

Je dédie ce travail à mon grand frère IMAD et ma petite sœur LYNA, à mes plus proches amis pour leur encouragement, leur soutien, leur temps et leur présence aux plus durs des moments mais aussi aux meilleurs.

A la mémoire de tous ceux qui ne sont plus avec nous aujourd'hui. Que Dieu tout puissant, les acceptent en son vaste royaume.

- *Rimy*

Table des matières

Table des figures	vii
Liste des tableaux	ix
Liste des Abréviations	x

Introduction Générale	1
Chapitre 1 : Généralités sur SDN	4
1.1 Introduction	4
1.2 Définition du SDN	5
1.3 Les réseaux SDN vs les réseaux traditionnels	5
1.4 Objective du SDN	6
1.4.1 Scalabilité :	6
1.4.2 Haute disponibilité :	6
1.4.3 Sécurité :	7
1.4.4 Fiabilité :	7
1.4.5 Performance :	7
1.4.6 Elasticité :	7
1.5 L'architecture du SDN	7
1.5.1 Data plane	8
1.5.2 Southbound API	9
1.5.3 Interfaces Est/Ouest	11
1.5.4 Contrôle plane	11
1.5.5 Northbound API	11
1.5.6 Application plane	12
1.6 Types de contrôleurs SDN	12
1.7 Contrôleur Ryu	12
1.7.1 les Bibliothèques de Ryu	13
1.7.2 Protocole OpenFlow et contrôleur Ryu	14
1.7.3 Ryu manager et Fonctionnement	14
1.7.4 Ryu Northbound API	14
1.7.5 Les applications Ryu	14
1.8 Openvswitch	14
1.9 Conclusion	15

Chapitre 2 : Qualité de Service dans les réseaux	17
2.1 Introduction	17
2.2 Définition de la QoS	18
2.2.1 Critères de la qualité de service	18
2.3 Niveaux de service	18
2.3.1 IntServ	18
2.3.2 DiffServ	19
2.4 Application Multimédia	23
2.4.1 La vidéo conférence	23
2.4.2 La Télésurveillance	23
2.4.3 La vidéo à la demande	24
2.5 Conclusion	24
Chapitre 3 : QoS dans SDN : Implémentation et résultats	27
3.1 Introduction	27
3.2 Mininet	27
3.3 Workload	28
3.3.1 iPerf	28
3.3.2 Wireshark	29
3.3.3 VLC	29
3.4 Implémentation de la topologie Mininet	30
3.5 Expérimentation	34
3.5.1 Implémentation de DiffServ	34
3.6 1er Scénario : Avant l'implémentation de la QoS	38
3.7 2ème Scénario : Apres l'implémentassions de la QoS	39
3.8 Évaluation du retard pour le multimédia streaming :	41
3.9 Évaluation des pertes de paquets pour le multimédia streaming :	42
3.10Évaluation de la qualité du multimédia streaming avant et après l'im- plémentatation de la QoS :	42
3.11Conclusion	43
Conclusion Générale	44
Références	46

Appendice A

A1

Appendice B

B1

Table des figures

1.1 Les bénéfices du SDN	6
1.2 La structure logique du SDN	8
1.3 processus de transmission d'un paquet avec OpenFlow	10
1.4 Ryu Framework	13
2.1 Le champ ToS dans l'entête IP	19
2.2 Le champ DS et DSCP PHB dans l'entête IP	20
2.3 Les classes DSCP	21
2.4 class-selector PHB	22
3.1 Topologie réseaux créé	30
3.2 Topologie de réseau créé	31
3.3 Pingall non réussie	31
3.4 Configuration des adresses IP de h1 et h2	32
3.5 Configuration des routeurs OpenFlow.	32
3.6 Activation Controller Ryu	32
3.7 Configuration des routes.	33
3.8 Ping réussie	33
3.9 Configuration d'adresse OVSDB	34
3.10Résultat de l'implémentation.	35
3.11Création fils d'attentes	36
3.12Configuration QoS sur le routeur S1	36
3.13Configuration QoS sur le routeur S2	37
3.14Vérification des QOS et Queue settings	37
3.15Avant implémentation de QoS	38
3.16Avant implémentation de QoS	38
3.17Après implémentation de la QoS	39
3.18Après implémentation de QoS	40
3.19Comparaison de la bande passante avec et sans QoS	41
3.20Comparaison du délai avec et sans QoS	41
3.21Comparaison des pertes de paquets avec et sans QoS	42

Liste des tableaux

1.1 Structure d'une entrée de table de flux d'un commutateur OpenFlow	
1.0	9
1.2 Quelques contrôleurs SDN les plus connus	12
2.1 AF codepoints	22
3.1 Exigences du test	30
3.2 Les filles d'attente	35
3.3 Configuration de la QoS	36
3.4 Le marquage DSCP sur le Routeur S2	37

Liste des Abréviations

Symbole	Description
AF	Assured Forwarding
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BA	Behaviour Aggregator
BE	Best Effort
BW	Bandwidth
Cos	Class of services
CS	Class Selector
CSP	Constrained Shortest Path
DIFFSERV	Differentiated Service
DS field	Differentiated Services field
DSVP	Differentiated Service Code Point
ECN	Explicit Congestion Notification

Symbole	Description
EF	Expedited forwarding
GRE	Generic Routing Encapsulation
IETF	Internet Engineering Task Force
INTSERV	Integrated Service
IOS	Internet work Operating System
IP	Internet protocol
KPI	Key Performance Indicator
NBI	Northbound Interface
NOS	Network Operating System
OF	OpenFlow
OVS	OpenvSwitch
OVSDDB	Open VSwitch Database
PHB	Per Hop Behavior
PQ	Packet Queuing
QoS	Quality of Service
RSVP	Resource Reservation Protocol
SDN	Software-Defined Networks
SLA	Service-level agreement

Symbole	Description
SONET	Synchronous Optical NETwork
SBI	Southbound Interface
TCP	Transmission Control Protocol
TOS	type of service
TOS	Type of Service
UDP	User Datagram Protocol
VLC	Video LAN Client
VoIP	Voice over Infrastructure Procedure
WAN	Wide Area Network

Introduction Générale

Introduction Générale

La consommation croissante de services multimédia et la demande de services de haute qualité de la part des clients ont déclenché un changement fondamental dans les méthodes d'administration des réseaux en termes d'allocation des ressources limitée comme la bande passante et la garantie d'une connexion fiable avec le moins de coupures et de latence. L'industrie et le monde universitaire considèrent la 5G comme le futur réseau capable de prendre en charge la prochaine génération d'applications verticales avec différentes exigences de service. Pour concrétiser cette vision dans le réseau 5G, la softwarisation à l'aide de la mise en réseau définie par logiciel (SDN) est devenu une nécessité pour combler le vide du contrôle et de la gestion des ressources réseaux dont le but est d'assurer l'implémentation d'une bonne la qualité de service (QoS).

La gestion de la QoS est un concept important dans les réseaux informatiques et les télécommunications, car elle affecte directement l'expérience perçue par l'utilisateur final, en particulier pour les applications en temps réel telles que le streaming de données et le multimédia. L'objectif de la QoS de bout en bout est de satisfaire l'utilisateur final tout en maintenant un certain niveau de qualité. L'évolution de l'industrie des réseaux au cours des deux dernières décennies est basée sur un modèle dans lequel les périphériques réseaux exécutent à la fois des fonctions de contrôle et de distribution des paquets de données de manière distribuée. Bien que efficaces, ces techniques se sont avérées très difficiles à implémenter. L'avènement du concept de réseau SDN, ainsi que du protocole OpenFlow, ont considérablement simplifié le contrôle de la QoS.

SDN en tant que nouveau paradigme de mise en réseau offre une opportunité de réintroduire le contrôle de la QoS dans le réseau. La nature centralisée du SDN réduit considérablement la complexité généralement associée aux garanties QoS. Avec son adoption, le SDN est en route pour être adopté comme norme de référence. En créant des contrôles QoS solides dans le SDN, les futurs réseaux peuvent avoir une prise en charge native de la QoS.

L'objectif de ce travail est d'évaluer la QoS dans un environnement SDN pour le flux multimédia. Pour avoir une bonne QoS, les caractéristiques suivantes doivent être appliquées :

Donner des garanties de bande passante par flux, prioriser correctement le trafic en fonction de la garantie toute en respectant les limites fixées et d'autoriser le trafic excessif lorsque les ressources sont disponibles.

Ce travail est structuré en trois chapitres :

- **Chapitre 1** : présente des généralités sur les réseaux SDN.
- **Chapitre 2** : présente la notion de QoS et ses métriques.
- **Chapitre 3** : présente les étapes afin de concevoir une solution de mise en œuvre de la QoS dans un environnement SDN et donne les étapes à suivre pour implémenter cette solution. Il présente également les résultats obtenus des différents scénarios tels que l'envoi de données avec et sans QoS pour montrer l'efficacité de nos approches sur le multimédia streaming dans un environnement SDN.

Chapitre 1

Généralités sur SDN

Chapitre 1

Généralités sur SDN

1.1 Introduction

Les réseaux informatiques traditionnels ont connu d'énormes défis ces dernières années, principalement de la part des applications modernes déployées sur ces réseaux. Par conséquent, à mesure que le nombre d'utilisateurs et d'appareils augmente, le besoin de traitement distribué et l'adoption d'informations par plusieurs entreprises entraînent le besoin des réseaux dynamiques qui offrent la possibilité de s'adapter rapidement aux changements des requis.

C'est dans ce contexte que le paradigme du Software-Defined Networking (SDN) émerge, principalement pour s'adapter à la nature dynamique des différentes applications. En effet, le SDN permet principalement de centraliser la logique qui détermine les politiques de gestion du réseau dans une ou plusieurs unités appelées contrôleurs. Ces contrôleurs communiquent avec le reste des équipements du réseau [1]. Par conséquent, définir des politiques de gestion de réseau revient à écrire des programmes et à les déployer dans le contrôleur. Dans la plupart des cas, ces programmes seront compilés, en tenant compte de la topologie et des ressources disponibles, afin de générer la configuration nécessaire à chaque équipement réseau pour mettre en œuvre la politique souhaitée [1]. L'architecture SDN a été adoptée par plusieurs grandes entreprises et universités telles que Google et Stanford. D'autre part, les fabricants d'équipements réseau tels que Cisco, HP et Juniper Networks proposent désormais des solutions SDN pour gérer les centres de données [1].

1.2 Définition du SDN

Le SDN est une architecture de réseaux qui permet une gestion logicielle du réseau. Pour arriver à cela, le plan de contrôle est séparé de celui des données. SDN est une architecture réseau à quatre piliers :

- Les plans de contrôles et de données sont séparés. La fonction du contrôle est supprimée des périphériques réseaux pour qu'ils deviennent des éléments de transfert.
- Le transfert de données est basé sur le flux et non sur le destinataire. Dans le SDN le flux représente une séquence de trame entre une source et la destination, Toutes les trames d'un flux reçoivent des politiques de service identiques sur les éléments de transfert [2]. L'abstraction de flux permet de centraliser le comportement de différents types d'équipement du réseau, comme les routeurs, les commutateurs, les pares-feux et les middlebox [3]. La programmation du flux permet une flexibilité sans précédent, limitée aux capacités des tableaux de flux mis en œuvre [4] .
- Le contrôleur central est utilisé pour le plan de contrôle, c'est-à-dire le contrôleur SDN. Il peut s'agir d'une machine physique ou même d'une machine virtuelle, et son rôle est d'alimenter le plan de données des équipements réseaux avec son plan de contrôle. Par conséquent, un contrôleur SDN est une plate-forme logicielle qui s'appuie sur la technologie du serveur de base et fournit les ressources et les abstractions nécessaires pour faciliter la programmation des dispositifs de transfert, logiquement basée sur une vue abstraite et centralisée du réseau. En tant que tel, il représente un véritable cerveau virtuel du réseau et offre aux administrateurs une meilleure vision de tous les processus. Il réduit les coûts d'exploitation, améliore la fiabilité et la stabilité opérationnelles et augmente l'efficacité globale.
- Le réseau est programmable au moyen d'applications logicielles (Python, Java...) s'exécutant sur le contrôleur SDN qui interagit avec les équipements du plan de donnée sous-jacent. C'est une des caractéristiques fondamentales du SDN.

1.3 Les réseaux SDN vs les réseaux traditionnels

Le SDN est devenu l'actualité dans le monde des réseaux. Le marché a redirigé sa tendance vers les SDN et les considèrent comme une solution qui permet de supprimer les frontières existantes entre le monde des applications et du réseau [5].

Les réseaux traditionnels ne semblent pas avoir évolué et chaque modification nécessite encore des interventions manuelles sans orchestration dans la plupart des organisations. La mise en place d'un simple VLAN peut parfois prendre plusieurs jours, avec la nécessité de configurer un par un de nombreux équipements : commutateurs, routeurs, firewalls... [5]

Ce qui met les réseaux traditionnels face à plusieurs problématiques : simplifier le déploiement des applications sur les réseaux, mettre en place des politiques de sécurité / QoS homogènes

globalement pour les applications, automatiser des fonctions réseau directement à partir des applications [5].

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

- permettre aux applications de programmer le réseau afin d’en accélérer le déploiement
- permettre au réseau de mieux identifier les applications transportées pour mieux les gérer (qualité de service, sécurité, ingénierie de trafic...) [5]

Recourir à cette solution a apporté : scalabilité, haute disponibilité. Meilleure sécurité. Fiabilité, élasticité [6].

1.4 Objective du SDN

L’objectif de cette technologie est de simplifier l’administration réseaux en rendant les réseaux programmables (dynamiques), et aussi de rendre la consommation des ressources par le réseau plus flexible. La figure 1.1 présente les avantages d’adoption de l’approche SDN.

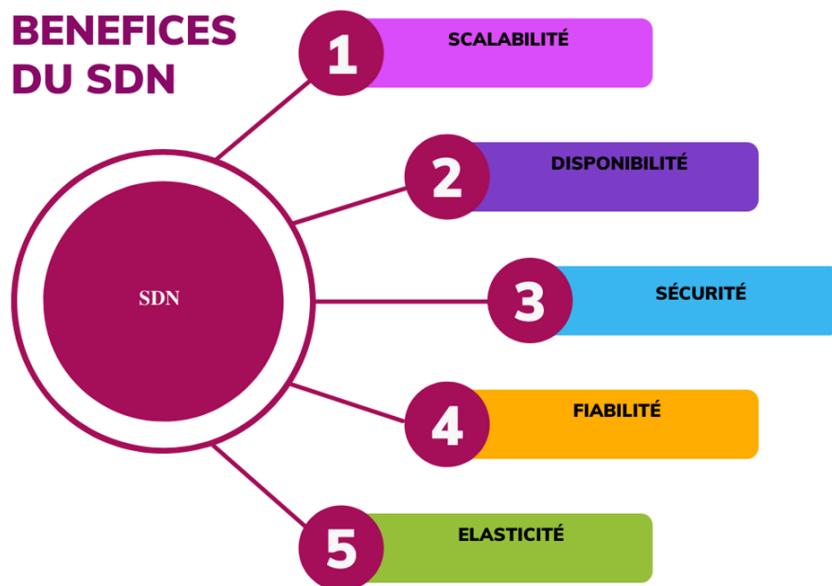


FIGURE 1.1 – Les bénéfices du SDN

1.4.1 Scalabilité :

Définit la capacité du SDN plus spécifiquement dans le plan contrôle, elle vise à étendre la capacité du SDN grâce à des mécanismes tels que le « Devolving » [7], le « Clustering » [8] et le « High Processing » [9] pour faire face à la charge croissante.

1.4.2 Haute disponibilité :

Elle représente un aspect important des services d’aujourd’hui, qui consiste en la disponibilité d’un service à chaque fois qu’un client le demande. L’indisponibilité des services peut générale-

ment se produire en raison de pannes de réseau ou de pannes de système [10], [11]. Une solution que les fournisseurs du réseau ont trouvée est le déploiement des services de sauvegarde pour offrir une "Haute disponibilité" en implémentant du matériel serveur redondant, des composants d'OS serveur et de réseau.

1.4.3 Sécurité :

La sécurité du SDN consiste à protéger les informations contre le vol ou l'endommagement du matériel et des logiciels ainsi que contre l'interruption des services [12] , [13]. La sécurisation du SDN englobe la sécurité physique du matériel, ainsi que la prévention des menaces logiques qui peuvent provenir du réseau ou des données. Il améliore également l'accès aux données sur les flux réseau. Ces précieuses données permettent de programmer des alertes, de détecter des attaques en temps réel ou d'analyser des événements passés. Cette fonctionnalité est essentielle à un moment où les entreprises mettent régulièrement plusieurs mois à détecter des intrusions sur leur réseau [14].

1.4.4 Fiabilité :

Les SDN sont considérées comme fiables en raison des annonces en temps réel des pannes de données. Dans un réseau, le routage des données critiques doit avoir un minimum de fiabilité spécifique. Dans l'implémentation actuelle, le contrôleur SDN doit être capable de répondre aux exigences de fiabilité et de rapidité de routage en temps réel [15].

1.4.5 Performance :

Dans SDN, cela signifie que le nombre de fonctions exécutées par les composants SDN est lié au temps et aux ressources utilisées [16]. Il existe de nombreuses façons différentes de mesurer les performances d'un réseau [17], [18], car chaque réseau est de nature et de conception différentes, pour le SDN, les métriques importantes sont la bande passante, le débit, la latence.

1.4.6 Elasticité :

L'élasticité dans le SDN est la capacité d'assurer et de maintenir un bon niveau de service même dans le cas de défaillance, d'un réseau ou d'un nœud [19].

1.5 L'architecture du SDN

Le SDN est une nouvelle façon d'appréhender les Réseaux. Il s'agit d'un renversement de paradigme : la fonction Control n'est plus distribuée mais centralisée. Le SDN ne laisse que la fonction Data Plane dans les noeuds du réseau [20].

La centralisation apporte une vision complète du réseau et une capacité à le modifier très rapidement. Dit autrement, la fonction Control donne des ordres à la fonction Data Plane de chaque nœud. Les nœuds ne sont plus que des exécutants pilotés par un serveur central [20].

Le schéma ci-dessous (Figure 1.2) décrit une nouvelle Architecture Réseau avec l'implémentation de la Technologie SDN [5].

Le SDN est composé principalement de 3 couches et d'interfaces logiques de communication.

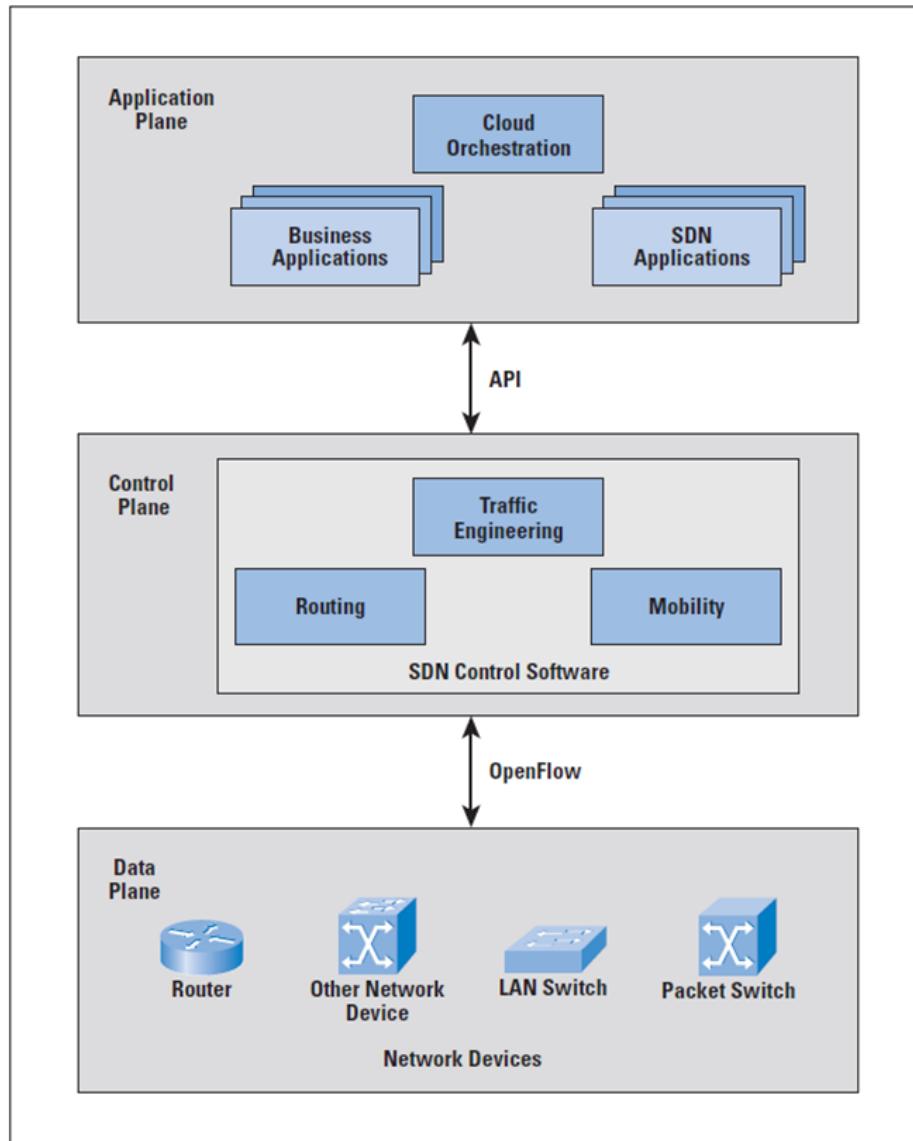


FIGURE 1.2 – La structure logique du SDN

Le niveau le plus bas, inclut le data plane. Le plus haut niveau on trouve l'application plane, le contrôle plane se trouve entre eux. La communication entre les contrôleurs et le data plane est géré via le SBI (Southbound Interface) qui se trouve au niveau du commutateur SDN et la communication entre les applications logicielles et les contrôleurs est assuré par NBI (Northbound Interface) qui se trouve dans le contrôle plane (Figure 1.2).

1.5.1 Data plane

Le plan de données dans le SDN est composé d'un ensemble d'équipements réseaux, tels que les commutateurs et les routeurs, des équipements virtuels, la principale différence avec les réseaux traditionnels est dans le fait que les équipements réseaux sont maintenant de simples éléments de transfert moins intelligent et sans fonction de contrôle et logicielle intégrée pour prendre des décisions autonomes.

L'intelligence réseau est retiré des équipements du plan de données et elle sera attribué à un système de contrôle logiquement centralisé (le contrôleur SDN) [21] .

Le but du plan de données est de transférer le trafic réseau selon un ensemble de règles de transfert ordonnées par le plan de contrôle. La communication entre le plan de données et le plan de contrôle est effectuée par une interface logique sud appelée API Southbound.

1.5.2 Southbound API

Les interfaces sud représentent les interfaces de communications, qui permettent au contrôleur SDN d'interagir avec les équipements du plan de données tels que les switchs et routeurs.

Elle contrôle les opérations de transfert, les notifications d'évènements, les rapports statistiques et annonce également la capacité du réseau. L'API Southbound (SBI) fournit une interface de couche supérieure commune qui permet aux contrôleurs de gérer des périphériques physiques ou virtuels existants ou nouveaux (tels que SNMP, BGP et NetConf) à l'aide d'API Southbound et de plugins de protocole. Celles-ci sont essentielles à la fois pour la compatibilité et l'hétérogénéité du fond. Par conséquent, sur le plan de donnée, un mélange de périphériques physiques, de périphériques virtuels et de diverses interfaces de périphérique (par exemple OpenFlow, OVSDB [21], OF-Config, NetConf et SNMP) peut coexister [22]. Le protocole le plus utiliser comme interface sud est OpenFlow, c'est le plus répandu dans les réseaux SDN.

Protocole OpenFlow

Openflow est le protocole utilisé pour la communication entre la couche transmission et la couche de contrôle, standardisé par l'Open Networking Fondation (ONF) il décrit l'interaction d'un ou plusieurs serveurs de contrôle avec les commutateurs compatibles OpenFlow.

Architecture OpenFlow

L'architecture OpenFlow est la mise en œuvre proprement dite d'un réseau SDN et repose principalement sur trois composants :

- Plan de données, composé de commutateurs OpenFlow
- Plan de contrôle, composé de contrôleurs OpenFlow.
- Une chaîne de sécurité qui permet au commutateur de se connecter au plan de contrôle.

Selon la spécification ONF [23], un commutateur OpenFlow doit contenir une ou plusieurs tables de flux contenant plusieurs entrées correspondant à des règles, où chaque entrée se compose principalement de trois champs suivants (Tableau 1.1) :

TABLE 1.1 – Structure d'une entrée de table de flux d'un commutateur OpenFlow 1.0

Champs d'en-tête	Compteurs	Actions
------------------	-----------	---------

L'En-tête de paquet : il définit le flux de données, il contient les informations nécessaires pour déterminer le paquet auquel cette règle sera appliquée, il peut identifier différents

protocoles tels que IPV4, IPV6, Ethernet.

L’Action : Décrit comment traiter les paquets dans un flux. Il peut s’agir de l’un des éléments suivants :

- Acheminer les paquets vers un ou plusieurs ports.
- Supprimer les paquets.
- Transmettre les paquets au contrôleur.
- Modifier le champ d’en-tête .

[24]

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 [22].

Fonctionnement OpenFlow

Lorsqu’un paquet arrive à un commutateur, le commutateur vérifie s’il y a une entrée dans la table de flux qui correspond à l’en-tête de paquet. Si c’est le cas, le commutateur exécute l’action correspondante dans la table de flux. Dans le cas contraire, c'est-à-dire il 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’, puis le contrôleur décide selon sa configuration une action pour ce paquet, et envoie une nouvelle règle de transmission sous la forme d'un ‘Packet_out’ et ‘Flow-mod’ au commutateur -3-, et enfin, la table de flux du commutateur est actualisée, pour prendre en compte la nouvelle règle installée par le contrôleur -4- [22].

La Figure 1.3 décrit le processus de transmission d'un paquet avec OpenFlow [25].

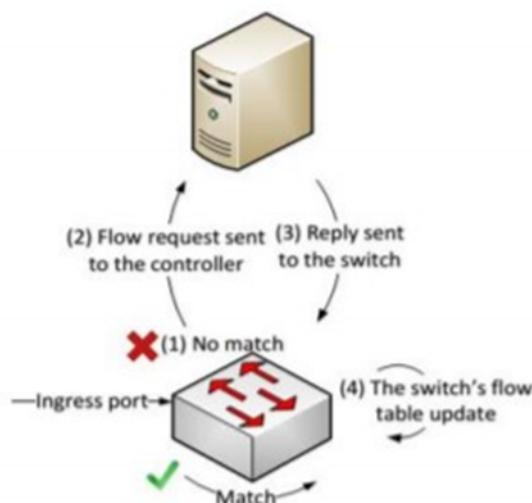


FIGURE 1.3 – processus de transmission d'un paquet avec OpenFlow

Protocole OVSDB

OpenFlow est un protocole sud qui permet la communication entre les contrôleurs et les commutateurs. La gestion du commutateur lui-même n'est pas gérée par OpenFlow. Pour cela, deux protocoles différents peuvent être utilisés, OVSDB et OF-Config. OF-Config est un protocole de gestion développé par l'ONF et devrait être applicable à tout appareil compatible OpenFlow, tandis que le protocole OVSDB est spécialement développé pour OpenvSwitch. OVSDB fonctionne avec les implémentations logicielles et matérielles d'OVS et gère les opérations de commutation telles que la création d'interfaces, la définition de politiques QoS ou l'arrêt de ports physiques [26].

1.5.3 Interfaces Est/Ouest

Les interfaces Est/Ouest sont des interfaces de communication qui permettent la communication entre les contrôleurs dans une architecture multi-contrôleurs pour synchroniser l'état du réseau [22]. Ces architectures sont très récentes et aucun standard de communication inter-contrôleur n'est actuellement disponible.

1.5.4 Contrôle plane

Le plan de contrôle est aussi connu comme la couche de contrôle SDN. Le plan de contrôle inclut un composant réseau spécial appelé contrôleur SDN, qui est logiquement centralisé mais physiquement distribué [27]. Le contrôleur est une plateforme logicielle qui est responsable d'établir et d'arrêter les flux et les acheminer dans les SDN. Le contrôleur SDN fournit des interfaces de programmation au système sous-jacent. La fonction de gestion globale de SDN est simplement confiée au contrôleur SDN tout en facilitant la programmation de l'ensemble du réseau. De façon similaire, la couche de contrôle fournit aussi une abstraction des ressources sous-jacentes. De plus, le modèle de contrôle logique centralisé du SDN peut être appliqué à une grande variété d'applications, de réseaux sous-jacents et de supports physiques, tels qu'Ethernet ou bien la fibre optique [27].

1.5.5 Northbound API

L'interface Nord est utilisée pour programmer les dispositifs de transmission, en tirant parti de l'abstraction du réseau fournie par le plan de contrôle. Contrairement à Southbound API qui est standardisé, l'interface Northbound API est toujours une question ouverte. L'architecture du contrôleur SDN et de l'interface Nord varie selon le fournisseur. Certains fournisseurs intègrent des contrôleurs SDN dans leurs applications, tandis que d'autres définissent des interfaces Nord pour faciliter les protocoles de communication entre les contrôleurs et leurs propres services SDN au niveau de la couche application. L'avantage d'une API nord ouverte est qu'elle permet plus d'innovations et d'expérimentation. Il existe plusieurs implémentations de cette interface, chacune offrant des fonctionnalités très différentes [21]. RESTful (ou bien REST) est considéré comme l'API nord la plus populaire des réseaux SDN.

1.5.6 Application plane

Représente les applications qui aident à déployer de nouvelles fonctionnalités réseau, telles que l'ingénierie du trafic, la qualité de service, la sécurité, etc. Grâce au plan de contrôle, les applications SDN peuvent facilement accéder à une vue globale du réseau instantanément via l'interface Northbound API. Dotées de ces informations, les applications SDN peuvent mettre en œuvre par programme des politiques pour manipuler le réseau physique sous-jacent à l'aide du langage de haut niveau fourni par la couche de contrôle. Dans ce domaine, le SDN fournit un modèle de « plate-forme en tant que service » pour la mise en réseau [28].

1.6 Types de contrôleurs SDN

Avec tous les outils nécessaires pour un réseau, un contrôleur SDN doit être sélectionné pour le gérer. De nombreux facteurs doivent être pris en compte lors du choix d'un contrôleur. Plusieurs contrôleurs ont été développés, dont la plupart sont open source et supportent le protocole OpenFlow. Le tableau 1.2 montre les contrôleurs SDN les plus populaires.

TABLE 1.2 – Quelques contrôleurs SDN les plus connus

Contrôleur	Organisation	Langage	Fonctionnalités	Protocole	L'architecture	API supportés	Plateformes supportées
NOX	Nicira	C++	Le premier contrôleur openflow	OpenFlow	Centralisée	Ad-hoc API	Linux Windows Mac OS
POX	Nicira	Python	Améliorer les performances de NOX	OpenFlow	Centralisée	Ad-hoc API	Linux Windows Mac OS
Ryu	NTT, OSRG group	Python	Supporte l'OpenStack	OpenFlow OVSDB	Centralisée	Ad-hoc API	Linux
Beacon	Stanford	Java	Basé sur le Multithreading	OpenFlow	Centralisée	Ad-hoc API	Linux Windows
Floodlight	Big Switch	Java	Testé avec des commutateurs OpenFlow physiques et virtuels	OpenFlow	Centralisée	REST API	Linux Windows Mac OS
Opendaylight	Linux Foundation	Java	Supporte le Framework OSGi et le REST API	OpenFlow OVSDB	Distribuée	REST API	Linux

1.7 Contrôleur Ryu

Dans l'environnement SDN, il est essentiel de sélectionner le contrôleur pour réaliser l'application proposée. Dans cette section, un contrôleur SDN, Ryu [29] sera discuté à travers ce flux d'opérations.

Le contrôleur Ryu est un contrôleur (SDN) conçu pour augmenter l'agilité du réseau en facilitant la gestion et le traitement du trafic. En général, le contrôleur SDN est le cerveau de l'environnement SDN, transmettant des informations aux commutateurs et aux routeurs via des API sud, et aux applications via des API nord comme il est montré dans la figure 1.4. Le contrôleur Ryu est alimenté par NTT et est également déployé dans les centres de données cloud de NTT, il est exécuté entièrement en Python.

Le contrôleur Ryu fournit des composants logiciels avec des interfaces de programmation d'applications (API) bien définies qui permettent aux développeurs de créer facilement de nou-

velles applications de gestion et de contrôle de réseau. Cette approche basée sur les composants logiciels aide les organisations à personnaliser les déploiements pour répondre à leurs besoins spécifiques ; les développeurs peuvent rapidement et facilement modifier les composants existants ou implémenter les leurs pour s'assurer que le réseau sous-adjacent peut répondre aux besoins changeants de leurs applications.

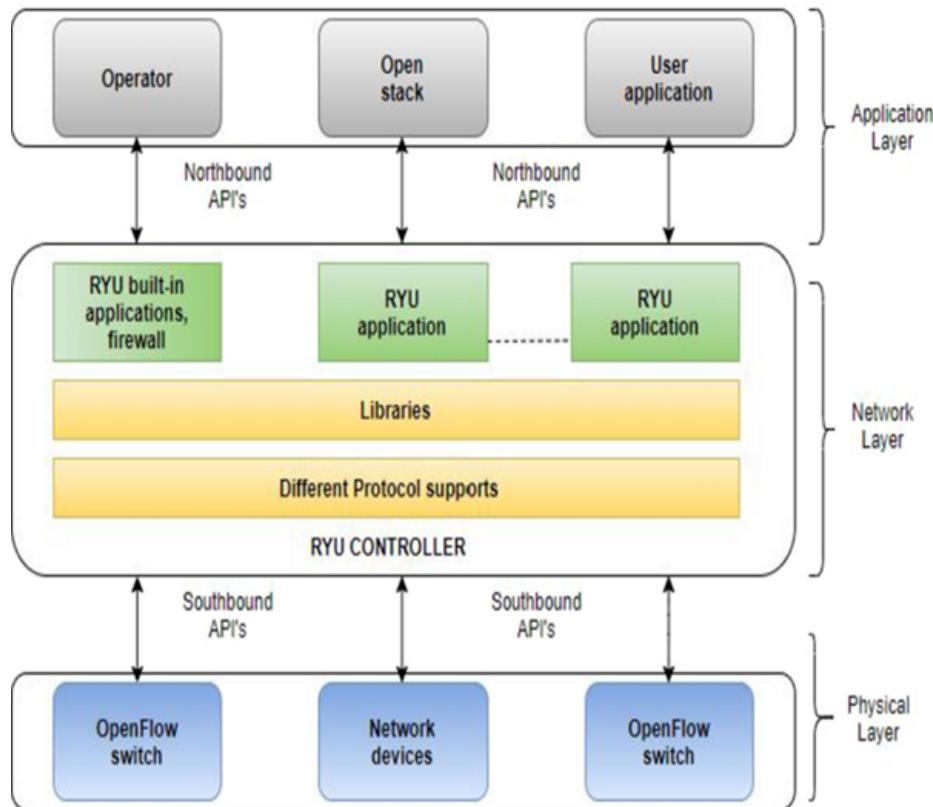


FIGURE 1.4 – Ryu Framework

Ryu supporte multiples protocoles API sud pour la gestion des périphériques, comme le protocole de configuration du réseau (NETCONF), OpenFlow, le protocole de gestion et de configuration OpenFlow (OF-Config), etc. Les composants essentiels de l'architecture sont expliqués dans la sous-session ci-dessous.

1.7.1 les Bibliothèques de Ryu

Ryu prend en charge plusieurs bibliothèques et plusieurs protocoles d'API sud. En ce qui concerne les protocoles API sud, Ryu prend également en charge le protocole de gestion de base de données Open vSwitch (OVSDB), OF-Config, NETCONF, Sflow, Netflow et d'autres protocoles. Les protocoles Sflow et Netflow peuvent être utilisés pour mesurer le trafic réseau en utilisant diverses méthodes telles que l'échantillonnage et l'agrégation de paquets. Avec l'aide de la bibliothèque de paquets de Ryu, le développeur réseau peut analyser et construire plusieurs paquets de protocole, comme VLAN, MPLS, etc.

1.7.2 Protocole OpenFlow et contrôleur Ryu

Ryu comprend un contrôleur interne et le protocole OF qui est l'un des protocoles sud supportés, il supporte le protocole OpenFlow à partir de la version 1.0 jusqu'à la dernière version.

Dans l'architecture Ryu, le contrôleur OpenFlow est l'une des sources d'événements internes qui peut gérer les commutateurs. De plus, Ryu inclut une bibliothèque d'encodeurs et de décodeurs du protocole OpenFlow.

1.7.3 Ryu manager et Fonctionnement

Ryu manager est l'exécutable principal du contrôleur ryu. Au démarrage, il écoute sur l'adresse IP spécifiée (ex : 0.0.0.0) et le port spécifié (6633 par défaut). Ensuite, n'importe quel commutateur OpenFlow (matériel, ou Open vSwitch) peut se connecter à Ryu Manager. Le gestionnaire d'applications (app manager) est le composant central de toutes les applications Ryu. Toutes les applications héritent de la classe RyuApp du gestionnaire d'applications. Les principaux composants de l'architecture incluent la gestion des événements, la transmission des messages, la gestion de l'état en mémoire, etc.

1.7.4 Ryu Northbound API

Au niveau de l'API, Ryu prend en charge une interface REST pour ses opérations OpenFlow. Ryu inclut également un plug-in Openstack Neutron qui prend en charge les configurations typiques basées sur VLAN et GRE. De plus, les chercheurs peuvent facilement créer des API REST à l'aide d'un framework Python appelé WSGI pour connecter des serveurs Web et des applications.

1.7.5 Les applications Ryu

L'application Ryu est l'un des éléments de base, il définit la logique de contrôle et de comportement. Plusieurs applications sont déjà incluses dans le framework Ryu, tel que topologie, simple_switch, pare-feu, routeur, etc. Bien que les applications Ryu soient implémentées et fournissent diverses fonctions, elles fonctionnent comme une seule unité d'exécution. Auparavant, les applications Ryu s'envoyaient des événements asynchrones.

1.8 Openvswitch

Bien que la spécification OpenFlow ait été mise à niveau, les commutateurs OpenFlow les plus populaires ne prennent pas encore en charge toutes les dernières fonctionnalités définies dans la spécification. En effet, la spécification OpenFlow répertorie les caractéristiques requises pour qu'un commutateur ait implémenté soit appelé commutateur compatible OF. Cependant, la spécification conserve certaines fonctionnalités facultatives. Ces fonctionnalités sont laissées au fabricant du commutateur pour décider de les mettre en œuvre. Il en résulte un état de l'art peu clair dans le domaine de la prise en charge de la QoS dans les commutateurs OpenFlow.

La plupart des commutateurs matériels commerciaux prêts pour la production prennent en charge la plupart des fonctionnalités QoS. Cela est compréhensible car ils sont censés être

déployés dans des réseaux réels qui nécessitent la plus haute qualité de service. Les commutateurs logiciels, d'autre part, sont principalement constitués d'implémentations open source et sont en retard dans l'implémentation de toutes ces fonctionnalités. L'équipe du contrôleur Ryu SDN test régulièrement divers commutateurs populaires pour la compatibilité OpenFlow et délivre des certifications. La liste peut être consultée dans [29]. Nous avons choisi d'utiliser OpenvSwitch dans notre projet, c'est l'un des meilleurs commutateurs SDN du marché.

1.9 Conclusion

Dans ce chapitre, nous nous concentrons sur les réseaux SDN, qui changent toutes les perspectives précédemment acquises sur les réseaux traditionnels. Tout d'abord nous décrivons ces avantages clés dans le domaine du réseau, puis nous comparons le SDN à l'architecture traditionnelle, ensuite nous avons vu en détail son architecture de bas en haut, en étudiant les parties essentielles de cette solution pour appliquer ses concepts à notre contexte. Enfin, dans la dernière partie de ce chapitre nous présentons les contrôleurs SDN les plus courants ainsi que le contrôleur Ryu qui va être utilisé lors de notre simulation.

Chapitre II

Qualité de Service dans les réseaux

Chapitre 2

Qualité de Service dans les réseaux

2.1 Introduction

La qualité de service (QoS) sur le réseau est un domaine de recherche et d'amélioration continue depuis des décennies. Les réseaux traditionnels, qui ont été conçus à l'origine sans QoS à l'esprit, ont depuis été complétés par un certain nombre d'approches pour obtenir le réglage de performance requis [30].

La QoS est la capacité d'un réseau à mieux servir le trafic réseau sélectionné grâce à diverses technologies, telles que les réseaux Ethernet et 802.1, le réseau optique synchrone (SONET), le relais de trames, le mode de transfert asynchrone (ATM) et les réseaux routés IP peuvent utiliser n'importe laquelle de ces technologies. L'objectif principal de la QoS est de fournir une bande passante dédiée, une gigue et un délai contrôlés et des taux de perte de paquets acceptables. Ces exigences représentent des prérequis pour le streaming en direct, comme le streaming vidéo et la téléphonie IP. Il est également important d'assurer que la priorité accordée à un flux n'est pas faite au détriment d'une baisse de qualité d'un autre flux moins important.

Les applications multimédias sont très courantes de nos jours, tant dans le monde informatique que dans la vie de tous les jours. Le terme multimédia est largement utilisé, et on parle souvent de la révolution multimédia ou tout simplement du multimédia. Les performances actuelles des différents types de périphériques disponibles sur le marché (téléphones mobiles, assistants personnels, ordinateurs portables, etc.), ainsi que l'avènement des réseaux et des débits actuels ont contribué à une large démocratisation du terme. Nous pouvons désormais échanger tout type de média à travers ces environnements.

2.2 Définition de la QoS

La qualité de service est la capacité à délivrer un type de flux donné dans de bonnes conditions (débit, latence, gigue, perte de paquets). Le concept vise à contrôler et à gérer les ressources du réseau en priorisant certains types de données réseau et permet de garantir de la bonne performance aux applications critiques et assurer de bonnes performances pour les applications critiques. Il permet aux fournisseurs de services de respecter les SLA négociés avec les clients et de fournir aux utilisateurs des débits et des temps de réponse spécifiques aux applications en fonction des accords utilisés sur leur infrastructure réseau.

2.2.1 Critères de la qualité de service

La qualité du service consiste essentiellement à assurer les paramètres suivants :

Débit : Ce paramètre désigne la quantité de données que l'application véhicule pendant un intervalle de temps donné.

Latence : C'est la durée du trajet d'un paquet de sa source jusqu'à sa destination. Elle doit s'approcher autant que possible de zéro pour avoir une meilleure expérience durant l'utilisation des applications au temps réel comme la transmission vidéo, la communication vocale, les applications interactives, les jeux vidéo...etc.

Gigue : Ce paramètre fait référence à la variation de latence des paquets. Le réseau doit respecter ce paramètre au cours de la transmission vocale et de la visioconférence.

Perte de paquets : Corresponds à la non-délivrance d'un paquet de données, la plupart du temps dû à un encombrement du réseau.

2.3 Niveaux de service

Nous avons généralement trois niveaux de QoS :

Best Effort : Best Effort ne fournit aucune distinction entre plusieurs flux réseau et n'autorise aucune garantie. Ce niveau de service est appelé « Manque de qualité de service ».

Soft QoS : Soft QoS (ou DiffServ) peut définir des priorités pour différents flux réseau, mais ne fournissent pas de garanties strictes.

Hard QoS : Hard QoS (ou IntServ), qui consiste à réservé des ressources réseau pour certains types de flux. Le principal mécanisme utilisé pour obtenir ce niveau de service est le RSVP (Resource Reservation Protocol).

2.3.1 IntServ

Le modèle Integrated Services a été la première tentative d'établir un contrôle QoS dans les réseaux IP, c'est une architecture qui spécifie les éléments qui garantissent la qualité de service de bout en bout dans un réseau. Il est proposé pour assurer la QoS au réseau sans perturber le bon fonctionnement du protocole IP, il utilise pour cela un protocole spécifique de signalisation appelé RSVP (Resource Reservation Protocol) [31].

Les composantes clés de cette architecture offrent deux nouveaux services supplémentaires

par rapport au service traditionnel du best-effort :

Le contrôleur de charge : ne garantit pas le délai de bout en bout et la perte de paquets, mais garantit l'absence de congestion du trafic réseau, équivalant au service Best Effort dans un environnement non surchargé.

Le Service Garanti : il garantit une bande passante et un délai d'acheminement limité. Bien qu'IntServ soit un moyen efficace de fournir une garantie Hard QoS, il n'a jamais été considéré comme une solution à long terme pour les exigences QoS du réseau. Cette situation s'explique principalement par les inconvénients suivants d'IntServ :

- Tous les routeurs du Traffic doivent supporter IntServ.
- Tous les routeurs doivent être capables de faire le contrôle d'admission, ainsi que la classification des paquets et leur ordonnancement.
- Chaque routeur doit stocker de nombreux états qui deviennent très coûteux lorsque le réseau prend de l'expansion.

À cause de ces raisons, IntServ n'a jamais été déployé sur de grands réseaux. Cependant, il a été largement utilisé par les réseaux locaux et les petites entreprises. Le principal avantage d'IntServ est qu'il offre une meilleure garantie et peut servir à identifier les classes de service.

En raison de ces nombreux inconvénients, l'IETF a décidé de développer un autre modèle (DiffServ) comme approche alternative de qualité de service avec peu de complexité.

2.3.2 DiffServ

DiffServ suit la philosophie de classement de plusieurs flux en quelques niveaux de services contrairement au IntServ qui offre des services garantis par flux. Cette approche est appelée Class of Services (Cos).

Les composants du réseau utilisent la spécification QoS pour classer, mettre en forme, marquer ou bien établir une politique pour gérer le trafic et pour effectuer une file d'attente intelligente [32].

Les 8 bits du champ TOS (Type of Service) ont été inclus dans le Header du paquet IP pour supporter la classification des paquets comme le montre la figure 2.1. Récemment, un nouveau format et usage de la classification par TOS ont été proposés comme une partie de l'architecture DiffServ [32].

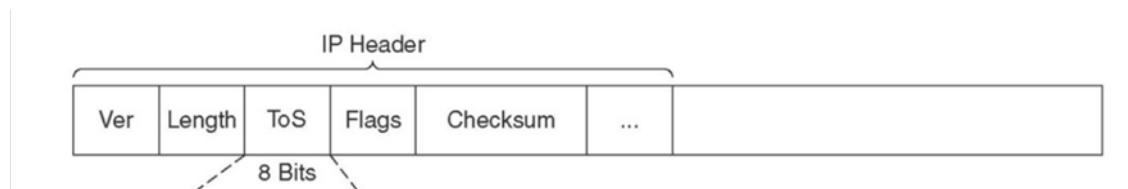


FIGURE 2.1 – Le champ ToS dans l'entête IP

L'octet du TOS a été divisé en 6 bits, réservés pour Differentiated Services Code Point (DSCP) pour le traitement par saut (PHB).

Les 2 autres restent non utilisés (unused field) et seulement réservés pour les notifications de congestion explicites (ECN) comme le montre la figure 2.2 [32].

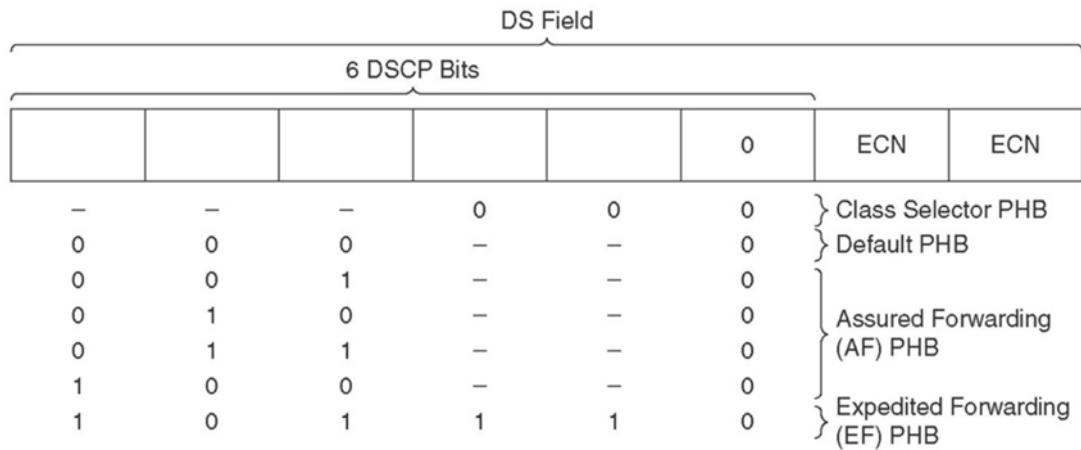


FIGURE 2.2 – Le champ DS et DSCP PHB dans l’entête IP

Le traitement par saut PHB

Par défaut, le PHB spécifie que tous les paquets marqués par la valeur DSCP 000000 vont recevoir un service traditionnel « best effort » qui signifie basiquement que tous les paquets sont traités de la même façon sans d’en donner de priorité à un flux spécifique.

Le champ DSCP qui est constitué de 6 bits, qui donne 64 combinaisons DSCP possibles, dont seulement une partie a été normalisée et sont énumérées dans la figure 2.3.

Expedited forwarding (EF)

Des applications comme la voix sur IP (VoIP), la vidéo et les programmes de trading en ligne nécessitent un type de service robuste qui offre une bande passante garantie. Le expedited forwarding (EF) PHB est un composant clé de DiffServ responsable de fournir ce type de service robuste en offrant une faible perte, une faible latence, une faible gigue et avec une bande passante assurée.

On peut implémenter EF en utilisant le Packet Queueing (PQ) et le Behaviour Aggregator (BA) en même temps. L’implémentation de ce type de PHB fournit une liaison virtuelle dédiée qu’on le considère comme un service premium. Pour utiliser cette dernière d’une façon optimale, elle ne devrait être réservée qu’aux applications les plus critiques parce que, en cas de congestion du trafic, il n’est pas possible de traiter la totalité du trafic comme une priorité élevée [33].

EF PHB est idéal pour les applications qui nécessitent une faible bande passante, une bande passante garantie, un faible délai et une faible gigue.

DSCP Class Name	Binary Value	Decimal Value
BE (best effort, default)	000000	0
AF11 (assured forwarding, see RFC 2597)	001010	10
AF12	001100	12
AF13	001110	14
AF21	010010	18
AF22	010100	20
AF23	010110	22
AF31	011010	26
AF32	011100	28
AF33	011110	30
AF41	100010	34
AF42	100100	36
AF43	100110	38
CS1 (class selector)	001000	8
CS2	010000	16
CS3	011000	24
CS4	100000	32
CS5	101000	40
CS6	110000	48
CS7	111000	56
EF (expedited forwarding, see RFC 2598)	101110	46

FIGURE 2.3 – Les classes DSCP

Class-Selector PHB

Pour préserver la rétrocompatibilité avec tout système de "IP Precedence" actuellement utilisé sur le réseau, DiffServ a défini une valeur DSCP sous la forme « xxx000 » où x prend la valeur "0" ou bien "1". Ces valeurs DSCP sont appelées « Class-Selector Code Points » et illustrées dans la figure 2.4.

(La valeur DSCP pour un paquet avec PHB 000000 par défaut est aussi appelée Class-Selector CodePoint) [32].

Le PHB associé à un Class-Selector Code Point est appelé "PHB Class-Selector". Ces "PHBs

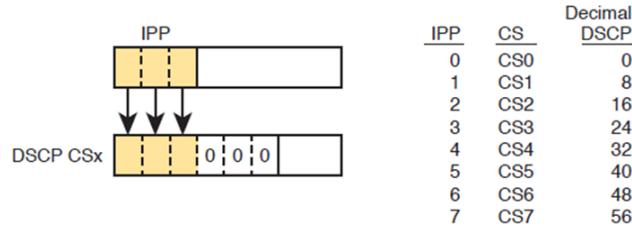


FIGURE 2.4 – class-selector PHB

"Class-Selector" conservent la plupart des comportements de redirection en tant que noeuds qui implémentent la classification et le transfert des paquets en se basant sur "IP Precedence".

Par exemple, les paquets avec la valeur DSCP de 11000 (l'équivalent de la valeur de "IP Precedence" de 110) bénéficient d'un traitement de transfert préférentiel (scheduling, queueing, etc.) par rapport aux paquets avec une valeur DSCP de 100000 (l'équivalent de la valeur de 100 basée sur le "IP Precedence"). Ces PHB de sélecteur de classe garantissent que les noeuds compatibles DS peuvent coexister avec les noeuds basés sur "IP Precedence".

Assured Forwarding (AF)

Assured forwarding (AF) fournit quatre classes différentes de comportements de réacheminement de paquets qu'on peut spécifier au marqueur. Le tableau 2.1 montre :

Les classes, les trois "Precedence" de dépôt qui sont fournis avec chaque classe et les DSSP recommandés qui sont associés à chaque priorité. Chaque DSCP est représenté par sa valeur AF en décimal, en hexadécimal et en binaire [34].

	Class 1	Class 2	Class 3	Class 4
Low-Drop Precedence	AF11=	AF21=	AF31=	AF41=
	10(001010)	18(010010)	26(011010)	34(100010)
Medium- Drop Precedence	AF12=	AF22=	AF32=	AF42=
	12(001100)	20(010100)	28(011100)	36(100100)
High- Drop Precedence	AF13=	AF23=	AF33=	AF43=
	14(001110)	22(010110)	30(011110)	38(100110)

TABLE 2.1 – AF codepoints

Tout système compatible au DiffServ peut utiliser le code AF comme guide pour fournir des comportements de redirection différenciés aux différentes classes de trafic.

Lorsque ces paquets atteignent un routeur diffserv, le routeur évalue les points de code des paquets ainsi que les points de code DS d'un autre trafic dans la file d'attente. Le routeur transmet ou dépose ensuite des paquets, en fonction de la bande passante disponible et des priorités qui sont assignées par les codepoints DS des paquets. Notez que les paquets qui sont marqués avec le PHB EF sont garantis bande passante sur les paquets qui sont marqués avec les différents AF PHBs [34].

La coordination du marquage des paquets entre les systèmes IPQoS du réseau et le routeur DiffServ est nécessaire pour assurer que les paquets sont transmis comme prévu. Par exemple, supposons que les systèmes IPQoS du réseau marquent les paquets avec les points de code AF21 (010010), AF13 (001110), AF43 (100110) et EF (101110). Par la suite, l'ajout des codes AF21, AF13, AF43 et EF DS au fichier approprié sur le routeur diffserv est obligatoire.

Avantages de la mise en œuvre de DiffServ

Les avantages de la mise en œuvre des services différenciés comprennent les suivants :

- Réduit la charge sur les périphériques réseaux et s'adapte facilement à la croissance du réseau.
- Permet aux clients de conserver tout schéma de priorisation ToS de couche 3 existants qui peuvent être utilisés.
- Permet aux clients de mélanger des appareils conformes à DiffServ avec tout équipement compatible ToS existant utilisé.
- Réduit les « Network Bottlenecks » grâce à une gestion efficace des ressources actuelles du réseau d'entreprises [32].

2.4 Application Multimédia

La diversité des types de médias est une caractéristique importante des systèmes d'information modernes, et ce dernier a fait l'objet d'une attention considérable et a donc continué à se développer avec succès. Les systèmes multimédias doivent prendre en charge une variété de types de médias, du texte et des graphiques simples aux plus riches tels que l'animation, l'audio et la vidéo. Nous citons ici des exemples d'applications multimédias.

2.4.1 La vidéo conférence

Les applications multimédias de ce type sont de plus en plus populaires et plus fiables, réduisant la distance lorsque le déplacement n'est pas possible, peu pratique ou indésirable. La visioconférence utilise une infrastructure de télécommunications qui permet la transmission de l'audio et de la vidéo pour rassembler les gens à différents endroits. Compte tenu de l'importance des flux envoyés et reçus par de telles applications, l'application des politiques de QoS est essentielle dans les réseaux à ressources limitées.

2.4.2 La Télésurveillance

C'est une application qui nous permet de surveiller en temps réel des sites distants pour des raisons de sécurité. Compte tenu de son importance, la télésurveillance est devenue une partie intégrante de nombreuses organisations, entreprises et utilisateurs. Elle repose sur un système d'alarme relié à une centrale de surveillance qui assure une sécurité optimale de l'habitation à distance, en prévenant les cambriolages, les intrusions et les vols, ainsi que les incendies et les inondations de l'habitation. Cette application est appelée en utilisant à la fois la vidéo et

l’audio, ou la vidéo uniquement, selon les besoins. Comme dans le cas de la vidéo-conférence, la surveillance à distance nécessite la mise en place de politiques de QoS, et ainsi, de nombreux travaux ont été proposés dans ce sens [35].

2.4.3 La vidéo à la demande

La Vidéo à la demande permet de regarder de la vidéo en temps réel sur Internet. Ce type de diffusion s’appuie soit sur des infrastructures de communication à très haut débit (des fibres optiques par exemple), soit sur des techniques de streaming. Cependant, les services qui utilisent Internet sont confrontés à des problèmes de qualité de service en raison de l’instabilité et de la congestion du réseau. En effet, les performances se dégradent lorsque le contenu est servi à un grand nombre de consommateurs. Par conséquent, un mécanisme de mise en œuvre de la QoS est essentiel. Ces quelques exemples d’applications multimédia illustrent les multiples contraintes auxquelles ces applications sont soumises, en particulier la responsabilité de respecter les paramètres de QoS tels que le délai, la gigue et la perte de paquets. Si le réseau présente des carences en termes de ressources (bande passante, non prise en compte de la taille des équipements du réseau, congestion du réseau, perte de paquets, etc.), ces applications ne pourront pas fournir un service de qualité satisfaisant l’utilisateur final et deviennent ainsi inutilisables. Il est donc impératif de mettre en place des mécanismes permettant de gérer la QoS du réseau qui délivrera les flux résultants.

2.5 Conclusion

Dans ce chapitre, on a vu la QoS en général, sa définition, son besoin, ainsi que la définition de ses métriques (disponibilité de la bande passante, taux de perte de paquets, et la latence), nous avons vu aussi les différentes architectures de mise en œuvre de la QoS, les applications multimédias et leurs types.

Chapitre 3

QoS dans SDN : Implémentation et résultats

Chapitre 3

QoS dans SDN : Implémentation et résultats

3.1 Introduction

L'objectif réel du SDN est de rendre le réseau programmable via un contrôleur centralisé. Un contrôleur central permet aux ingénieurs de réseau de mettre en place des stratégies de transfert uniques et flexibles, dont la seule limite est la capacité d'exécution du logiciel du contrôleur.

Les chapitres précédents ont établi le contexte du travail présenté dans ce projet, qui consiste à implémenter la méthode DiffServ dans un environnement SDN qui utilise Ryu comme contrôleur pour gérer la topologie réalisée, pour d'éventuels tests de QoS pour le multimédia streaming. Pour réaliser ça ,nous avons créé deux scénarios dans lesquels le premier scénario est exécuté sans la QoS et dans le deuxième scénario nous avons implanté les capacités OpenFlow avec QoS à l'aide de services différenciés (DS). Tout d'abord, on utilisera l'expérimentation, puis des tests pour comprendre comment le contrôle de la bande passante,la gigue et les pertes de paquets par DS fonctionnent pour garantir la QoS souhaitée. Enfin, après chaque test de discussion, nous donnerons notre avis sur la méthode QoS implantée pour évaluer le flux multimédia streaming.

3.2 Mininet

Pour émuler une infrastructure réseau complète dans l'environnement SDN, l'émulateur de réseau Mininet est utilisé [36]. Il s'agit de l'outil principal pour les environnements de tests

de SDN pour concevoir, subir et vérifier les projets OpenFlow. Mininet offre un haut niveau de flexibilité puisque les topologies et les nouvelles fonctionnalités sont programmées à l'aide du langage python. Il fournit également un environnement de prototypage extensible, capable de gérer jusqu'à 4000 commutateurs sur un ordinateur ordinaire. Ceci est possible grâce à une fonction de virtualisation au niveau du système d'exploitation, ainsi qu'à l'inclusion de processus et d'espaces de noms de réseau, qui permettent de produire des instances complètement différentes et séparées d'interfaces réseau et de tables de routage qui fonctionnent généralement indépendamment les unes des autres.

Mininet permet principalement de créer le réseau virtuel à flux ouvert qui comprend un contrôleur SDN, des commutateurs logiciels OpenFlow, plusieurs hôtes et des liens dans une machine réelle ou virtuelle. Il fonctionne sur les différents systèmes d'exploitation tels que Mac OS, Windows et Linux. Pour la partie Recherche, Linux est préférable. Bien que Mininet permette la création de réseaux personnalisés de choix, il impose certaines limitations par rapport aux réseaux physiques réels. Mininet ne convient pas aux expériences jusqu'à 10 Gbps. Dans ce cas, le réseau physique offre généralement une meilleure expérience.

3.3 Workload

Pour ce projet, l'exactitude de la mesure est un objectif obligatoire. Le besoin est de générer un trafic de flux marqué d'une valeur DSCP. Par conséquent, les outils suivants ont été sélectionnés pour la mesure de la bande passante et le jitter, ainsi que l'analyse du trafic sous différents paramètres de QoS :Wireshark,iPerf3 [37].

3.3.1 iPerf

Iperf est un outil très utile pour mesurer la bande passante maximale disponible entre deux noeuds. Il est utilisé dans les connexions TCP (Transport Control Protocol) et UDP (User Datagram Protocol), par le biais de la modulation de différents paramètres.

L'outil obtient ses statistiques en fonction du nombre de paquets transmis pendant la durée et l'intervalle de transmission. Il se compose de deux éléments, un client et un serveur. Le client génère du trafic vers le serveur en fonction de la spécification de trafic requise et détermine la bande passante de transmission moyenne, tandis que le serveur reçoit le trafic généré, établit des statistiques et renvoie les résultats au client. iPerf fonctionne au niveau de l'application, ce qui signifie que le débit mesuré n'exclut pas les performances de la liaison, mais inclut également les paquets traités via le modèle TCP/IP. De plus, iPerf s'exécute au niveau de l'espace utilisateur du système d'exploitation Linux, fonctionne au-dessus de l'architecture système et utilise des appels système pour utiliser les ressources.

Le travail suivant peut être effectué en utilisant Iperf :

- Mesurer la bande passante dans un réseau client-serveur, le client génère un flux UDP, avec une bande passante spécifique (BW).

- Mesurer la perte de paquets.
- Mesurer la gigue.
- Travailleur dans un environnement multicast.

3.3.2 Wireshark

Wireshark est un analyseur de paquets réseau commun qui peut être utilisé sur le contrôleur ou l'hôte Mininet pour examiner correctement les messages d'échange OpenFlow entre le contrôleur et les commutateurs individuels. Il capture adéquatement un paquet dans le réseau pour montrer instantanément ses informations de couche TCP/IP aussi détaillée que possible, permettant ainsi d'examiner son contenu explicite à des fins telles que le dépannage du réseau, les examens de sécurité, le débogage de protocoles et l'apprentissage des protocoles réseau.

Cet outil permet de capturer correctement les paquets en direct d'une interface réseau (physique ou virtuelle), en affichant les informations sur les paquets avec des informations élaborées sur le protocole, et en sauvegardant toutes les captures de paquets pour des études supplémentaires et des statistiques fiables.

Les informations capturées peuvent être analysées selon différents critères, dans lesquels elles peuvent être modifiées par des horodatages, des ports TCP/UDP, des protocoles, des sessions TCP, etc. Comme Wireshark reste un outil efficace qui fonctionne en espace utilisateur, il utilise la bibliothèque pcap pour lui permettre de capturer des paquets à un niveau inférieur. Cela permet généralement capturer des paquets avec un horodatage plus précis puisqu'il n'y a pas de retard supplémentaire causé par le processus de communication interne entre l'utilisateur et le serveur. Cette recherche a utilisé le dissecteur Wireshark fourni par un paquet Mininet, qui permet au filtre OpenFlow de capturer les messages OpenFlow et d'observer en détail leur format., ainsi que les entrées de flux et de groupe.

3.3.3 VLC

VLC (VideoLAN Client) est un logiciel multimédia qui permet de lire la grande majorité des formats vidéo (MPEG, AVI, MP4, MOV, WMV, MIDI...) et audio (MP3, AAC, FLAC, WMA, AC3...), grâce aux multiples codecs qu'il intègre [38]. Le logiciel VLC possède l'avantage d'être disponible sur plusieurs les systèmes d'exploitation tels que Windows, Mac OS, Linux, Android, Chrome OS, iOS, et bien d'autres systèmes d'exploitation basés sur UNIXb [38] . VLC peut également être utilisé pour envoyer et recevoir des flux multimédias réseaux via d'autres protocoles de diffusion comme TCP, UDP, HTTP, FTP [38], ce qui va être exploité dans ce travail pour diffuser un flux multimédia depuis un hôte considéré comme serveur vers un autre hôte pris comme client. Il est capable de lire des vidéos incomplètes, endommagées ou qui encore en cours de téléchargement .ce dernier point est considéré très utile car il permet l'évaluation de la qualité du flux multimédia dans le cas où il y aurait des pertes de paquets.

3.4 Implémentation de la topologie Mininet

Le contrôleur SDN, Ryu, est conçu pour fonctionner sur l'ordinateur hôte avec une connexion TCP à la topologie de réseau Mininet émulée. Les versions détaillées des logiciels sont présentées dans le tableau 3.1.

TABLE 3.1 – Exigences du test

	Nom	Spécification
1	Système exploitation	Ubuntu 20.04.3 LTS (64bits)
2	Ryu Controller	Version 4.34
3	Mininet Emulateur	Version 2.3.1b1
4	OpenFlow Protocol	Version 1.3

La structure de l'implémentation du test est illustrée à la Figure 3.1.

Tout d'abord, on exécute une topologie de réseau simple, comme le montre la Figure 3.1, composée d'un client h2 et un serveur h1.

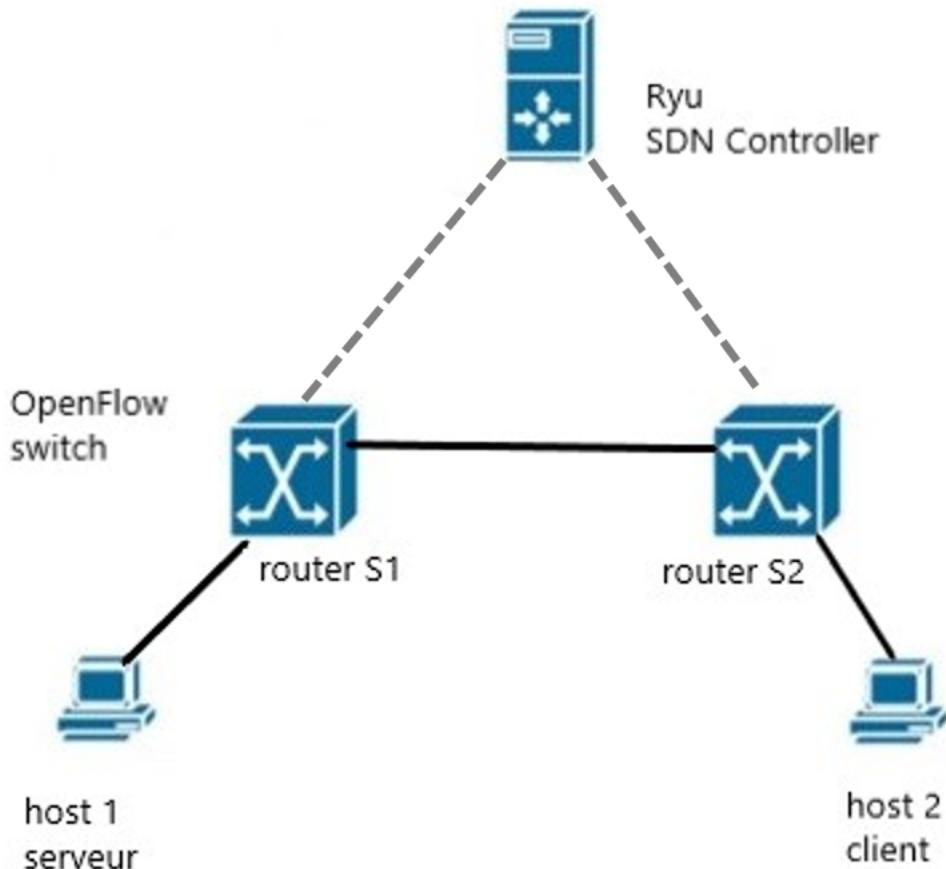


FIGURE 3.1 – Topologie réseaux créée

Il existe un chemin pour communiquer entre les clients et le serveur. Par conséquent, le chemin (s1, s2) est le lien de communication direct et constitue le saut minimal.

La topologie dans mininet est créé avec cette commande :

```
sudo mn --topo linear,2 --mac --switch ovsk --controller remote -x --link tc,bw=10
```

Après le lancement de cette commande, le CLI dévolue la sortie montrée dans la Figure 3.2.

```
bouhalissa@bouhalissa-VirtualBox:~$ sudo mn --topo linear,2 --mac --switch ovsk --controller remote -x --link tc,bw=10
[sudo] password for bouhalissa:
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2
*** Adding links:
(10.00Mbit) (10.00Mbit) (h1, s1) (10.00Mbit) (10.00Mbit) (h2, s2) (10.00Mbit) (10.00Mbit) (s2, s1)
*** Configuring hosts
h1 h2
*** Running terms on :
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ... (10.00Mbit) (10.00Mbit) (10.00Mbit)
*** Starting CLI:
mininet> 
```

FIGURE 3.2 – Topologie de réseau créé

La commande Ping est utilisée pour vérifier la connectivité entre les appareils. Dans Mininet, il est possible d'utiliser la commande pingall, qui effectue un ping de toutes les paires. Cette commande vérifie que les liens créés fonctionnent. La figure 3.3 ci-dessous montre que lorsque l'on exécute la première fois la commande **pingall**, le premier hôte n'a pas de connectivité avec les autres hôtes, alors que tous les autres liens sont bons.

```
mininet> nodes
available nodes are:
c0 h1 h2 s1 s2
mininet> pingall
*** Ping: testing ping reachability
h1 -> X
h2 -> X
*** Results: 100% dropped (0/2 received)
mininet> 
```

FIGURE 3.3 – Pingall non réussie

Pour faire fonctionner la topologie on doit d'abord configurer les adresses des hôtes ainsi que le routage et définir la version d'OpenFlow à utiliser dans chaque routeur à la version 1.3 et configurer pour écouter sur le port 6632 pour accéder à OVSDB.

Tout d'abord, on donne une Add IP à h1 (172.16.20.10/24) et à h2 (172.16.10.10/24) comme le montre la figure 3.4.

```
"host: h2"
root@rimynouto-VirtualBox:/home/rimynouto# ip addr del 10.0.0.2/8 dev h2-eth0
root@rimynouto-VirtualBox:/home/rimynouto# ip addr add 172.16.10.10/24 dev h2-e
th0
root@rimynouto-VirtualBox:/home/rimynouto# [REDACTED]

"host: h1"
root@rimynouto-VirtualBox:/home/rimynouto# ip addr del 10.0.0.1/8 dev h1-eth0
root@rimynouto-VirtualBox:/home/rimynouto# ip addr add 172.16.20.10/24 dev h1-e
th0
root@rimynouto-VirtualBox:/home/rimynouto# [REDACTED]
```

FIGURE 3.4 – Configuration des adresses IP de h1 et h2

Ensuite, on configure la version OpenFlow à utiliser dans chaque routeur comme le montre la figure 3.5.

```
"switch: s2" (root)
root@rimynouto-VirtualBox:/home/rimynouto# ovs-vsctl set Bridge s2 protocols=OpenFlow13
root@rimynouto-VirtualBox:/home/rimynouto# [REDACTED]

"switch: s1" (root)
root@rimynouto-VirtualBox:/home/rimynouto# ovs-vsctl set Bridge s1 protocols=OpenFlow13
root@rimynouto-VirtualBox:/home/rimynouto# ovs-vsctl set-manager ptcp:6632
root@rimynouto-VirtualBox:/home/rimynouto# [REDACTED]
```

FIGURE 3.5 – Configuration des routeurs OpenFlow.

Avant de configurer les routes, on doit commencer par activer le contrôleur (figure 3.6) qui effectue le routage (qui est pour l'instant désactivé). On l'active grâce à la commande :

```
$ryu-manager ryu.app.rest_qos ryu.app.qos_rest_router ryu.app.rest_conf_switch
```

```
"controller: c0" (root)
root@bouhalissa-VirtualBox:/home/bouhalissa/ryu# ryu-manager ryu.app.rest_qos ryu.app.qos_rest_router ryu.app.rest_conf_switch
loading app ryu.app.rest_qos
loading app ryu.app.qos_rest_router
loading app ryu.app.rest_conf_switch
loading app ryu.controller.ofp_handler
instantiating app None of IPSet
creating context dpset
instantiating app None of ConfSwitchSet
creating context conf_switch
creating context wsgi
instantiating app ryu.app.rest_qos of RestQoSAPI
instantiating app ryu.app.qos_rest_router of RestRouterAPI
instantiating app ryu.app.rest_conf_switch of ConfSwitchAPI
instantiating app ryu.controller.ofp_handler of OFPHandler
(5718) wsgi starting up on http://0.0.0.0:8080
[QoS][INFO] dpid=0000000000000001: Join qos switch.
[RT][INFO] switch_id=0000000000000001: Set SW config for TTL error packet in.
[RT][INFO] switch_id=0000000000000001: Set ARP handling (packet in) flow [cookie
[RT][INFO] switch_id=0000000000000001: Set L2 switching (normal) flow [cookie=0x0
[RT][INFO] switch_id=0000000000000001: Set default route (drop) flow [cookie=0x0
[RT][INFO] switch_id=0000000000000001: Start cyclic routing table update.
[RT][INFO] switch_id=0000000000000001: Join as router.
[QoS][INFO] dpid=0000000000000002: Join qos switch.
[RT][INFO] switch_id=0000000000000002: Set SW config for TTL error packet in.
[RT][INFO] switch_id=0000000000000002: Set ARP handling (packet in) flow [cookie
[RT][INFO] switch_id=0000000000000002: Set L2 switching (normal) flow [cookie=0x0
[RT][INFO] switch_id=0000000000000002: Set default route (drop) flow [cookie=0x0
[RT][INFO] switch_id=0000000000000002: Start cyclic routing table update.
[RT][INFO] switch_id=0000000000000002: Join as router.
```

FIGURE 3.6 – Activation Controller Ryu

Pour que la topologie fonctionne, on réalise la configuration de toutes les routes comme le montre la figure 3.7.

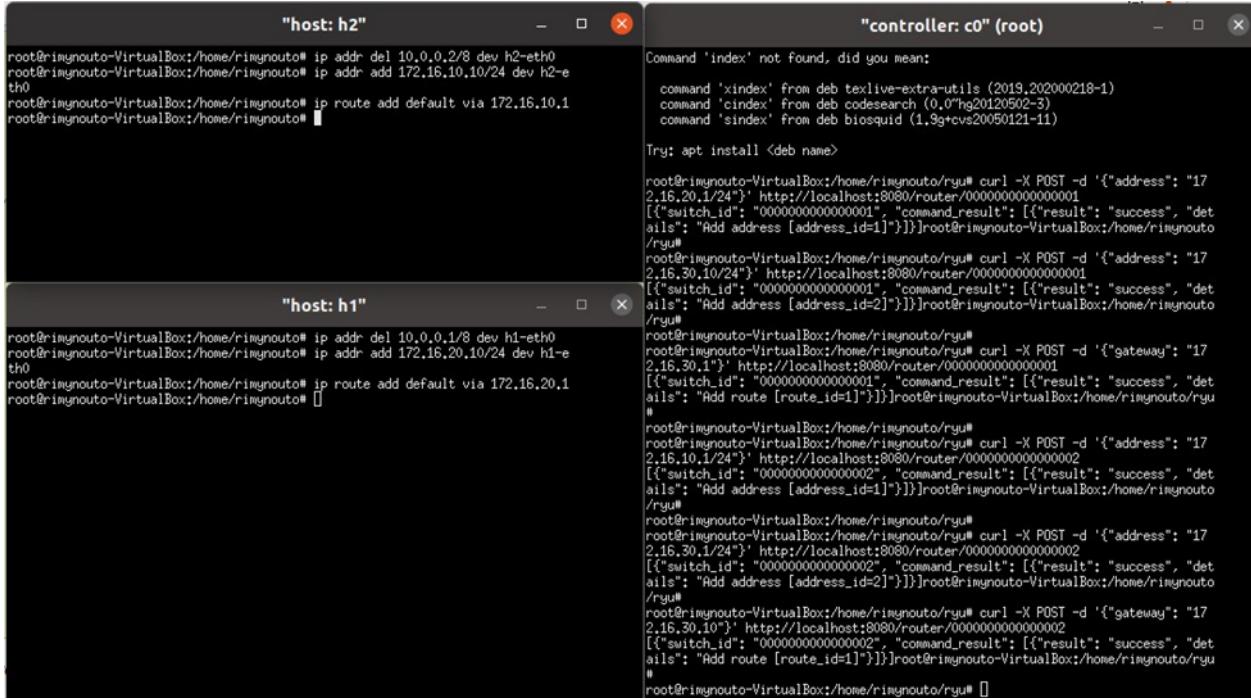


FIGURE 3.7 – Configuration des routes.

Maintenant, la topologie fonctionne comme le montre la figure 3.8 ,on peut commencer à implémenter les files d'attente et la QoS.

```

mininet> h1 ping h2
PING 172.16.10.10 (172.16.10.10) 56(84) bytes of data.
64 bytes from 172.16.10.10: icmp_seq=1 ttl=62 time=0.384 ms
64 bytes from 172.16.10.10: icmp_seq=2 ttl=62 time=0.102 ms
64 bytes from 172.16.10.10: icmp_seq=3 ttl=62 time=0.077 ms
^C
--- 172.16.10.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
rtt min/avg/max/mdev = 0.077/0.187/0.384/0.139 ms
mininet> h2 ping h1
PING 172.16.20.10 (172.16.20.10) 56(84) bytes of data.
64 bytes from 172.16.20.10: icmp_seq=1 ttl=62 time=0.101 ms
64 bytes from 172.16.20.10: icmp_seq=2 ttl=62 time=0.086 ms
64 bytes from 172.16.20.10: icmp_seq=3 ttl=62 time=0.064 ms
c64 bytes from 172.16.20.10: icmp_seq=4 ttl=62 time=0.075 ms
^?64 bytes from 172.16.20.10: icmp_seq=5 ttl=62 time=0.100 ms
^C
--- 172.16.20.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4102ms
rtt min/avg/max/mdev = 0.064/0.085/0.101/0.014 ms
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> 

```

FIGURE 3.8 – Ping réussie

3.5 Expérimentation

L'organisation de la section expérimentale est la suivante. Chaque première partie présente un énoncé de l'expérience qui explique ce que nous voulons accomplir. Ensuite, la mise en œuvre de cette expérience est expliquée en détail. Les résultats expérimentaux sont ensuite présentés. Chaque section se termine par une note sur les observations des expériences réalisées. Dans toutes les expériences, voici les valeurs de charge de travail utilisées :

- Numéro de port utiliser 5001,5002,5003,5004.
- Modifier l'intervalle de capture des flux -i (temps de capture).
- Modifier le temps du test -t (valeur de temps en secondes).
- UDP taille du paquet 208 KByte par défaut.
- Débit entre les liens 10 Mbps maximum.

3.5.1 Implémentation de DiffServ

Dans cette partie, nous allons implémenter la méthode DiffServ dans un environnement SDN. Tout d'abord, nous devons nous assurer que Ryu peut interagir avec les fonctionnalités QoS. Une fois la QoS créée dans le contrôleur Ryu, il l'utilisera pour créer une entrée dans la table de flux OpenvSwitch, puis identifiera le flux prioritaire en lui attribuant la priorité souhaitée.

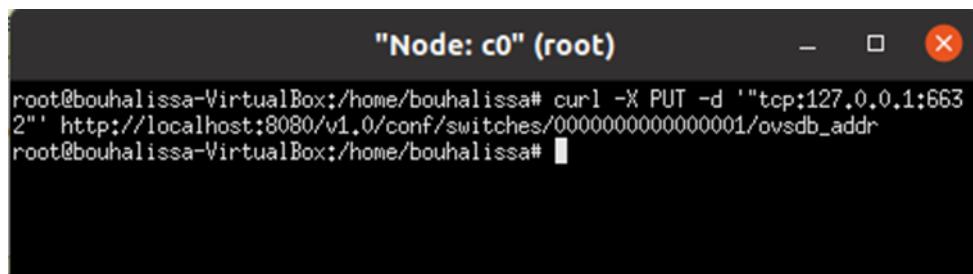
Pour commencer on active les fonctionnalités nécessaires sur notre contrôleur pour qu'il accepte une commande complexe contenant le DSCP, il suffit d'entrer ces lignes sur la CLI Ryu :

- Apt-get update
- Apt install curl

Maintenant que Ryu est prêt à recevoir notre code de QoS, nous pouvons choisir une approche pour l'injecter aux OpenvSwitch

Méthode d'injection de la QoS

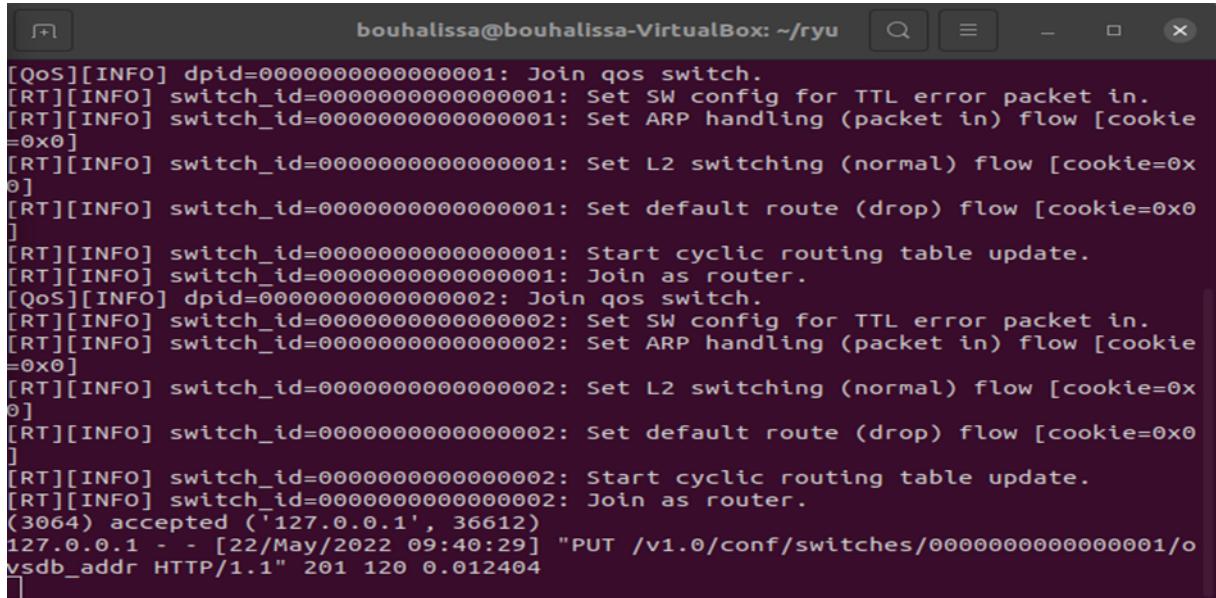
On commence d'abord par configurer l'adresse OVSDB pour pouvoir accéder au switch s1 à partir du contrôleur RYU. La commande utilisée est montrée dans la figure 3.9 suivante :



```
"Node: c0" (root)
root@bouhalissa-VirtualBox:/home/bouhalissa# curl -X PUT -d '{"tcp:127.0.0.1:6632"}' http://localhost:8080/v1.0/conf/switches/0000000000000001/ovsdb_addr
root@bouhalissa-VirtualBox:/home/bouhalissa#
```

FIGURE 3.9 – Configuration d'adresse OVSDB

L'adresse IP du switch OVSDB est implémentée avec succès comme il est montré dans la figure 3.10 ci-dessus :



```
[QoS][INFO] dpid=0000000000000001: Join qos switch.
[RT][INFO] switch_id=0000000000000001: Set SW config for TTL error packet in.
[RT][INFO] switch_id=0000000000000001: Set ARP handling (packet in) flow [cookie=0x0]
[RT][INFO] switch_id=0000000000000001: Set L2 switching (normal) flow [cookie=0x0]
[RT][INFO] switch_id=0000000000000001: Set default route (drop) flow [cookie=0x0]
[RT][INFO] switch_id=0000000000000001: Start cyclic routing table update.
[RT][INFO] switch_id=0000000000000001: Join as router.
[QoS][INFO] dpid=0000000000000002: Join qos switch.
[RT][INFO] switch_id=0000000000000002: Set SW config for TTL error packet in.
[RT][INFO] switch_id=0000000000000002: Set ARP handling (packet in) flow [cookie=0x0]
[RT][INFO] switch_id=0000000000000002: Set L2 switching (normal) flow [cookie=0x0]
[RT][INFO] switch_id=0000000000000002: Set default route (drop) flow [cookie=0x0]
[RT][INFO] switch_id=0000000000000002: Start cyclic routing table update.
[RT][INFO] switch_id=0000000000000002: Join as router.
(3064) accepted ('127.0.0.1', 36612)
127.0.0.1 - - [22/May/2022 09:40:29] "PUT /v1.0/conf/switches/0000000000000001/o
vsdb_addr HTTP/1.1" 201 120 0.012404
```

FIGURE 3.10 – Résultat de l'implémentation.

Une fois le switch OVSDB est connecté correctement avec le contrôleur RYU, alors on passe à la création des files d'attente sur l'interface s1-eth1 (s1-eth1 est relié entre s1 et h1)

TABLE 3.2 – Les filles d'attente

Queue ID	Débit Max	Débit Min	Classe	Nom du flux
0	10 Mbps	-	Best effort 0	Standard
1	10 Mbps	6 Mbps	AF41	Multimédia streaming
2	10 Mbps	4 Mbps	AF31	Multimédia Conferencing
3	10 Mbps	3 Mbps	AF21	Faible latence/Data

Le tableau 3.2 présente la configuration des files d'attente qu'on va utiliser dans cette manipulation :

Queue ID=0 : Best effort, c'est la classe par défaut. On va lui attribuer un débit max de 10M.
 Queue ID=1 : c'est pour la classe AF41 qui va avoir un débit max de 10M et débit min de 6M.
 Queue ID=2 : c'est pour la classe AF31. Cette dernière possède une priorité moins élevée que celle de la classe AF41. On va lui attribuer un débit max égal à 10M et un débit min de 4M.
 Queue ID=3 : c'est pour la classe AF21 à qui on va lui donner toujours un débit max de 10M et un débit minimum de 3M.

La figure 3.11 ci dessus montre la création des files d'attentes.

```

"Node: c0" (root)
root@bouhalissa-VirtualBox:/home/bouhalissa# curl -X POST -d '{"port_name": "s1-eth1", "type": "linux-htb", "max_rate": "10000000", "queues": [{"max_rate": "10000000"}, {"min_rate": "6000000"}, {"min_rate": "4000000"}, {"min_rate": "3000000"}]} http://localhost:8080/qos/queues/0000000000000001
[{"switch_id": "0000000000000001", "command_result": {"result": "success", "details": {"0": {"config": {"max-rate": "10000000"}}, "1": {"config": {"min-rate": "6000000"}}, "2": {"config": {"min-rate": "4000000"}}, "3": {"config": {"min-rate": "3000000"}}}}]root@bouhalissa-Vi
root@bouhalissa-VirtualBox:/home/bouhalissa#

```

FIGURE 3.11 – Création fils d'attentes

Après avoir configurer la liste des filles d'attente avec succès on passe vers l'étape suivante qui est de créer la QoS dans le router S1 comme montrer dans le tableau 3.3 si dessus.

TABLE 3.3 – Configuration de la QoS

Priorité	DSCP	Queue ID	QOS ID
1	34 (AF41)	1	1
1	26 (AF31)	2	2
1	18 (AF21)	3	3

Dans cette partie, on va classer les flux marqués avec la valeur 34 (AF41) dans « Queue 1 » et les paquets marques avec une valeur DSCP égale a 26 (AF31) vont être classer dans « Queue 2 ».et pour finir les paquets 18 (AF21) vont être classe dans « Queues 3 », le reste est le BEST EFFORT « Queues 0 ».

Une fois notre QoS créé, on doit l'appliquer au flux dans les deux sens entre les machines qu'en veut prioriser par rapport aux autres. Pour cela on utilise la commande « curl » suivante sur la CLI du contrôleur Ryu (figure 3.12).

```

"Node: c0" (root)
root@bouhalissa-VirtualBox:/home/bouhalissa# curl -X POST -d '{"match": {"ip_dscp": "34"}, "actions": {"queue": "1"}}' http://localhost:8080/qos/rules/0000000000000001
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "QoS added. : qos_id=1"}}]]root@bouhalissa-VirtualBox:/home/bouhalissa#
root@bouhalissa-VirtualBox:/home/bouhalissa#
root@bouhalissa-VirtualBox:/home/bouhalissa# curl -X POST -d '{"match": {"ip_dscp": "26"}, "actions": {"queue": "2"}}' http://localhost:8080/qos/rules/0000000000000001
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "QoS added. : qos_id=2"}}]]root@bouhalissa-VirtualBox:/home/bouhalissa#
root@bouhalissa-VirtualBox:/home/bouhalissa#
root@bouhalissa-VirtualBox:/home/bouhalissa# curl -X POST -d '{"match": {"ip_dscp": "18"}, "actions": {"queue": "3"}}' http://localhost:8080/qos/rules/0000000000000001
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "QoS added. : qos_id=3"}}]]root@bouhalissa-VirtualBox:/home/bouhalissa#
root@bouhalissa-VirtualBox:/home/bouhalissa#

```

FIGURE 3.12 – Configuration QoS sur le routeur S1

Une fois la configuration du routeur s1 est terminée on passe à la configuration du routeur s2 comme le montre le tableau 3.4 suivant :

TABLE 3.4 – Le marquage DSCP sur le Routeur S2

Priorité	ADD destination	Port destination	Protocol	DSCP	QOS ID
1	172.16.10.20	5002	UDP	34 (AF41)	1
1	172.16.10.20	5003	UDP	26 (AF31)	2
1	172.16.10.20	5004	UDP	18 (AF21)	3

Dans cette partie, on va configurer le routeur 2 (s2) pour envoyer les flux marqués avec les valeurs DSCP 34 et 26 et 18 vers les ports 5002,5003 et 5004 du h1 (@ip 172.16.20.10) (comme représenter dans la figure 3.13).

```

"Node: c0" (root)
root@bouhalissa-VirtualBox:/home/bouhalissa# curl -X POST -d '{"match": {"nw_dst": "172.16.20.10", "nw_proto": "UDP", "tp_dst": "5002"}, "actions": [{"mark": "34"}]} http://localhost:8080/qos/rules/0000000000000002
[{"switch_id": "0000000000000002", "command_result": [{"result": "success", "details": "QoS added. : qos_id=1"}}}]root@bouhalissa-VirtualBox:/home/bouhalissa#
root@bouhalissa-VirtualBox:/home/bouhalissa#
root@bouhalissa-VirtualBox:/home/bouhalissa# curl -X POST -d '{"match": {"nw_dst": "172.16.20.10", "nw_proto": "UDP", "tp_dst": "5003"}, "actions": [{"mark": "26"}]} http://localhost:8080/qos/rules/0000000000000002
[{"switch_id": "0000000000000002", "command_result": [{"result": "success", "details": "QoS added. : qos_id=2"}}}]root@bouhalissa-VirtualBox:/home/bouhalissa#
root@bouhalissa-VirtualBox:/home/bouhalissa# curl -X POST -d '{"match": {"nw_dst": "172.16.20.10", "nw_proto": "UDP", "tp_dst": "5004"}, "actions": [{"mark": "18"}]} http://localhost:8080/qos/rules/0000000000000002
[{"switch_id": "0000000000000002", "command_result": [{"result": "success", "details": "QoS added. : qos_id=3"}}}]root@bouhalissa-VirtualBox:/home/bouhalissa#
root@bouhalissa-VirtualBox:/home/bouhalissa# 

```

FIGURE 3.13 – Configuration QoS sur le routeur S2

Après avoir fini la partie configuration on peut vérifier si la QoS et Queue setting ont été implémentées avec succès en injectant la commande suivante :

« curl -X GET http ://localhost :8080/qos/rules/0000000000000001 pour le routeur S1».
« curl -X GET http ://localhost :8080/qos/rules/0000000000000002 pour le routeur S2».

```

"Node: c0" (root)
root@bouhalissa-VirtualBox:/home/bouhalissa# curl -X POST -d '{"match": {"nw_dst": "172.16.20.10", "nw_proto": "UDP", "tp_dst": "5002"}, "actions": [{"mark": "34"}]} http://localhost:8080/qos/rules/0000000000000002
[{"switch_id": "0000000000000002", "command_result": [{"result": "success", "details": "QoS added. : qos_id=1"}}}]root@bouhalissa-VirtualBox:/home/bouhalissa#
root@bouhalissa-VirtualBox:/home/bouhalissa#
root@bouhalissa-VirtualBox:/home/bouhalissa# curl -X POST -d '{"match": {"nw_dst": "172.16.20.10", "nw_proto": "UDP", "tp_dst": "5003"}, "actions": [{"mark": "26"}]} http://localhost:8080/qos/rules/0000000000000002
[{"switch_id": "0000000000000002", "command_result": [{"result": "success", "details": "QoS added. : qos_id=2"}}}]root@bouhalissa-VirtualBox:/home/bouhalissa#
root@bouhalissa-VirtualBox:/home/bouhalissa# curl -X POST -d '{"match": {"nw_dst": "172.16.20.10", "nw_proto": "UDP", "tp_dst": "5004"}, "actions": [{"mark": "18"}]} http://localhost:8080/qos/rules/0000000000000002
[{"switch_id": "0000000000000002", "command_result": [{"result": "success", "details": "QoS added. : qos_id=3"}}}]root@bouhalissa-VirtualBox:/home/bouhalissa#
root@bouhalissa-VirtualBox:/home/bouhalissa# 

```

FIGURE 3.14 – Vérification des QOS et Queue settings

Nous remarquons dans la dernière Figure que nos flux ont bien été créés dans la table des flux des OpenvSwitch, la priorité du flux (1) a été automatiquement fixée par notre contrôleur.

3.6 1er Scénario : Avant l'implémentation de la QoS

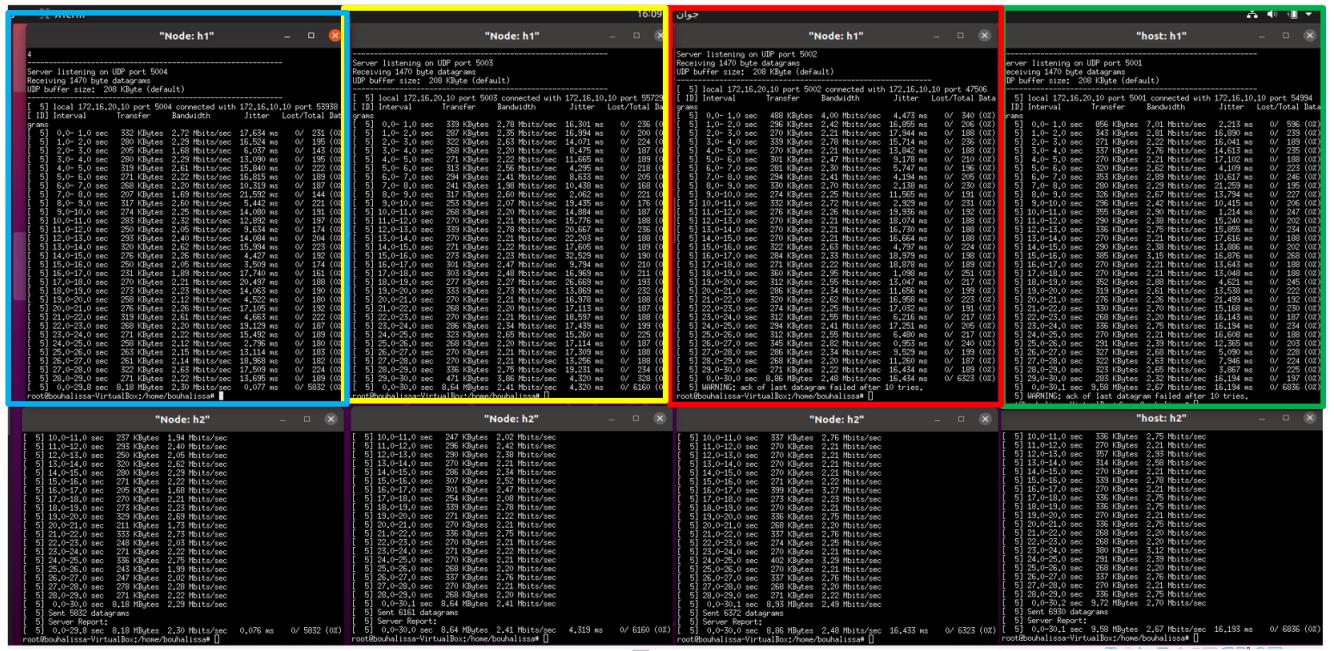


FIGURE 3.15 – Avant implémentation de QoS

La figure 3.15 représente la bande passante et jitter et paquet loss en fonction du temps (capturés chaque seconde pendant 30 seconde), suite à la génération d'un trafic UDP allant de H2 au H1 sans appliquer une politique QoS avec congestion.

Constats (avis) :

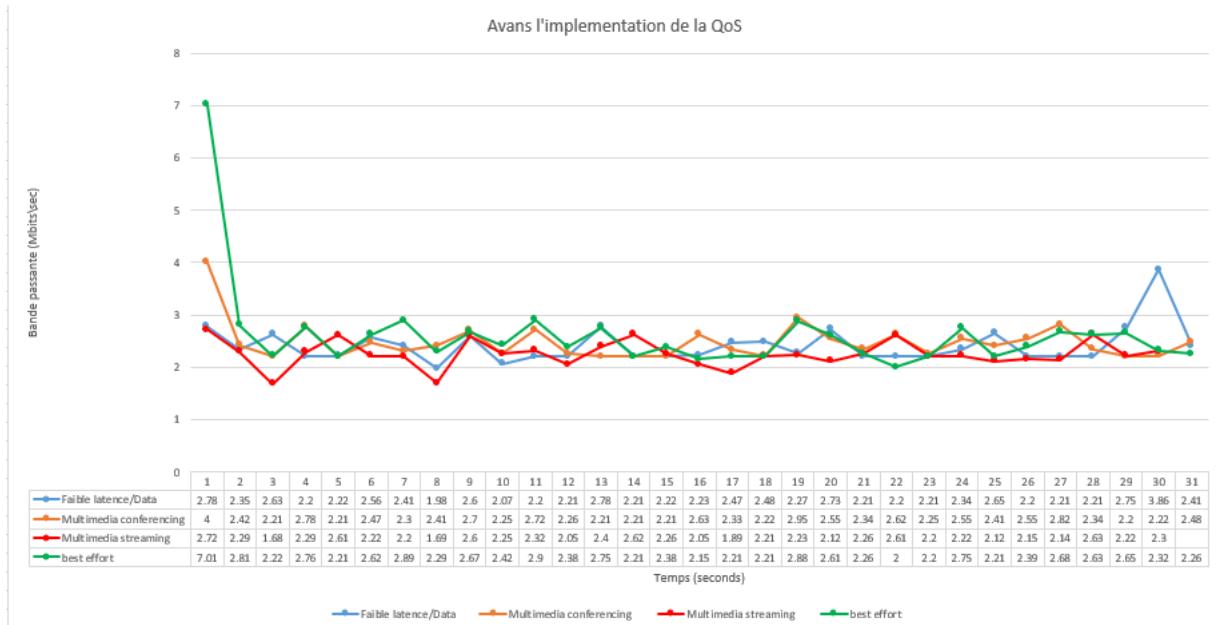


FIGURE 3.16 – Avant implémentation de QoS

Comme le montre la figure 3.15 et 3.16 le test a duré 30 secondes. Le graphe vert représente la bande passante et le jitter entre H1 et H2, pour le flux data (best effort) ou on a obtenu une moyenne de la bande passante de 2.67 mb/s et une moyenne de jitter de 16.193 ms.

Le graphe rouge représente la bande passante et le jitter pour le flux multimédia streaming, ou on a obtenu une moyenne de bande passante de 2.48 mb/s, et une moyenne de jitter de 16.433 ms.

Le graphe jaune représente la bande passante et le jitter pour le flux multimédia conférence, ou on a obtenu une moyenne de bande passante de 2.41 mb/s, et une moyenne de jitter de 16.279 ms.

Le graphe bleu représente la bande passante et le jitter pour le flux de Faible latence/Data, ou on a obtenu une moyenne de bande passante de 2.30 mb/s, et une moyenne de jitter de 16.587 ms.

Avec tous ces trafics qui se disputent la bande passante, aucun d'entre eux n'a bénéficié d'une priorité quelconque en termes de bande passante et jitter.

3.7 2ème Scénario : Apres l'implémentassions de la QoS

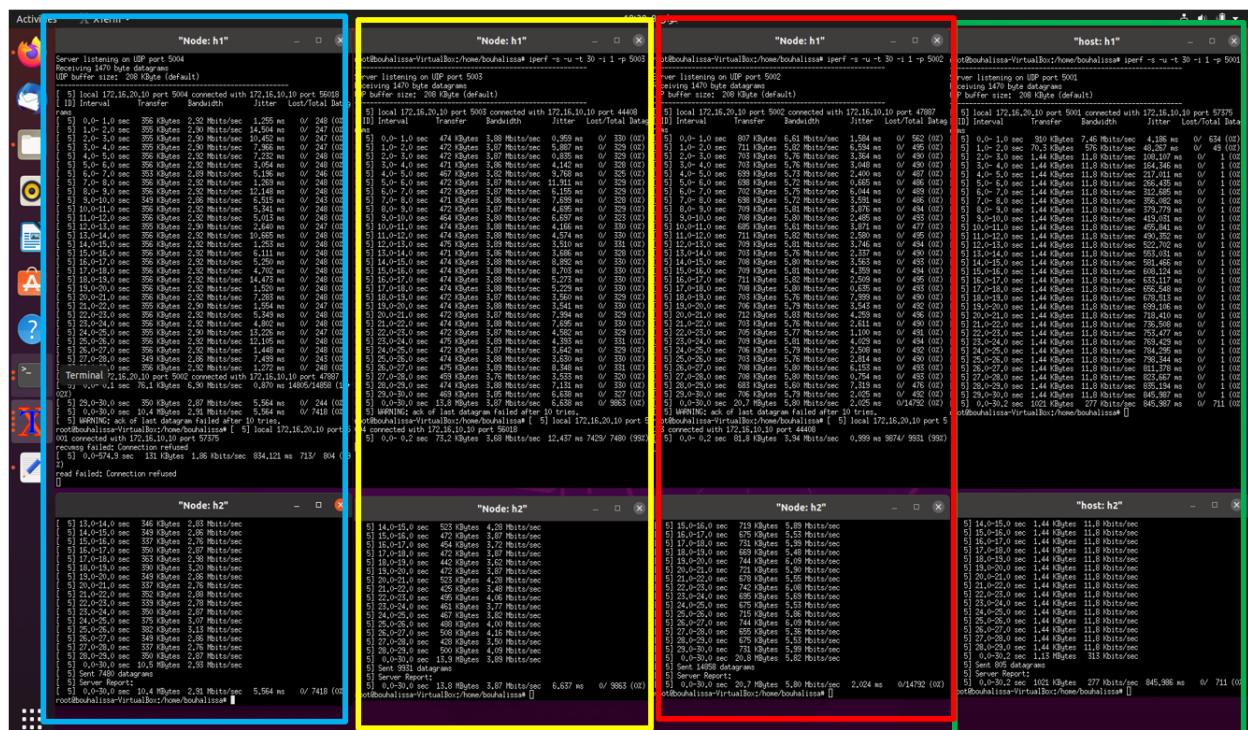


FIGURE 3.17 – Après implémentassions de la QoS

La Figure 3.17 représente la bande passante et jitter en fonction du temps (capturé à chaque seconde pendant 30 secondes), après avoir appliqué la méthode DSCP (DS), en utilisant NBI (REST API).

- Le premier flux (rouge) qui est (flux multimédia streaming) avec une priorité DSCP qui égale à (34).
- Le deuxième flux (jaune) qui est (flux multimédia Conferencing) avec une priorité DSCP qui égale à (26).
- Le troisième flux (bleu) qui est (flux Faible latence/Data) avec une priorité DSCP qui égale à (18).
- Le quatrième flux (Vert) qui est (standard best effort) avec une priorité DSCP qui égale à (0).

Constats (avis) :

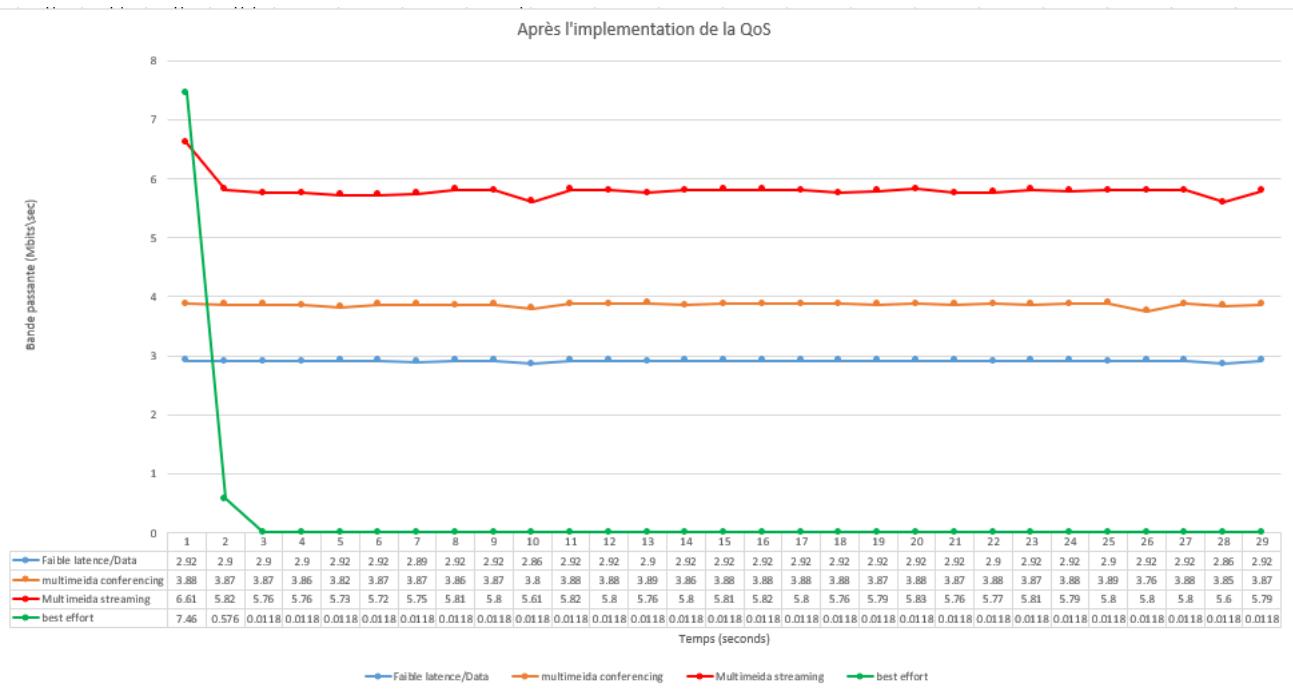


FIGURE 3.18 – Après implémentation de QoS

Comme le montre la figure 3.17 et 3.18 , le test a duré 30 secondes. Le graphe en rouge représente la bande passante ainsi que le jitter entre H1 et H2 pour le flux du multimédia streaming (le trafic le plus favorisé), on a obtenu une moyenne de bande passante de 5.80 Mb/s et des valeurs de jitter rapprochées d'une moyenne de 2.024 ms.

Le graphe en vert représente la bande passante ainsi que le jitter pour le flux standard best effort (le trafic le moins favorisé), on a obtenu une moyenne de bande passante de 277 Kbits/s

et des valeurs de jitter rapprochées d'une moyenne d de 845.986ms.

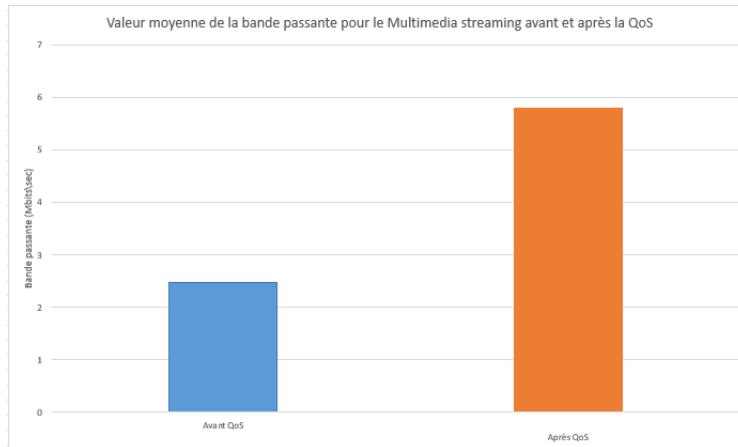


FIGURE 3.19 – Comparaison de la bande passante avec et sans QoS

Comme le montre la figure 3.19, la QoS a bien été respectée pour notre trafic prioritaire qui a surpassé le trafic non prioritaire en termes de bande passante et jitter et qui a respecté les règles imposées dans la file d'attente où le flux de multimédia streaming devait avoir une bande passante minimale de 6 Mbits/s.

3.8 Évaluation du retard pour le multimédia streaming :

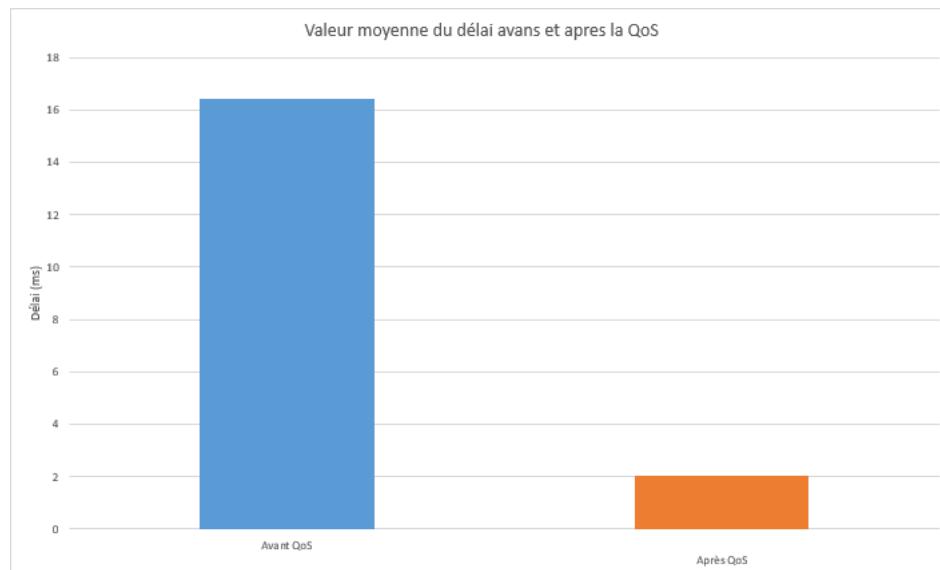


FIGURE 3.20 – Comparaison du délai avec et sans QoS

Les résultats des essais menés par le mécanisme de modèle de file d'attente OpenFlow permettent de prioriser le trafic dans les applications multimédias.

Cela peut être vu à partir des tests exécutant la simulation (Figure 3.20) , qui ont entraîné

une valeur de retard inférieure à celle du réseau sans le mécanisme de modèle de mise en file d'attente dans OpenFlow. La plus petite valeur de délai est due à la priorité accordée au trafic multimédia afin que les paquets entrants soient prioritaires afin qu'ils ne causent pas trop de retard.

3.9 Évaluation des pertes de paquets pour le multimédia streaming :

Les résultats de perte de données avec le mécanisme de modèle de file d'attente sont également meilleurs, comme le montre la figure 3.21.

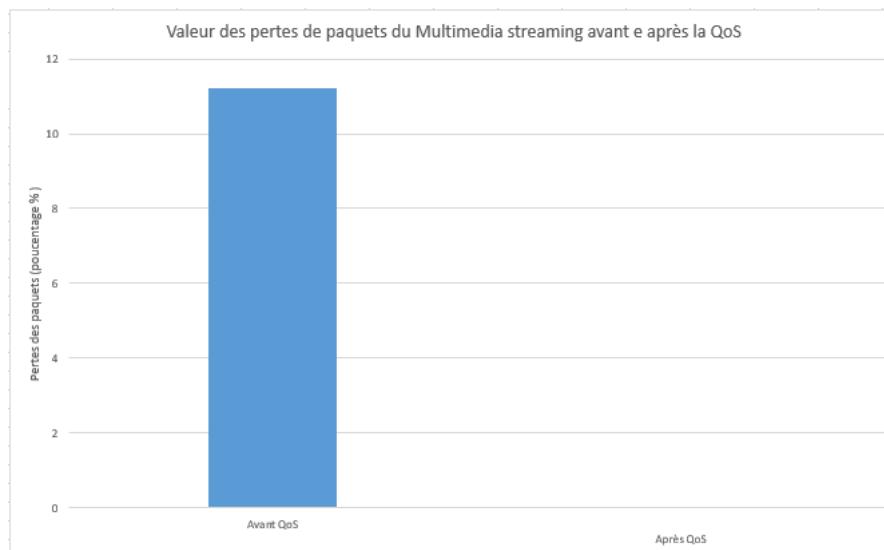


FIGURE 3.21 – Comparaison des pertes de paquets avec et sans QoS

Le trafic sur le réseau multimédia avec le mécanisme de modèle de file d'attente ne subit presque pas de perte de paquets dans le processus de transmission. Les résultats sont meilleurs que sans le mécanisme du modèle de file d'attente.

Les résultats des expériences menées par le mécanisme de mise en file d'attente sur OpenFlow montrent de meilleures performances réseau dans les applications de streaming multimédia à la fois avec des conditions de réseau sans surcharge et avec des conditions de réseau avec un trafic simulé.

3.10 Évaluation de la qualité du multimédia streaming avant et après l'implémentation de la QoS :

A l'aide de l'outil VLC , on a réussi à envoyer un flux multimédia depuis H2 vers H1 avant et après l'implémentation de la QoS. la figure 3.22 représente les résultats obtenus lors de cette manipulation :

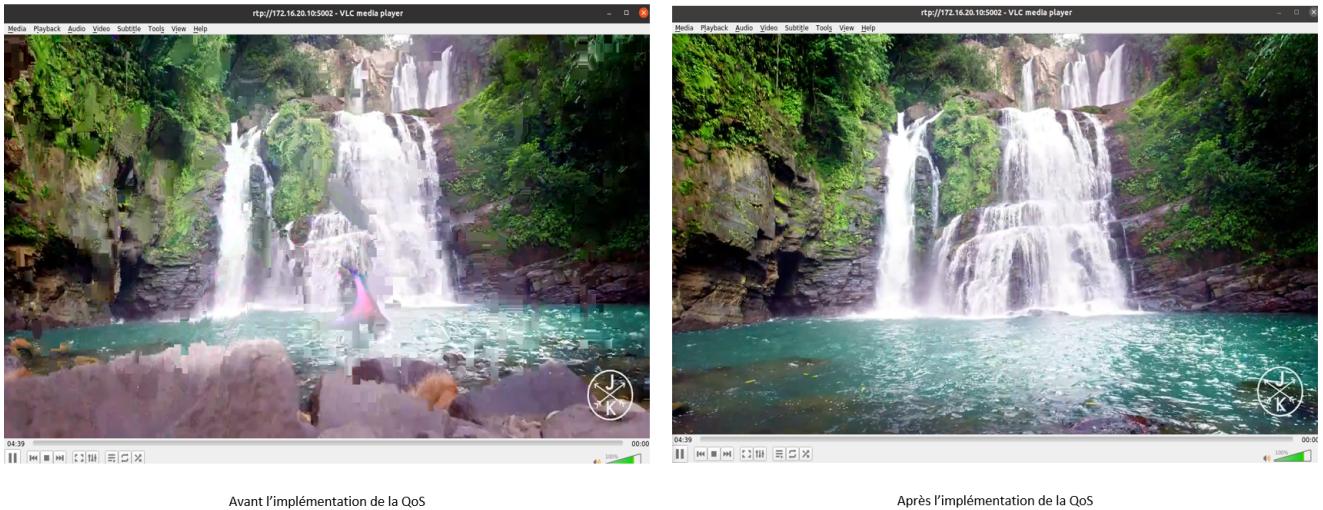


FIGURE 3.22 – L’envoi d’une vidéo avec et sans l’implémentation de la QoS

Comme le montre la figure 3.22 le flux multimédia envoyé sans le mécanisme de modèle de file d’attente (QoS) subit beaucoup de perte de paquet dans le processus de transmission ce qui en résulte la diminution de la qualité de la vidéo.

Cette diminution a été accompagnée par des coupures dans le flux multimédia et ceci est dû au retard de la transmission (delay) causé par la congestion du trafic dans le réseau, tandis que dans le cas où la QoS est implémentée DiffServ a donné la priorité au flux multimédia envoyé par le port 5002 en lui accordant plus de bande passante ce qui a amélioré la qualité de la transmission sans avoir des pertes ou des retards.

3.11 Conclusion

Ce chapitre a été consacré aux détails de l’étude de performances de la QoS du flux multimédia streaming dans le SDN. On a vu à travers les deux scénarios effectués, que l’acheminement des flux sur les chemins les plus courts classiques (sans prise en charge de la QoS), n’est pas toujours la meilleure solution. Le concept SDN ainsi que le protocole OpenFlow nous permettent de faire moins d’efforts, avec plus de flexibilité afin d’effectuer des expérimentations et la QoS avec DiffServ nous permet d’assurer un débit, un délai, une gigue, ainsi qu’un taux de perte de paquets meilleur pour le flux choisi.

Conclusion et Perspectives

Conclusion Générale

La gestion de la qualité de service (QoS) est un concept important dans les réseaux informatiques et les télécommunications car elle est directement liée à l'expérience de l'utilisateur final, en particulier pour les applications en temps réel tel que le streaming vidéo, ToIP, etc. Ces applications génèrent divers flux, chacun nécessitant un traitement différent pour fournir le niveau de service requis par les clients. L'objectif de la mise en œuvre de la QoS est de garantir ce niveau.

La réalisation de ce projet de fin d'étude a été motivée par l'essor rapide du SDN, qui prétend remplacer l'architecture actuelle des réseaux, ainsi que la participation accrue des industries et des recherches dans le domaine de la QoS en SDN. Comme nous l'avons vu tout au long du projet, pour atteindre l'objectif mentionné de priorisation des flux, le SDN doit fournir une meilleure qualité de service. La QoS est le besoin de l'heure, et les chercheurs ont tenté de mettre au point un modèle complet de la QoS.

Dans ce projet, nous proposons la méthode Diffserv pour prendre en charge la gestion de la QoS pour un flux multimédia streaming à l'aide des concepts SDN, cette dernière représentant une nouvelle approche permettant aux opérateurs de réseau de contrôler leur infrastructure via un contrôleur central. Actuellement, le protocole OpenFlow est le plus couramment utilisé pour échanger des informations réseau entre les contrôleurs et les commutateurs pour échanger des informations réseau. Au cours des dernières années, le SDN est passé du domaine de la recherche pure à celui de l'industrie. De nombreux fournisseurs ont publié leurs propres solutions SDN pour le public, ce qui indique que ce concept est entré dans l'étape de mise en œuvre.

On a commencé ce travail en introduisant le réseau SDN et ses caractéristiques sans oublier de le comparer au réseau traditionnel en donnant ces avantages dans le milieu d'entreprise, on a ensuite décrit son architecture et à la fin de ce premier chapitre on a vu ce qu'est un contrôleur SDN Ryu et ses caractéristiques. Dans le chapitre 2 on a commencé par voir quelques généralités

de la QoS en commençant par sa définition ainsi que les différents niveaux de services disponibles comme le IntServ et le DiffServ en citant leurs avantages et inconvénients, ainsi qu'une brève description de quelque application multimédia populaire. Enfin dans le dernier chapitre on a vu les différents outils et les composants nécessaires pour réaliser notre simulation comme mininet et iperf et la méthode Diffserv utiliser pour l'implémentation de la file d'attente ainsi que la QoS dans l'environnement SDN.

Pour finir dans la dernière partie de ce chapitre on a réalisé 2 scénarios pour tester notre simulation et à travers les résultats acquis on a évalué la QoS pour notre flux multimédia qui est le flux prioritaire.

A travers ce projet, nous avons découvert que la spécification OpenFlow Switch est le principal moteur des fonctionnalités disponibles dans les projets SDN, la mise en œuvre de la QoS dans le SDN est plus facile et moins coûteuse que sur les réseaux traditionnels. Il nous a été possible d'implémenter des techniques complexes de QoS comme DiffServ dans un réseau SDN, ce qui est jugé trop complexe à implémenter dans les réseaux actuels. Nous avons testé la QoS dans le SDN, en utilisant le contrôleur Ryu et on a constaté que l'implémentation d'une mesure de QoS est nécessaire pour assurer une bonne gestion des ressources réseaux dans une architecture SDN et pour améliorer la qualité de la communication entre les clients.

Perspectives

Notre travail de thèse est loin d'être terminé. Pour cela, on envisage les perspectives futures suivantes :

- Comparer la méthode DiffServ utilisée à d'autres méthodes.
- Etendre notre architecture afin de prendre en charge d'autres métriques.
- Utiliser les techniques d'apprentissage approfondi (Deep Learning) pour introduire une QoS intelligente.
- Implémenter les solutions QoS dans les réseaux SD-WAN.
- L'expérimentation sur d'autre méthode de QoS comme le Trafic Shapping en mode full SDN.
- Le SDN a une lacune dans le domaine de la sécurité, il serait intéressant d'implémenter une technique de sécurité pour combler ses problèmes (par exemple une interface NBI).

Références

- [1] Seifeddine Ben Chahed. *Mise en oeuvre des aspects de gestion des réseaux définis par logiciels (réseaux SDN)*. PhD thesis, École Polytechnique de Montréal, 2015.
- [2] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. Nox : towards an operating system for networks. *ACM SIGCOMM computer communication review*, 38(3) :105–110, 2008.
- [3] Hani Jamjoom, Dan Williams, and Upendra Sharma. Don’t call them middleboxes, call them middlepipes. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 19–24, 2014.
- [4] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow : enabling innovation in campus networks. *ACM SIGCOMM computer communication review*, 38(2) :69–74, 2008.
- [5] William Stallings. Software-defined networks and openflow. *The internet protocol Journal*, 16(1) :2–14, 2013.
- [6] LL Wanjiun et al. Software defined networks [guest editorial]. In *Communications Magazine*, volume 51, page 113. IEEE, 2013.
- [7] Minlan Yu, Jennifer Rexford, Michael J Freedman, and Jia Wang. Scalable flow-based networking with difane. *ACM SIGCOMM Computer Communication Review*, 40(4) :351–362, 2010.
- [8] Amin Tootoonchian and Yashar Ganjali. Hyperflow : A distributed control plane for openflow. In *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, volume 3, 2010.

- [9] Zheng Cai. *Maestro : Achieving scalability and coordination in centralizaed network control plane*. Rice University, 2012.
- [10] Bhaskar Prasad Rimal, Admela Jukan, Dimitrios Katsaros, and Yves Goeleven. Architectural requirements for cloud computing systems : an enterprise cloud approach. *Journal of Grid Computing*, 9(1) :3–26, 2011.
- [11] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4) :50–58, 2010.
- [12] Sandra Scott-Hayward, Gemma O’Callaghan, and Sakir Sezer. Sdn security : A survey. In *2013 IEEE SDN For Future Networks and Services (SDN4FNS)*, pages 1–7. IEEE, 2013.
- [13] Rowan Klöti, Vasileios Kotronis, and Paul Smith. Openflow : A security analysis. In *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE, 2013.
- [14] 2022 VMware Inc. Why software-defined networking is important ? <https://www.vmware.com/topics/glossary/content/software-defined-networking.html>, 2022.
- [15] Alexander Shalimov, Dmitry Zuikov, Daria Zimarina, Vasily Pashkov, and Ruslan Sme-liansky. Advanced study of sdn/openflow controllers. In *Proceedings of the 9th central & eastern european software engineering conference in russia*, pages 1–6, 2013.
- [16] Michael Jarschel, Simon Oechsner, Daniel Schlosser, Rastin Pries, Sebastian Goll, and Phuoc Tran-Gia. Modeling and performance evaluation of an openflow architecture. In *2011 23rd International Teletraffic Congress (ITC)*, pages 1–7. IEEE, 2011.
- [17] Rob Sherwood and K-K Yap. Cbench controller benchmark, 2011.
- [18] Michael Jarschel, Frank Lehrieder, Zsolt Magyari, and Rastin Pries. A flexible openflow-controller benchmark. In *2012 European Workshop on Software Defined Networking*, pages 48–53. IEEE, 2012.
- [19] David Tipper. Resilient network design : challenges and future directions. *Telecommunication Systems*, 56(1) :5–16, 2014.
- [20] Marie Hélène Delmond. Le sdn.
- [21] Yvon Zafimahefa Andrianirina. *Développement d’un réseau défini par logiciel (SDN) programmable, transparent et ouvert*. PhD thesis, Université du Québec en Outaouais, 2021.
- [22] Ihssane Choukri, Mohammed Ouzzif, and Khalid Bouragba. Software defined networking (sdn) : Etat de l’art. In *Colloque sur les Objets et systèmes Connectés*, 2019.

- [23] Open Networking Foundation. Sdn technical specifications, 2015. Accessed : 2022-04-28.
- [24] Manar Jammal, Taranpreet Singh, Abdallah Shami, Rasool Asal, and Yiming Li. Software defined networking : State of the art and research challenges. *Computer Networks*, 72 :74–98, 2014.
- [25] Fouad Benamrane, Redouane Benaini, et al. Performances of openflow-based software-defined networks : an overview. *Journal of Networks*, 10(6) :329, 2015.
- [26] Pica8 Inc. Picos overview, 2014. Accessed : 2022-05-12.
- [27] Adnan Akhunzada, Abdullah Gani, Nor Badrul Anuar, Ahmed Abdelaziz, Muhammad Khurram Khan, Amir Hayat, and Samee U Khan. Secure and dependable software defined networks. *Journal of Network and Computer Applications*, 61 :199–221, 2016.
- [28] Eric Keller and Jennifer Rexford. The " platform as a service" model for networking. *INM/WREN*, 10 :95–108, 2010.
- [29] Ryu SDN Framework Community. Ryu sdn framework. Accessed : 2022-04-14.
- [30] Markus Brandt, Rahamatullah Khondoker, Ronald Marx, and Kpatcha Bayarou. Security analysis of software defined networking protocols-openflow, of-config and ovsdb. In *The 2014 IEEE fifth international conference on communications and electronics (ICCE 2014), DA NANG, Vietnam*, 2014.
- [31] Robert Braden, David Clark, and Scott Shenker. Rfc1633 : Integrated services in the internet architecture : an overview. 1994.
- [32] Wendell Odom. *CCNA Routing and Switching ICND2 200-101 Official Cert Guide*. Cisco Press, 2013.
- [33] Jozef Babiarz, Kwok Chan, and Fred Baker. Configuration guidelines for diffserv service classes. Technical report, 2006.
- [34] Juha Heinanen, Fred Baker, Walter Weiss, and John Wroclawski. Assured forwarding phb group. Technical report, 1999.
- [35] Reza Mohammadi and Reza Javidan. An adaptive type-2 fuzzy traffic engineering method for video surveillance systems over software defined networks. *Multimedia Tools and Applications*, 76(22) :23627–23642, 2017.
- [36] Zuo Xiang and Patrick Seeling. Mininet : an instant virtual network on your computer. In *Computing in Communication Networks*, pages 219–230. Elsevier, 2020.
- [37] Mathijs Mortimer. iperf3 documentation, 2018.
- [38] VideoLan. Vlc media player, 2006.

Appendice A

Installation de Ryu

Etape 1 : Installer n'importe quelle hyperviseur type-2 (VMware ou virtual box).

Etape 2 : Créer une machine virtuelle avec Ubuntu comme système d'exploitation sous le nom de Ryu.

Etape 3 : On commence d'abord par installer pip et git pour faciliter l'installation de python et ryu par la suite.

L'installation se fait à travers les commandes suivantes :

- \$ sudo apt-get install git
- \$ sudo apt-get install python-pip
- \$ Sudo apt-get update
- \$ sudo pip install --upgrade pip
- \$ pip install -r tools/pip-requires
- \$ sudo python setup.py install

Etape 4 : Ensuite grâce à la commande pip installer précédemment, on pourra installer le Controller ryu à travers la commande :

- \$ pip install ryu
- \$ git clone git ://github.com/osrg/ryu.git

Etape 5 : Pour finir en entrant dans le contrôleur Ryu installer grâce à la commande :

\$ cd ryu On installe les outils nécessaires pour son bon fonctionnement grâce aux commandes suivantes :

- \$ sudo pip install -r tools/pip-requires
- \$ sudo python setup.py install

Appendice B

Installation de Mininet

Etape 1 : utiliser la commande :

```
$ git clone https://github.com/mininet/mininet pour cloner le fichier mininet  
dans notre machine virtuelle Ubuntu.
```

Etape 2 : pour installer mininet on utilise la commande :

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

Option : installation complete (Mininet +Ovs) (using your home directory) :
install.sh -a

Etape 3 : Si Mininet se plaint du fait qu'Open vSwitch ne fonctionne pas, assurez-vous qu'il est installé et en cours d'exécution grâce aux commandes suivantes :

- \$ sudo apt-get install openvswitch-switch
- \$ sudo service openvswitch-switch start

Etape 4 : l'installation de python se fait grâce à la commande :

```
$ sudo PYTHON=python3 mininet/util/install.sh -n install Python 3 Mininet
```