Conception d’un Réseau IoT Basé sur SDN

# Introduction

An de mettre en oeuvre notre conception , nous avons dédié ce chapitre aux details de l’évaluation des performances d’un réseau IoT basé sur SDN. Nous commençons par présenter les outils qu’on va utiliser pour atteindre l’objectif de ce project, puis nous présentons la topo-

logie de reseau et le processus de sa création. Nous terminons ce chapitre par une analyse des résultats en se basant sur diérentes Paramètres de QoS comme le débit, la gigue, le délai et la perte des paquets.

# Le Controleur ONOS :

* + 1. Aperçu sur ONOS :

ONOS (open network operating system) représente le plan de contrôle dans le SDN, assure la gestion des équipements réseau (ex : switches) et les liens qui les relie, et aussi l’exécution des programmes ou des modules pour fournir les services de communication entre les nİuds et les réseaux adjacentes. ONOS ore plusieurs fonctionnalités : des APIs et abstraction, ligne de commande CLI et une interface graphique GUI, allocation des ressources et permissions.

ONOS peut être exécuté comme un system de distribution sur plusieurs serveurs, ce qui permet d’utiliser les ressources de CPU et de la mémoire des serveurs et permet aussi la tolérance contre la panne de serveur.

Le kernel d’ONOS et les services core, et les application sont programme en JAVA sous forme

CHAPITRE 3

;A<

Le Controleur ONOS :

de bundles qui sont chargés dans le conteneur Karaf OSGi. OSGi est un système de composants pour Java qui permet d’installer et d’exécuter des modules de manière dynamique dans une seule JVM (JAVA Virtual Machine) [9].

* + 1. L’Architecture d’ONOS

L’architecture d’ONOS est composée de 4 éléments principaux [10] :

Le Noyau Distribué

Fournit l’évolutivité, la disponibilite, les performances, onos est deploye sous forme d’un service sur un group de serveur. Et le meme software execute dans chaque serveur. Ce qui permet d’assurer la disponibilite en cas d’une panne d’un des serveurs. Les instance d’ONOS fonctionnent ensemble pour créer ce qui apparaît au reste du réseau comme une plate-forme unique.

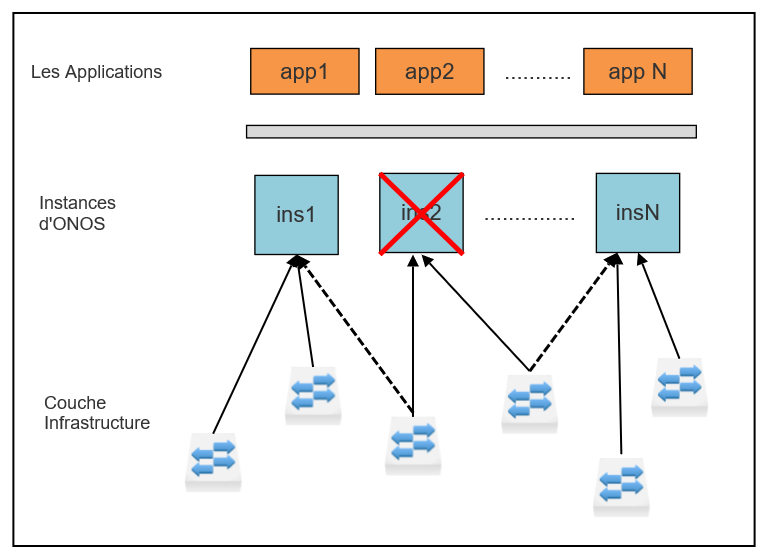


Figure 3.1 –Le Noyau Distribué.

Abstraction Northbound/API

Il existe deux abstractions northbound : le Framework d’intention et la vue de réseau :

Le Framework d’intentionpermet à une application de demander un service du réseau sans avoir de connaitre les détails de service. Cela permet aux operateurs réseau et les développeurs

d’applications de programmer le réseau a un niveau élevé.

La vue globale du réseaufournit une vue des hosts, switches,liens.une.

Les Abstractions Southbound

Active les protocoles de niveau bas pour Contrôler les équipements OpneFlow. L’abstraction southbound isole le contrôleur d’ONOS des détails des couches inferieurs.

La Modularité des Softwares

La construction des logiciels est importante, le logiciel doit être facile à améliorer, modier et maintenir. L’équipe ONOS a mis grandsoin dans la modularité pour faciliter les développeurs à travailler avec le logiciel.

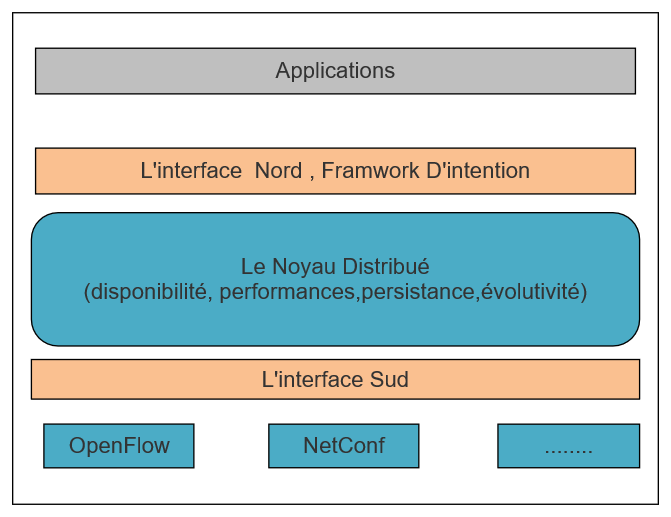


Figure 3.2 –Les Couches d’ONOS.

# Mininet-IoT

* + 1. Aperçu sur Mininet

Mininet est un émulateur réseau qui permet de créer des topologies réseau vitruele en utilisant les diérents neuds (switches, controleur, hosts) et des liens qui les relies, les switches de mininet supportent OpenFlow pour performer le routage dans le reseau SDN.

Après l’installation de mininet nous pouvons tester une topologie de 2 hosts, 1 switch et 1

contôrleur en exécutant "> sudo mn" [11].

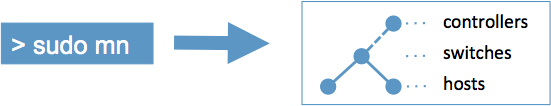


Figure 3.3 –Mininet.

[11]

* + 1. Mininet-IoT

Minininet-IoT est une extention du codeBase de mininet qui fournir les outils neccessaires pour créer et développer des réseau IoT. Mininet permet de :

* + - * Fournir une expérience d’essai simple pour développer des applications OpenFlow.
      * Inclure une linge de commande pour consulter et debugger les diérents noeuds de réseau.
      * Prêt pour l’utiliser sans programmation, mais dans le cas invers vous pouvez aller plus loin avecPYTHON API.
      * Tester des topologies complexes en utilisant l’environnement vituelle.

Sur mininet chaque host a un terminal linux, donc vous pouvez exécuter des commandes linux sur les diérents hosts.

Comme il est mentionné au-dessus, Il est possible d’utiliser Mininet directement via l’outil en

ligne de commande pour créer les diérents topologies SDN. Exemple : la commande suivante permet de créer un switch connecté à 3 hosts(PC) et à un contrôleur bien dénie.

> sudo mn - -topo single,3 - -controller remote,ip=172.0.0.1 :6653

Ou bien on peut utiliser lePYTHON API, pour programer notre propre réseau.

* + 1. Python API

Dans cette imlpémentation on va utiliser PYTHON pour créer notre propre topologies avec la librairie fournit par python API. Mininet prend en charge les topologies paramétrées. Avec quelques lignes de code Python, vous pouvez créer une topologieexible qui peut être congurée en fonction des paramètres que vous lui transmettez et réutilisez pour plusieurs expériences.

Remarque: mininet IoT nécessite un kernel 4.18 ou plus.

# Iperf

Iperf est un outil pour mesurer la bande passante et la qualité d’un lien réseau. Le lien dernier est délimité par deux machines sur lesquelles est installé Iperf. Les performances d’un lien peuvent être évaluées comme suit [12] :

* Le Délai : peut être mesurée par la commande de Ping.
* La Gigue (variation de la délai) : par un test Iperf UDP.
* Perte de paquet : mesurée avec un test Iperf UDP.
* Le Débit et Bande passante : peut être mesuré par iperf TCP ou UDP après avoir spécié la bande passante.
* tester des topologies complexes en utilisant l’environnement vituelle.

# La Réalisation de la Topologie

* + 1. Aperçu

Dans cette topologie illustrée dans lagure 3.4 on a 4 groupes chaque groupe contient des capteurs qui forment un réseau maillé et s’échangent des données via le protocole 6LOWPAN, tels que le capture qui est connecté avec le switch via Ethernet (lienlaire) représente le capteur Edge qui permet de relier réseau des capteurs avec le réseaulaire. dans ce test il n’y aura pas de traitement des données (data processing). Mais généralement le capture Edge qui reçoit les données faire le traitement avant l’envoi au serveur, ou on peut utiliser un serveur de computing qui va faire le traitement et le sauvegarde des données.

Et avec le l’application IPv6 réactive forwarding déjà installé sur ONOS les paquets IPv6 des captures destinés au serveur seront routé.

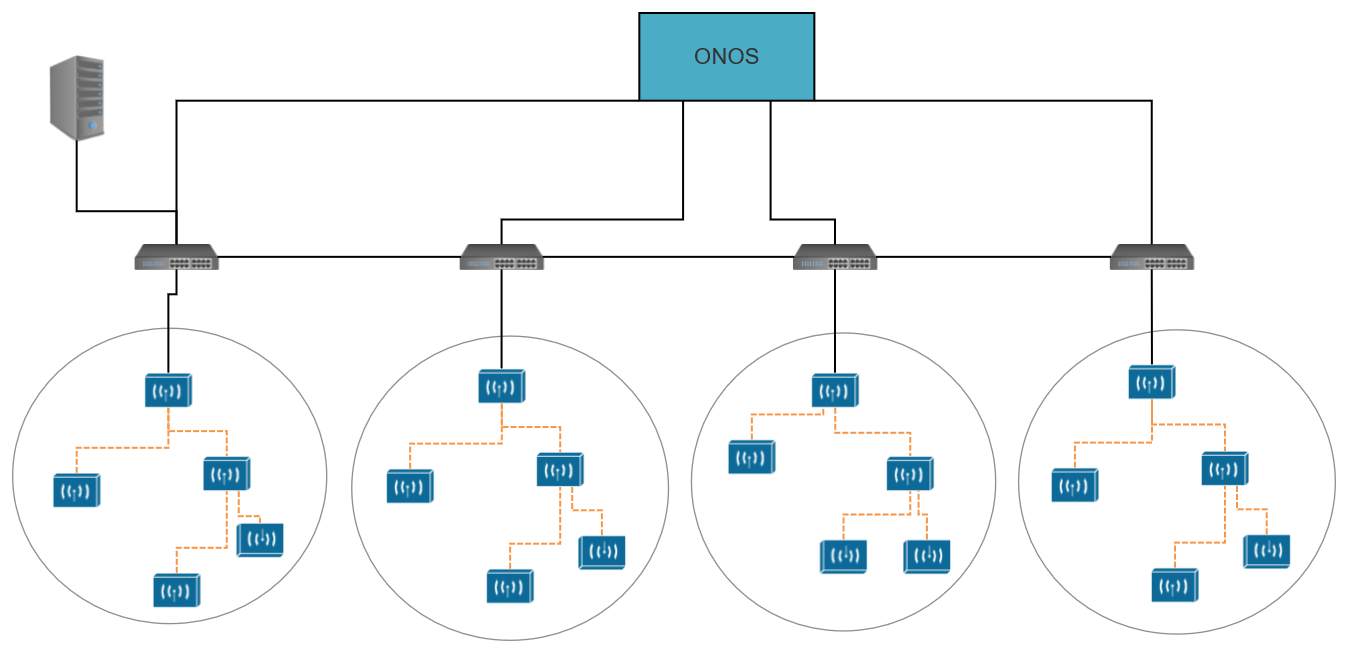


Figure 3.4 –La Topologie Proposée.

* + 1. Réalisation

On démarre le contrôleur et on entre dans linge de commande d’ONOS (ONOS CLI), puis on active l’application de réactive forwarding. par la command “app activate fwd”.

Après , il faut activer les fonctionnalité de IPV6 pour le routage des paquets IPV6. Pour cela on exécute les 2 command suivantes :

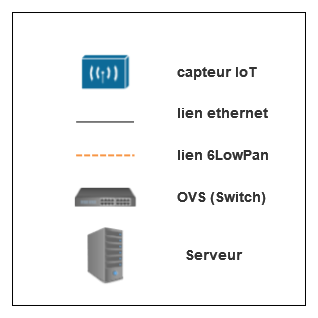


Figure 3.5 –Description.

"cfg set org.onosproject.fwd.ReactiveForwarding ipv6Forwarding true" "cfg set org.onosproject.fwd.ReactiveForwarding matchIpv6Adress true"

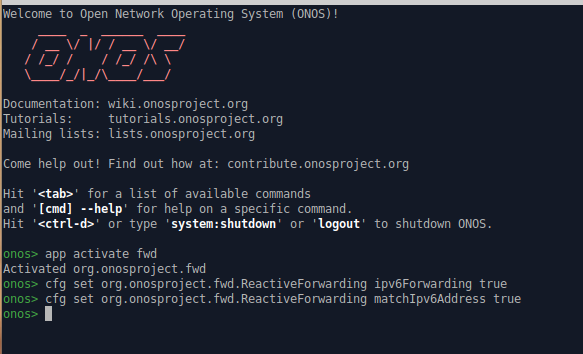


Figure 3.6 –La ligne de command d’ONOS.

Après avoir préparé le contrôleur, on exécute le script Python qui permet de créer la topologie sous de mininet-iot. Dans lan d’exécution nous consultons l’interface graphique d’ONOS comme il est montré dans lagure 3.7, et voyons les switches OVS (Open V Switch) connecté au contrôleur les capteurs Edge et au serveur, Le deuxième l’hôte connecté au switch 1 est le serveur. On ne voit pas les autre capteur paracerque il ne pas connecte directement au switch.

Pour l’évaluation des performances, on a pris 4 scénarios de test, celui qui est dans tolopogie de lagure 3.4, les 3 qui restent sont présentés dans lagure 3.9.

Notez bien qu’ nous avons besoin de faire un ping entre les capteurs et le serveur pour qu’ ils

soient détecté par le contrôleur, comme il est montré dans lagure 3.8.

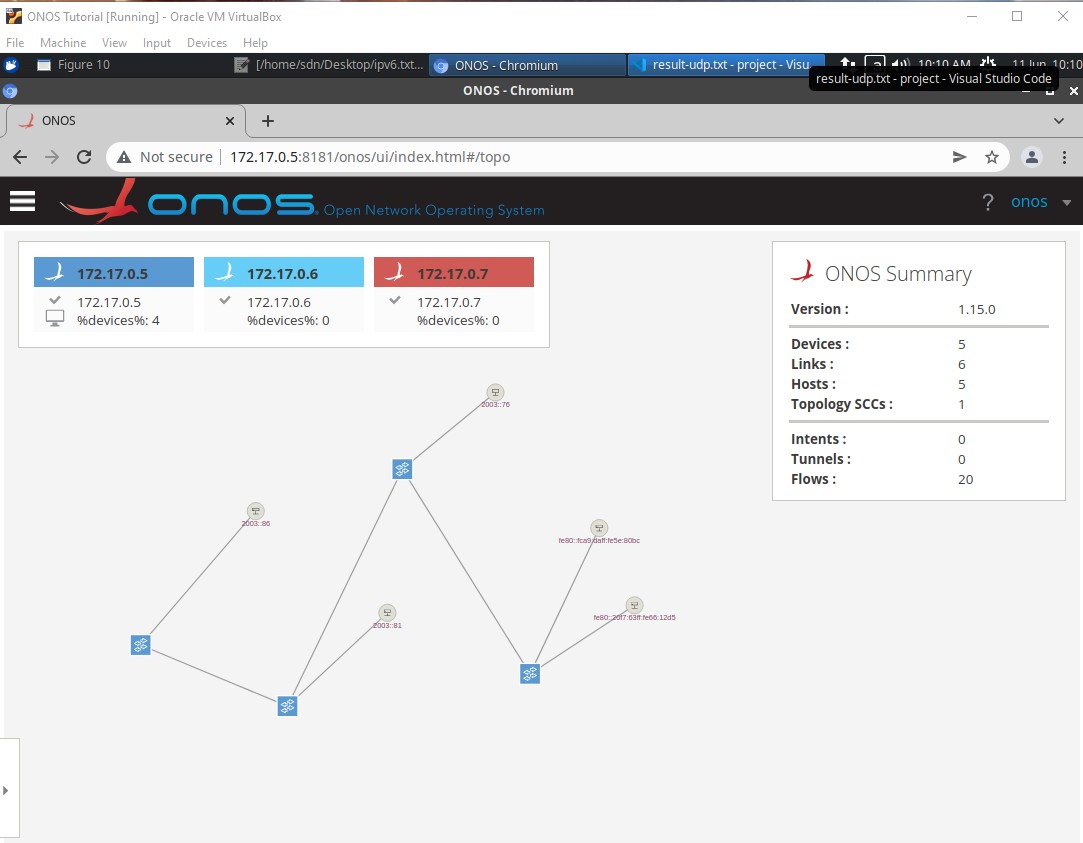


Figure 3.7 –l’Interface Graphique d’ONOS.

# La Qualité de Service

La QoS (Qualité de Service) représente la capacité d’un réseau à gérer la transmission de données de diérentsux avec les spécications requises des paramètres QoS. On peut évaluer la qualité de service avec ces paramètres :

—La Perte de Paquets: représente le pourcentage des paquets qui n’arrive pas à la destination à cause de congestions de réseau qui mène à la suppression des paquets. La perte doit être le plus minimal possible.

—Le Délai: est le temps nécessaire pour qu’un message arrive à la destination. Plus il est

proche à zéro plus la QOS est meilleur.

—La Gigue: représente la variation de délai, elle est causé par la congestion de réseau,il doit s’approcher à zéro .

—La Bande Passante: est la quantité d’information maximale que le lien peut supporter dans une période de temps.

Le tableau 3.1 montre les normes Tiphon utilisées dans l’analyse processus. Le débit mesuré

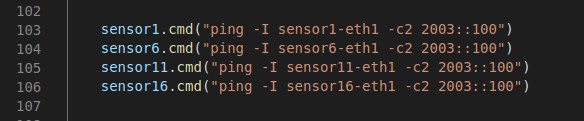


Figure 3.8 –Ping.

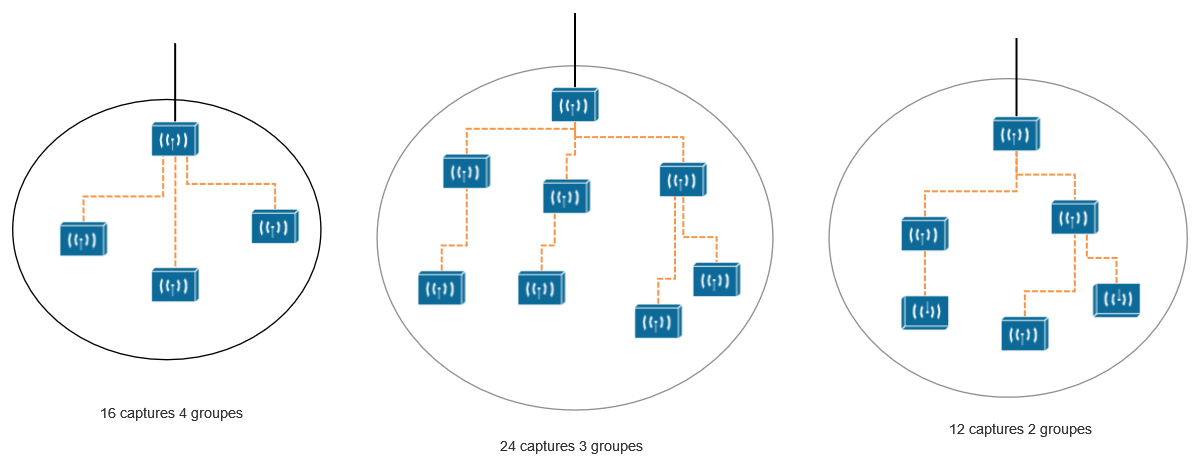


Figure 3.9 –Les Diirents Scénarios Testés.

est basé sur le nombre total d’arrivées de paquets réussies observé à destination pendant un intervalle de temps spécique divisé par la durée de cet intervalle de temp [13].

# Evaluation des performances

comme le montre lagure 3.10, pour que les capteurs qui reçoivent des données, nous devons les dénir comme serveurs dans iperf en utilisant l’option -s. On utilise le protocole UDP pour ces tests, -u signie UDP , -V est pour IPV6 et -p est pour spécier le numéro de port, -t est juste pour le temps, ce qui signie que le test ne doit pas dépasser 60 secondes.

pour les capteurs qui envoient, nous devons les dénir en tant que clients dans iperf, nous faisons ça par spécier l’adresse IP du destinataire dans l’option -c comme indiqué sur lagure 3.10, la bande passante peut être spéciée avec -b, comme vous pouvez le voir qe’elle est déni sur 120Kbits/s, et les résultats seront de chaque connexion seront sauvegarder dans lechier

Table 3.1 –Mesure de la QoS [13].

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Remarque | Index | Délai (ms) | Gigue (ms) | Perte (%) | Débit (%) |
| Trés Bien | 4 | 150 < | 0 | 0 | 100 |
| Bien | 3 | 150-300 | 0-75 | 3-15 | 75 |
| Moyen | 2 | 300-450 | 75-125 | 15-25 | 50 |
| Mauvais | 1 | > 450 | > 125 | 25-100 | < 25 |

result-udp.txt.

nous pouvons voir les résultats de débit,de la gigue et de la perte des paquets dans lagure 3.11, le délai peut être calculer par la command de ping. on refaire le test plusieurs fois et on prend la moyenne. le scénario de test utilisé est RRUL (Realtime Response Under Load). Ce type de test simule une réseau dans les pires conditions en mettant une charge sur le réseau puis faire la récupération des données. on refaire la meme processus avec d’autre scenario de test dans lagures 3.9, les résultats sont présentés dans lesgures 3.12, 3.13,??, 3.15.

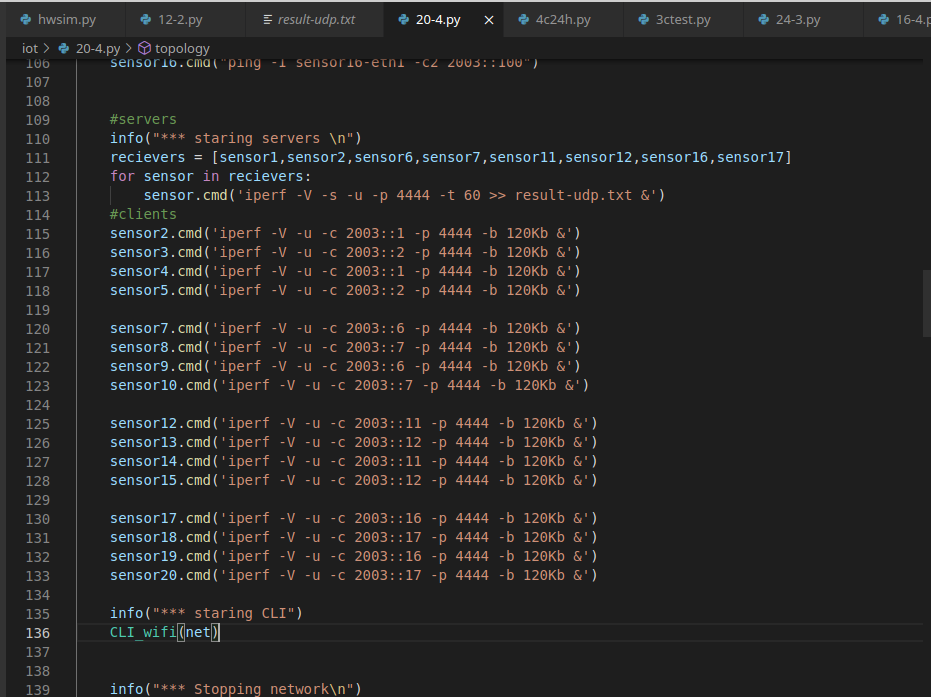


Figure 3.10 –Tests d’Iperf.

Le Débit

lagure 3.12 montre les les résultats de débit, Pour les 4 tests on a attribué 120KB/s de bande passante et nous avons obtenu un débit de 81Kbit/s (68%) et 63Kbit/s (50 %) respectivement. Et pour les 2 autre 65Kbit/s de bande passante et nous avons obtenu un débit de 45Kbits/s (30 %) et 46Kbits/s (36%), alors le débit tombe dans les catégories de moyen et bien.

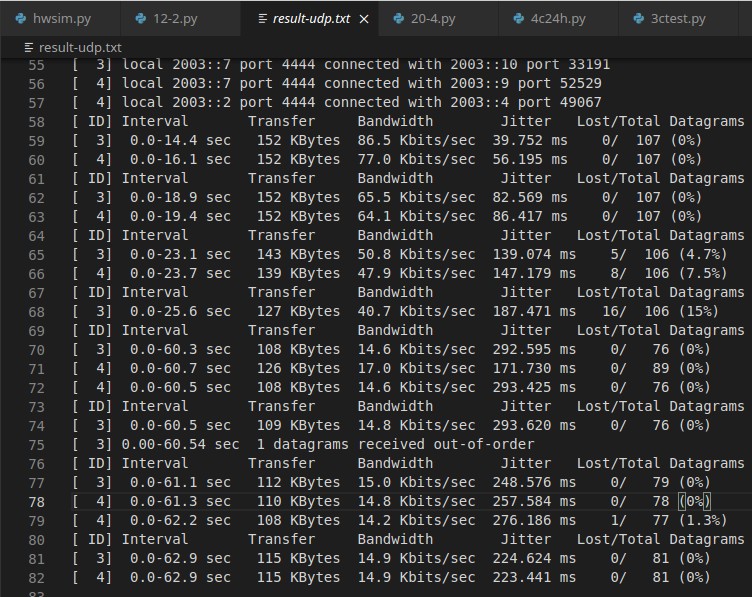


Figure 3.11 –Les Résultats de Test.

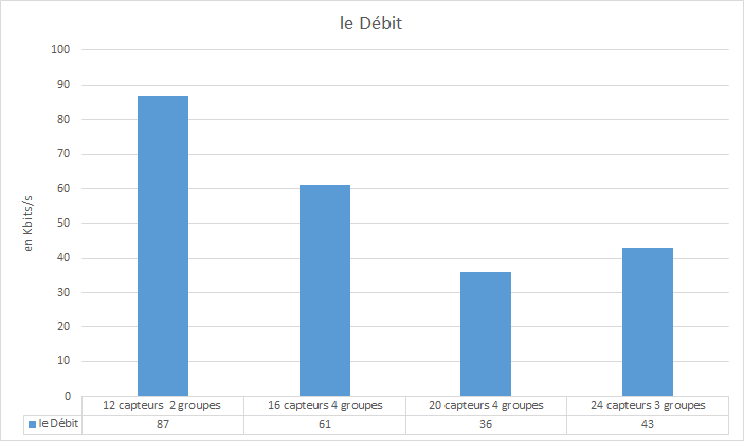


Figure 3.12 –Le Débit.

* + 1. La Gigue

lagure 3.13 montre les les résultats de la gigue, nous remarquons une relation de corrélation directe entre le nombre de captures et la gigue, Nous avons obtenu 57ms, 63ms, 185ms et 146ms pour les 4 scénarios. Des performances bonnes et moyennes pour 12 capteurs 2 groupes et 16 capteurs 4 groupes respectivement. De mauvaises performances pour les 24 capteurs 3 groupes

et 20 capteurs 4 groupes. Les tests de gigue tombent dans les catégories de bonne et moyenne.

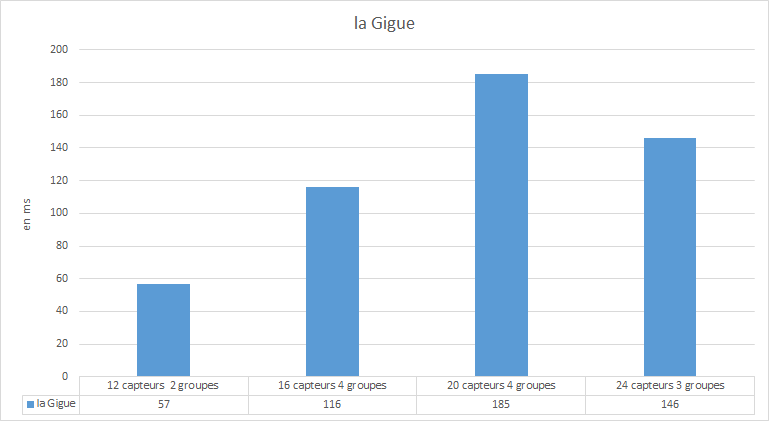


Figure 3.13 –La Gigue.

La Perte

dans lagure 3.14 nous remarquons que la perte augmente en fonction de nombre des capteurs, 0% pour le test de 12 capteurs ce qui est très bien, Et 2%, 2.2% et 4.5% pour les autres ce qui est bien.

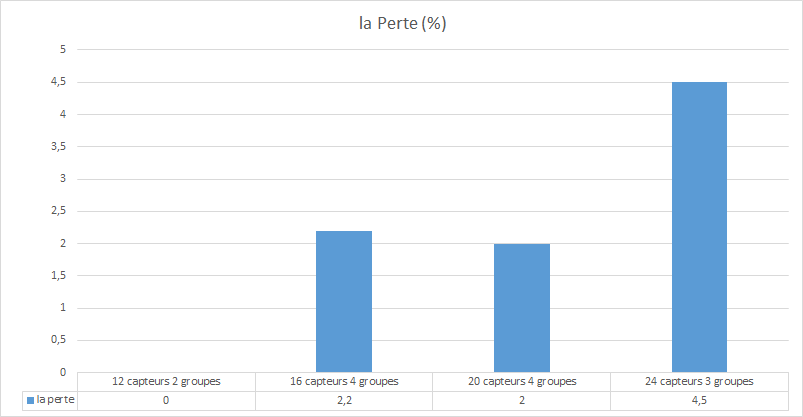


Figure 3.14 –La Perte.

Le délai

lagure 3.15 montre les les résultats de la délai, nous remarquons Encore une relation de corrélation directe entre le nombre de capture et le délai, On a obtenu 11.5ms, 15ms, 19ms et 20ms pour les 4 scénario. Les tests de gigue tombent dans la catégories de bien.

CHAPITRE 3

;A<

Conclusion

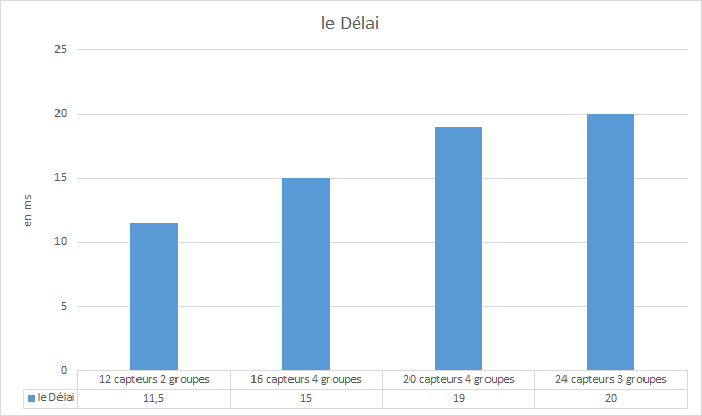


Figure 3.15 –Le Délai

Performances d’Ethernet (laire)

Les performances en réseaulaire sont très bien et similaire dans tous les scénarios.

—Le Débit :1 Mbit/s.

—La Gigue :0 - 0,04 ms.

—La Perte :0 %.

—Le Délai :0 - 35 ms.

# Conclusion

Sur la base des résultats des valeurs de débit, délai, gigue et de perte de paquets on peut dire que les performances de réseau 6Lowpan sont moyen-bien avec les meilleur résultats dans le cas de 2 groupes 12 captures. nous avons déduit que Plus le nombre de capture est élevé plus les performances baissent, les 2 derniers tests 3 groupes, 24 captures et 4 groupes, 20 captures ont donné de mauvaises performances de la gigue, Ce qui n’est pas bien pour les applications en temps réel comme la VoIP (Voice over Internet Protocol) par exemple en cas d’un grand réseau qui contient beaucoup de capteurs.