

## CHAPITRE 4

### RESULTATS ET ANALYSE

#### 1 Introduction

OMNeT++ IDE de simulation étend les fonctionnalités de la plate-forme Eclipse. Elle permet aux utilisateurs de créer et de configurer des modèles, effectuer des exécutions de traitement et d'analyser les résultats de la simulation. Elle est basée sur des composants, modulaire et à architecture ouverte, donc très ouvert pour les extensions. Popularité de OMNeT++ dans le milieu universitaire est en augmentation en raison de ses modèles et de cadres open-source et la documentation en ligne. Nous avons utilisé OMNeT++ version 4.2 pour nos simulations.

#### 2 INETMANET Framework

INETMANET Framework contient des protocoles et des composants supplémentaires qui sont extrêmement utiles pour la simulation de réseaux de communication sans fil sur OMNeT++. Il propose différents modèles de propagation, les protocoles de couche de liaison, les protocoles de routage mobiles, des modèles de mobilité, des modèles d'application et permet la poursuite du développement à travers le Github. Il est possible de mettre en œuvre des réseaux utilisant des mécanismes de routage MPLS, AODV, OLSR et BATMAN ou sur OMNeT++ grâce à INETMANET Framework.

#### 3 Les étapes d'une simulation avec OMNeT++:

1. Un modèle OMNeT++ est construit à partir de composants (modules) qui communiquent en échangeant des messages. Les modules peuvent être imbriqués, c'est plusieurs modules peuvent être regroupés pour former un module composé. Lors de la création du modèle, vous devez mapper votre système dans une hiérarchie de modules communicants.

2. Définir la structure du modèle dans la langue NED. Vous pouvez modifier NED dans un éditeur de texte ou dans l'éditeur graphique de l'OMNeT++ basé sur Eclipse IDE de simulation.

3. Les composants actifs du modèle (modules simples) doivent être programmés en langage C++, en utilisant le noyau de simulation et les bibliothèques de classes.

4. Fournir un `omnetpp.ini` apte à détenir OMNeT++ configuration et les paramètres de votre modèle. Un fichier de configuration peut décrire plusieurs runs de simulation avec des paramètres différents.

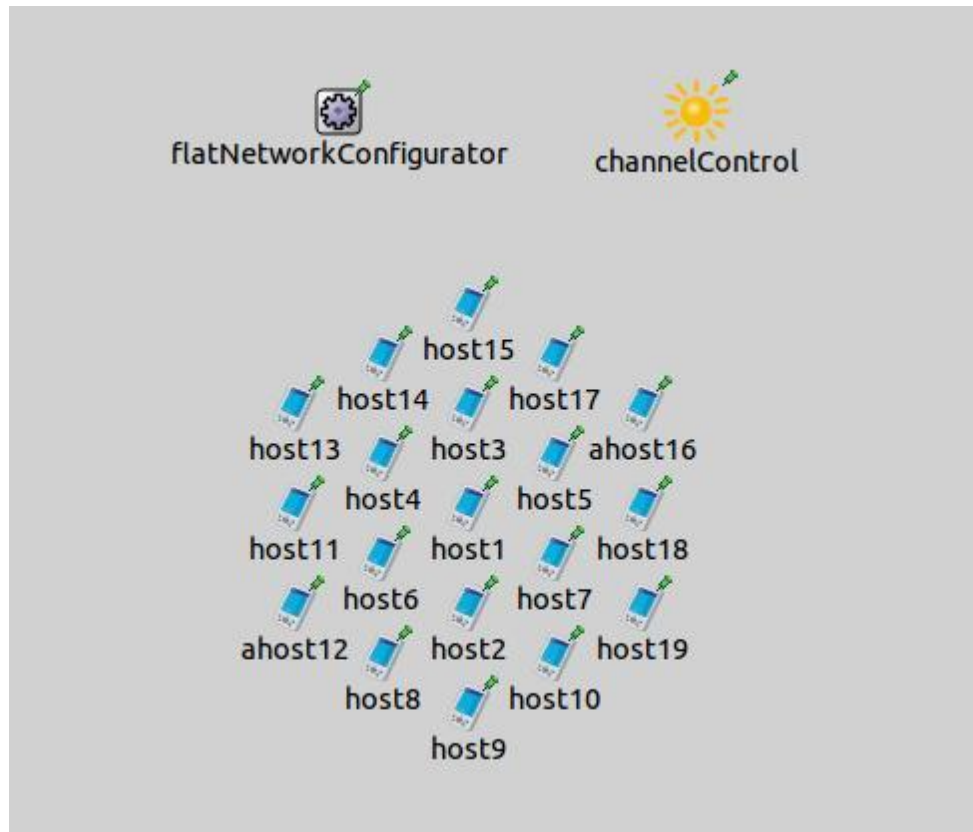
5. Construire le programme de simulation et l'exécuter. Le code sera lié avec le noyau de simulation OMNeT++ et l'une des interfaces utilisateur qu'offre OMNeT++. Il existe une ligne de commande (batch) et des interfaces utilisateurs interactives et graphiques.

6. Les résultats de simulation sont écrits dans le vecteur de sortie et les fichiers scalaires de sortie (output vector and output scalar files). Nous pouvons utiliser l'outil d'analyse dans l'IDE de simulation pour les visualiser. Les fichiers résultats sont à base de texte, de sorte que vous pouvez également traiter avec R, Matlab ou d'autres outils.

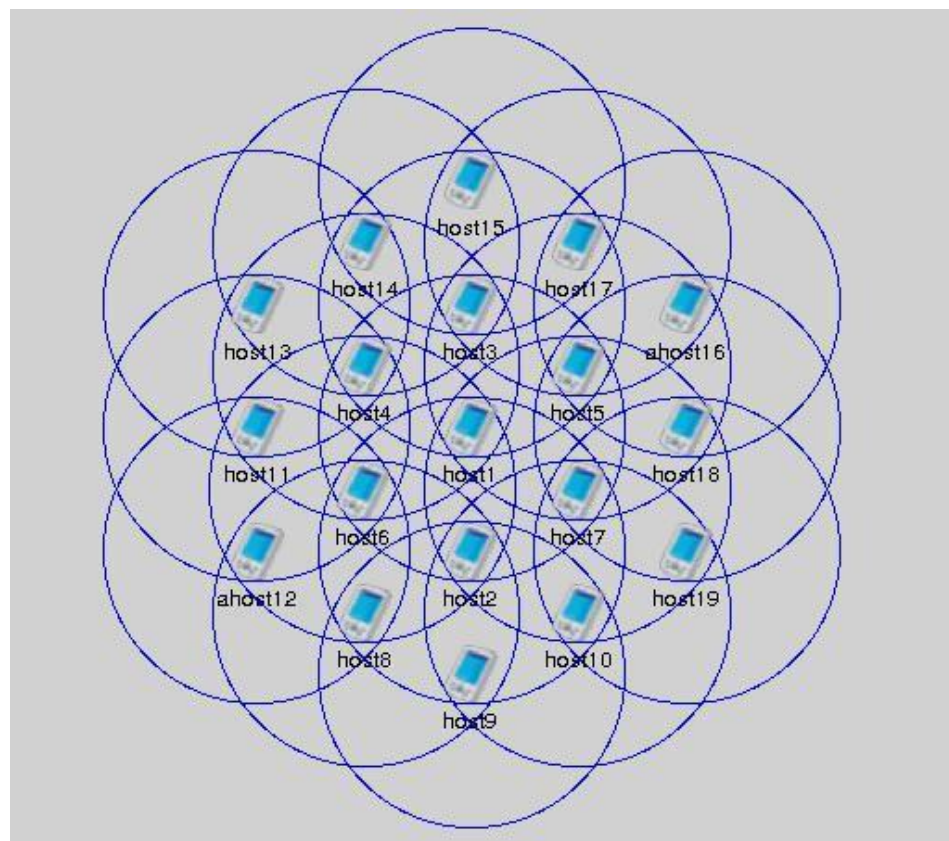
## **4 Configuration de simulation**

### **4.1 Architecture pour les simulations**

L'architecture de simulation est conçue pour observer les caractéristiques d'un WMN sous une topologie cellulaire telle que représentée sur la figure suivante. Les nœuds sont positionnés à des distances égales les uns des autres. Cette approche maintient une base plus facile d'observer les caractéristiques de routage. Il y a au total 19 nœuds formant un WMN client peer-to-peer. Chaque nœud dispose de 6 nœuds voisins dans sa zone de couverture et ne peut transmettre ou recevoir de ses voisins. Les cercles bleus centrés sur les nœuds marquent leurs zones de couverture.



**Figure 4.1** L'Architecture de simulation par défaut



**Figure 4.2** les zones de couverture des nœuds

## 4.2 Initialisation des valeurs de simulation

Les valeurs utilisées pour la configuration initiale des simulations sont présentées dans le tableau ci-dessous.

<b>MAC maximum Queue Size</b>	15
<b>MAC and Radio bitrate</b>	54 Mbps
<b>Radio Sensitivity</b>	-79 dBm
<b>Channel Carrier Frequency</b>	2.4 GHz
<b>Maximum Sending Power</b>	-110 dBm
<b>Signal Attenuation Threshold</b>	-110 dBm
<b>UDP Application Type</b>	UDP Basic Burst
<b>UDP Application Message Length</b>	1024B
<b>UDP Application Message Frequency</b>	uniform(0.1s,0.3s)
<b>Message Frequency Jitter</b>	uniform(-0.001s,0.001s)
<b>Radio Transmitter Power</b>	0.5mW
<b>Radio Thermal Noise</b>	-110dBm
<b>Radio Sensitivity</b>	-85dBm
<b>Radio Operation Mode</b>	802.11g
<b>Signal-to-Noise Ratio Threshold</b>	4dB
<b>Simulation Duration</b>	300 et 600s

**Table 4.1:** les valeurs d'initialisation de simulation

## 5 Phase de simulation

### 5.1 Le comportement des trois mécanismes de routage étudiés :

Le comportement d'un protocole c'est son attitude à gérer suivant sa vision d'ensemble et détaillée du réseau et ses constituants. On s'intéresse à comprendre la manière que chacun des trois protocoles utilise pour explorer le réseau et définir une vue logique qui lie ses constituants, en fait cette vue n'est autre que la topologie qu'a défini ce protocole pour le réseau.

L'exploration du réseau se fait par des messages éclaireurs, dont leur rôle est de fournir les informations pertinentes qui permettent de constituer un modèle topologique du réseau en question. Ce mécanisme de messages de contrôle est lui-même qui est au cœur de la différence existante entre les trois protocoles.

Et de ce fait, la présentation de ces trois mécanismes est une étape nécessaire pour le bon déroulement de notre simulation.

#### 5.1.1 OLSR :

Comme nous avons vu au deuxième chapitre ce protocole échange quatre types de messages, dans les simulations qu'on a réalisées, les nœuds ont une seule interface par

conséquent aucun message MID est généré, il en résulte que le nombre de types de messages de contrôle utilisés sont de nombre de deux. La figure suivante indique le nombre des différents messages produits par chaque nœud du réseau en adoptant le mécanisme OLSR.

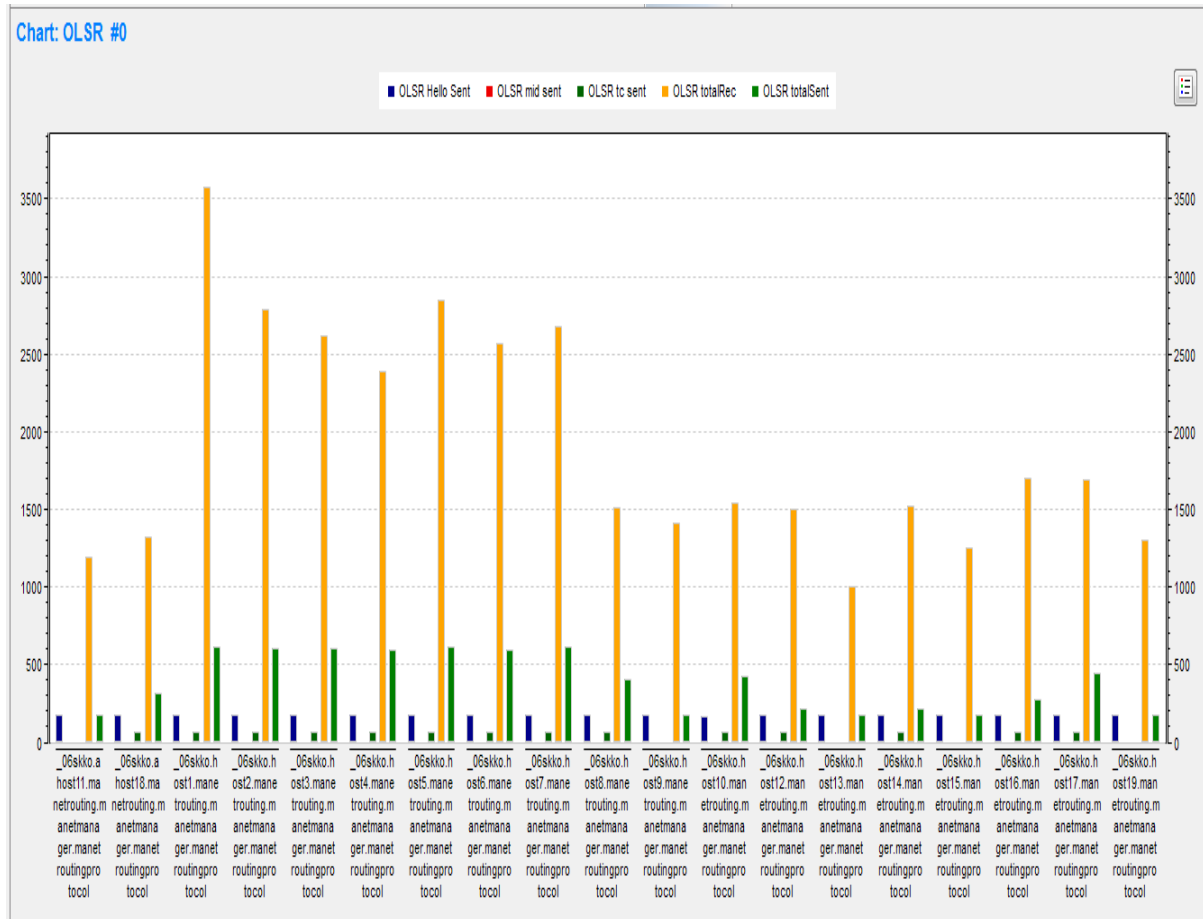
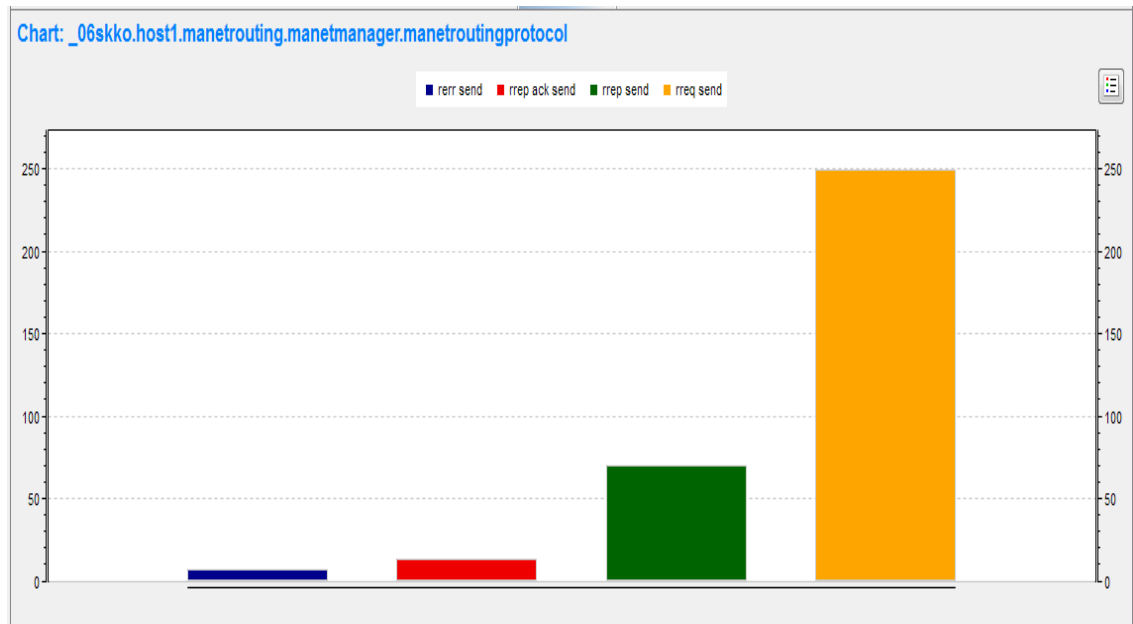


Figure 4.3 les messages OLSR

### 5.1.2 AODV :

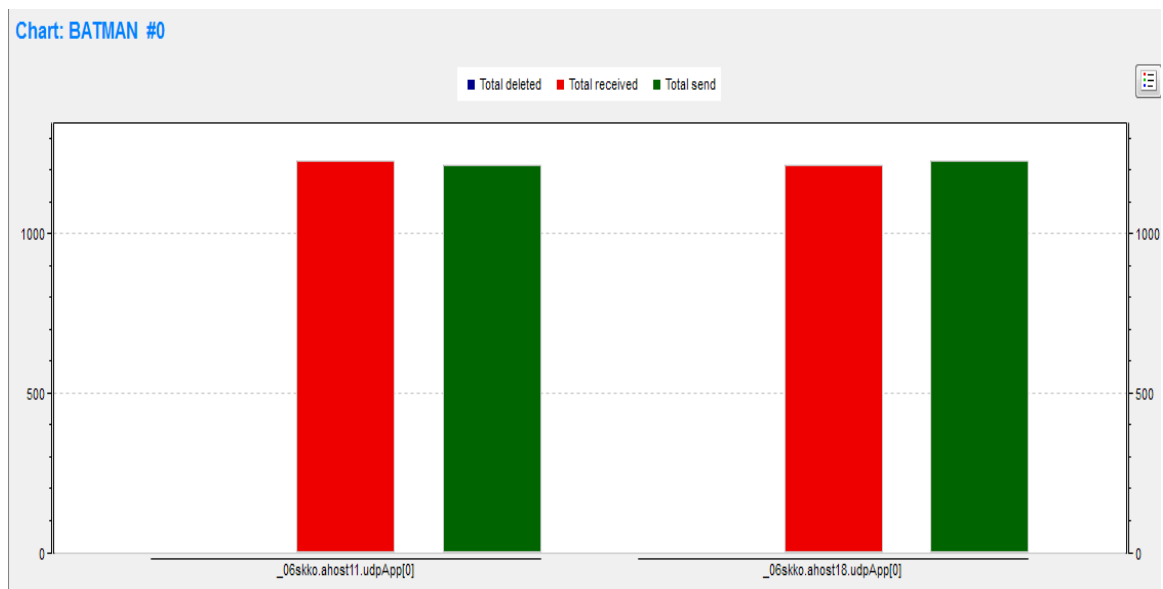
Le protocole AODV à son tour échange quatre types de messages, la figure suivante montre le nombre de chacun au sein du premier nœud de réseau.



**Figure 4.4** les messages AODV

### 5.1.3 BATMAN :

Dans ce mécanisme un seul message est échangé le nombre de message reçus et envoyés au sein des nœuds 11 et 18 sont illustrés dans la figure ci-dessous:



**Figure 4.5** les messages BATMAN

## 5.2 End-to-End delay

### 5.2.1 Configuration de simulation et les résultats

Le délai End-to-End est mesuré pour chaque hôte pour le trafic UDP généré par les valeurs d'initiation énoncées dans le tableau précédent. Le délai de bout-en-bout en moyenne pour l'ensemble des nœuds appartenant au réseau est calculé. Les résultats sont présentés dans les graphes suivants puis le délai moyen de chaque mécanisme est cité dans le tableau ci-dessous

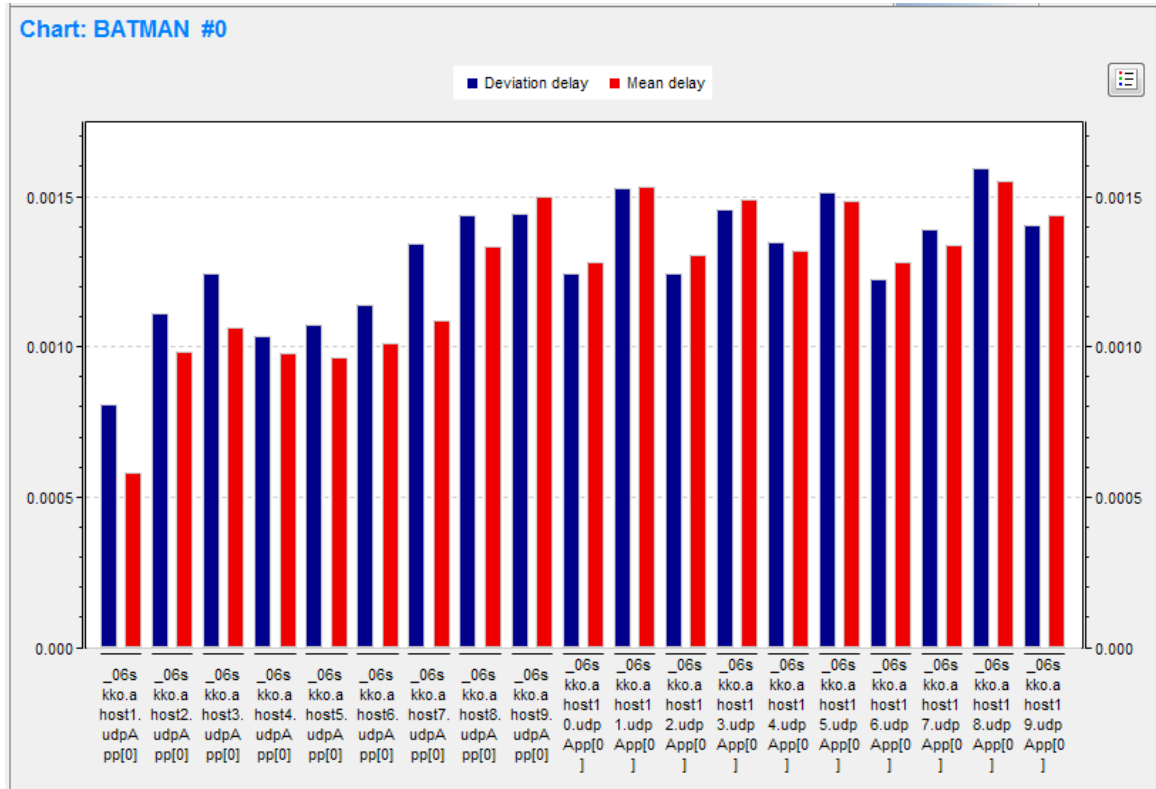


Figure 4.6 Mean and deviation end to end delays pour BATMAN

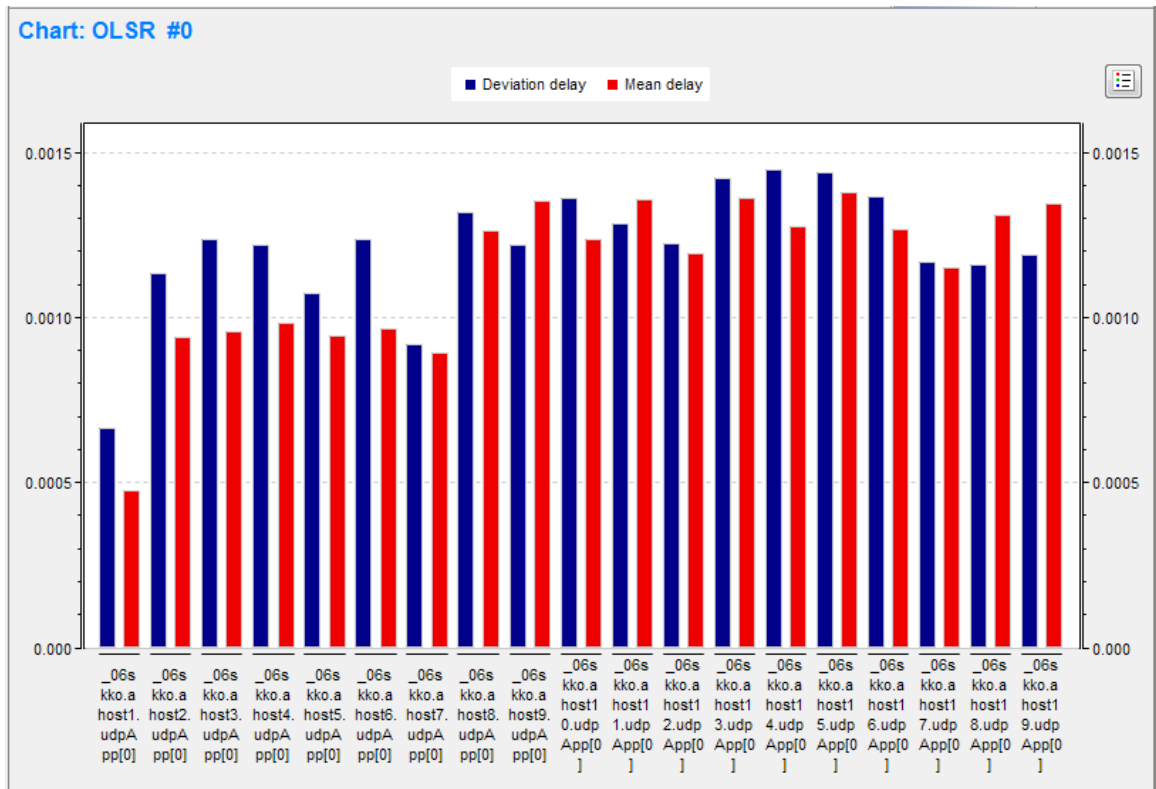


Figure 4.7 Mean and deviation end to end delays pour OLSR

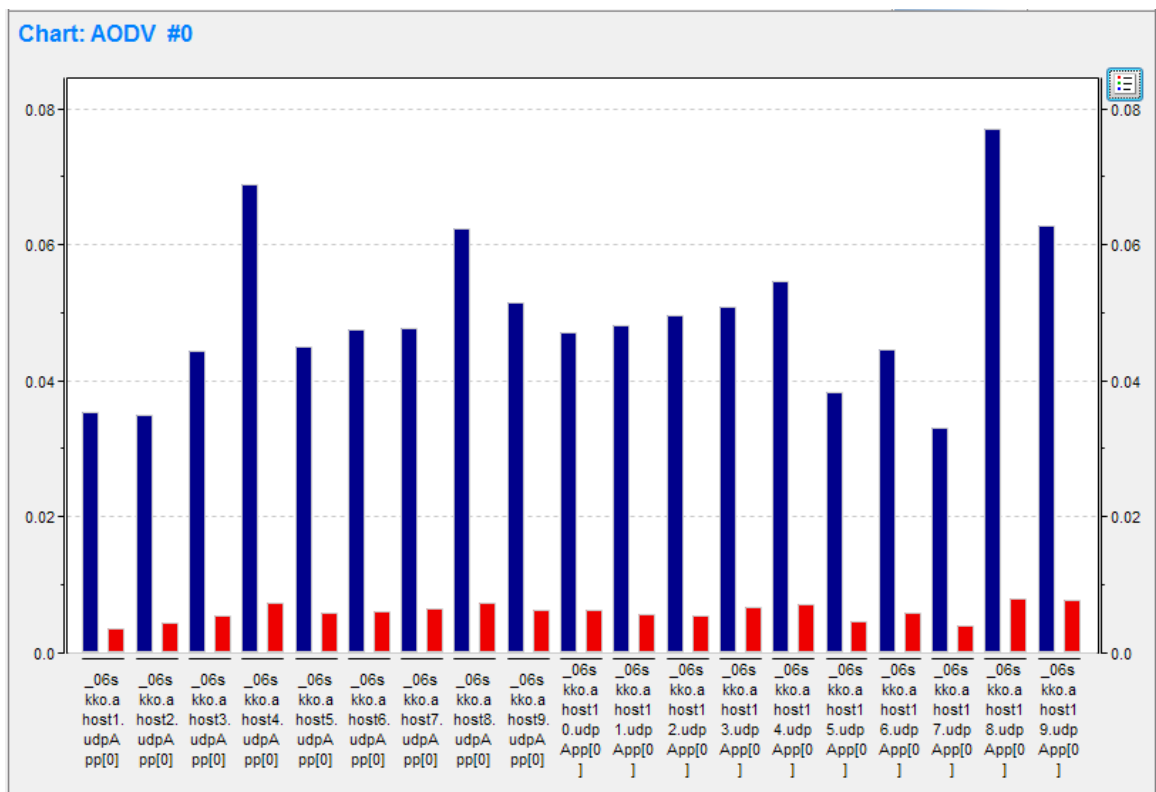


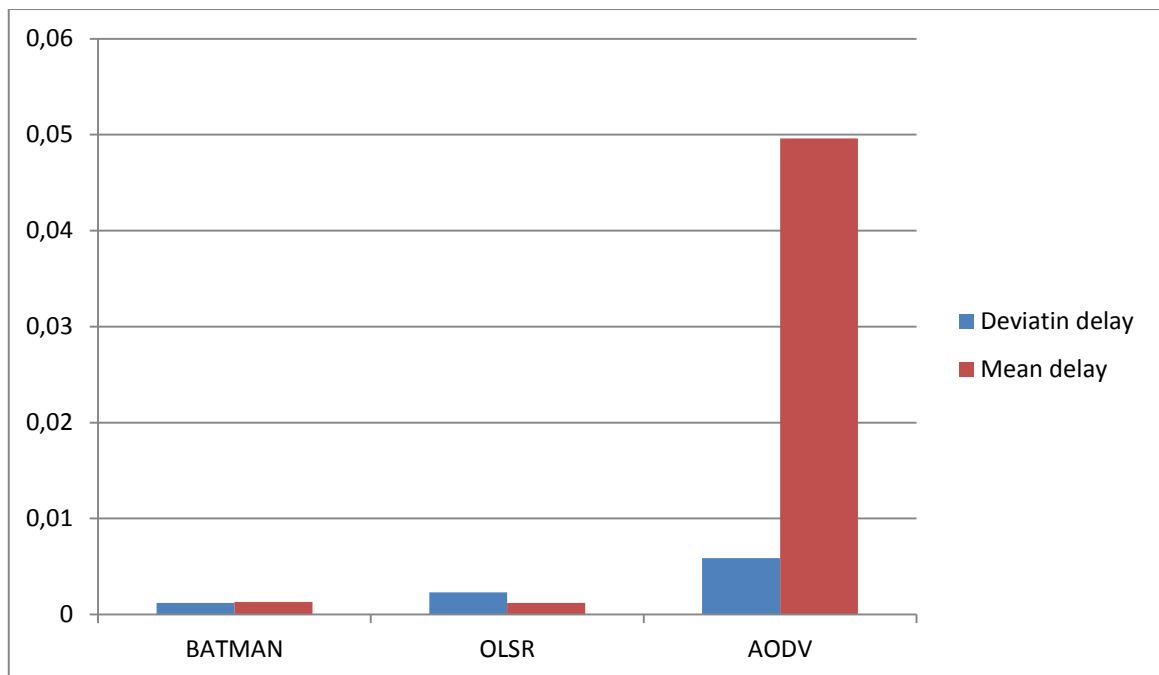
Figure 4.8 Mean and deviation end to end delays pour AODV



Le mécanisme de routage	End-to-End Mean Delay (en Seconde)	End-to-End Deviation Delay (en Seconde)
<b>B.A.T.M.A.N</b>	0,00120707461914	0,0012922711133
<b>OLSR</b>	0,0023050186742	0,00121357897827
<b>AODV</b>	0,0058645592736	0,04961046986842

**Table 4.2** délais de bout en bout

### 5.2.2 Interprétation des résultats :

**Figure 4.9** délais de bout en bout

AODV a un délai moyen de bout en bout plus élevé par rapport aux autres mécanismes de routage en raison de son caractère à la demande. Il prend du temps pour AODV d'établir des routes sur l'ouverture d'une session.

Ces résultats soulignent que l'information centralisée résultant de la table de routage proactif tirée du protocole OLSR est significativement efficace pour minimiser le retard.

Bien que BATMAN n'emploie pas les informations de routage centralisé, il a des retards presque aussi faibles qu'OLSR. Cela signifie que le concept OGM de Batman est très efficace et rapide pour sélectionner les meilleurs itinéraires.

### 5.3 Le taux de perte de paquets

#### 5.3.1 Configurations de simulation et les résultats

Le nombre des paquets UDP qui ne parviennent pas à atteindre leur destination est divisé par le nombre total de paquets. Ce ratio est variable durant la durée de la simulation pour chaque nœud du réseau.

Les résultats donc sont sous forme vectorielle :

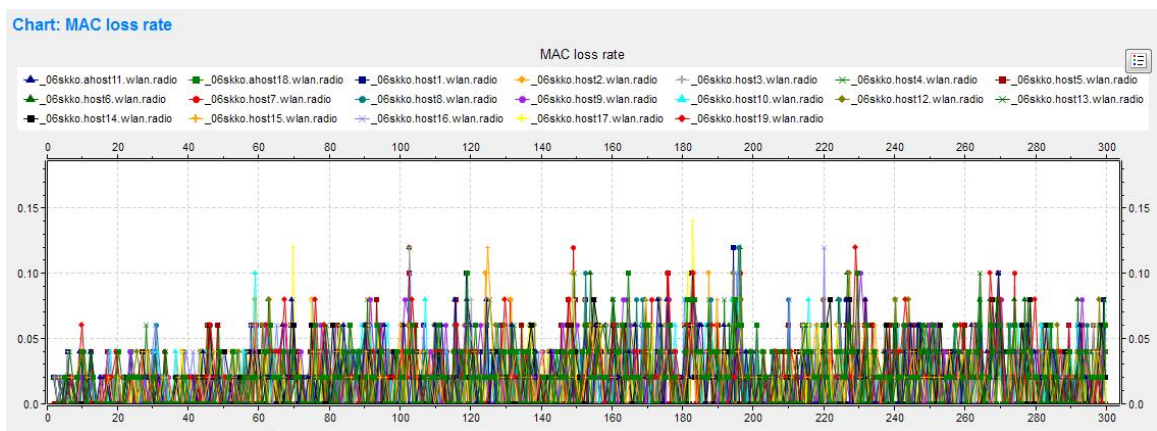


Figure 4.10 la perte de paquet pendant la durée de simulation de BATMAN

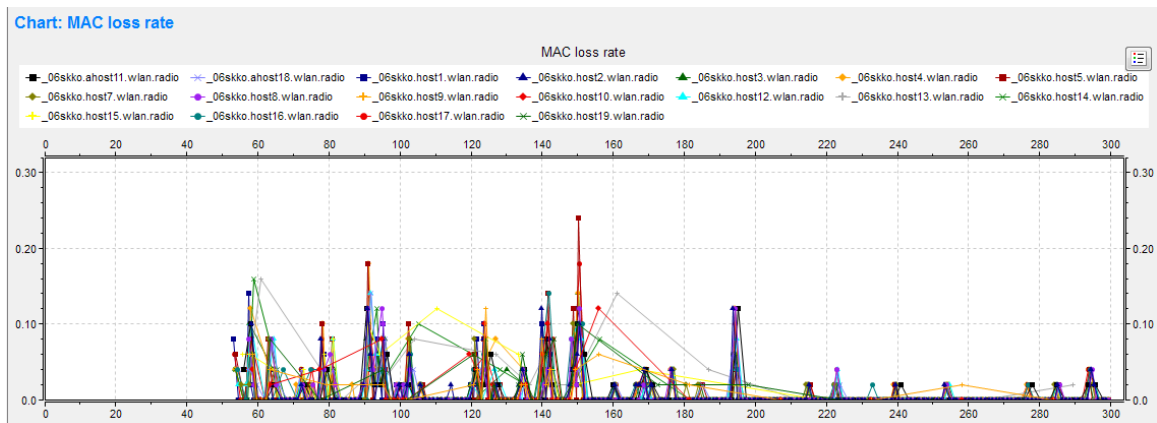
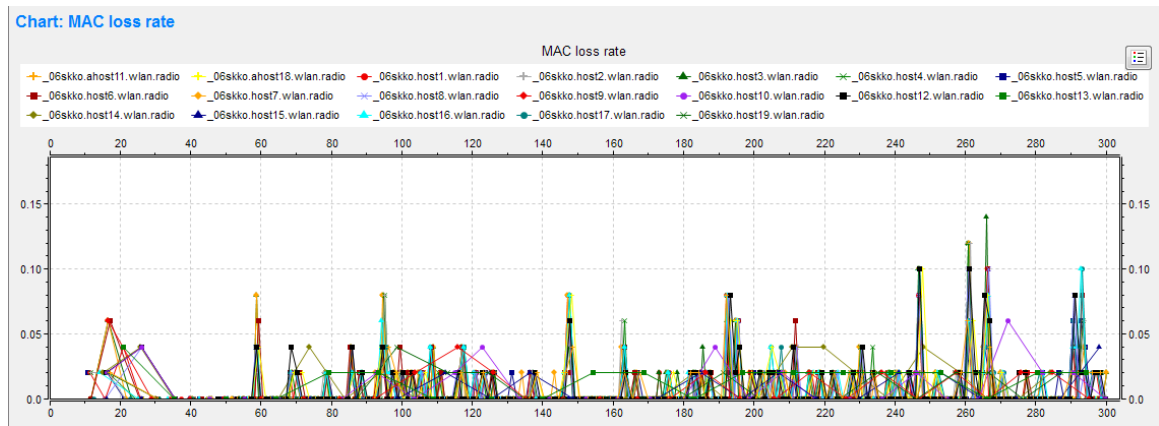


Figure 4.11 la perte de paquet pendant la durée de simulation d'AODV



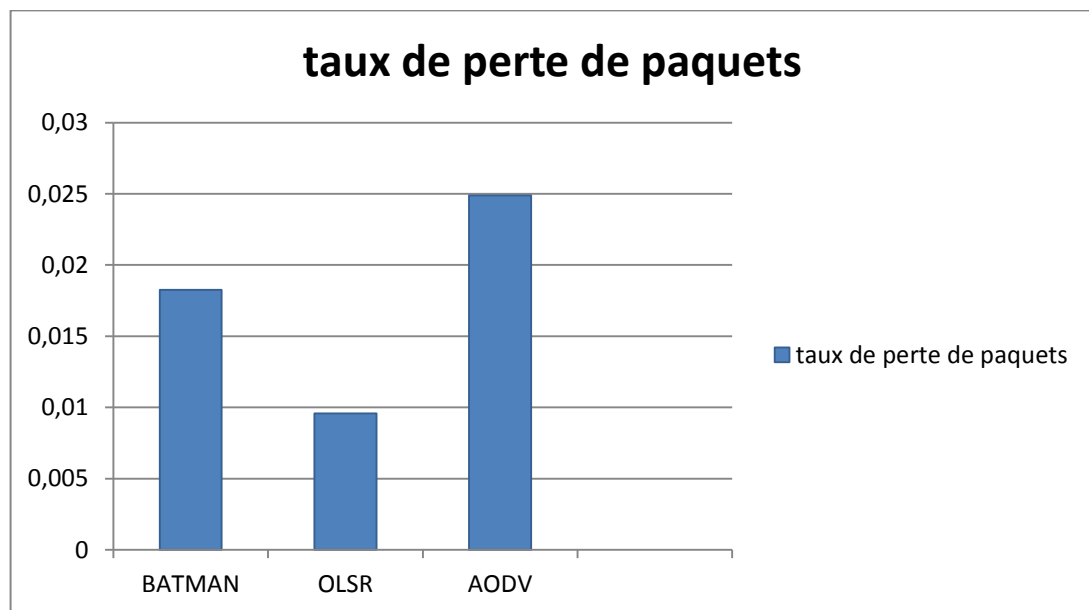
**Figure 4.12** la perte de paquet pendant la durée de simulation d’OLSR

Le taux moyen de perte de paquets est calculé pour tous les nœuds du réseau et est récapitulé dans la table ci-dessous :

Le mécanisme de routage	Le taux de perte de paquets
BATMAN	0,018263
OLSR	0,00956726
AODV	0,0248868

**Table 4.3** Le taux moyen de perte de paquets

### 5.3.2 Interprétation des résultats



**Figure 4.13** taux de perte de paquets

Nos résultats de simulation confirment les vérités déjà connues, que l'OLSR est nettement plus fidèle que le protocole AODV. Ceci est dû à la connaissance actualisée des chemins possibles vers les nœuds hôtes, et donc, l'OLSR est enrichie de plusieurs possibilités lui permettant de garantir presque toujours, un chemin pour délivrer le message. Alors que l'AODV a tendance à s'informer du réseau lui-même en l'inondant avec des messages pour reconstruire des chemins quand aucun chemin n'est spécifié ou bien sa fraîcheur est expirée.

Le BATMAN révèle une fidélité à délivrer les paquets mesurée avec un taux de perte qui se situe à mi-chemin entre les deux taux des deux autres protocoles. Ceci peut se comprendre, du fait de l'esprit du mécanisme du BATMAN qui essaye de minimiser l'exploration du réseau en maintenant que les chemins d'une certaine qualité, seulement en cas d'un besoin d'un chemin de moindre qualité, le BATMAN devrait mettre à jour sa table de routage, qui pourrait provoquer une perte de messages qui explique sa position entre les deux protocoles de bases dans le graphe.

#### 5.4 Per Flow Throughput (débit par flux):

##### 5.4.1 Configuration de simulation et les résultats

<b>udpApp[0].messageLength</b>	25000B/30000B/50000B/60000B
<b>Ahost12.udpAppType</b>	UDPBasicBust
<b>Ahost16.udpAppType</b>	UDPSink

**Table 4.4** valeurs d'initialisation spécifiques pour ce paramètre

Le débit est mesuré pour le trafic UDP entre "ahost12" et "ahost16". Dans cette approche, les valeurs calculées reflètent la capacité d'un lien unique plutôt que la capacité de l'ensemble du réseau. La fréquence d'un message de protocole UDP est ajustée de telle sorte que le débit binaire de la radio peut être utilisé de façon optimale et les files d'attente MAC ne soient pas inondées en raison du trafic réseau excessif.

Pour la détermination précise des performances du mécanisme de routage, on considère que le trafic UDP pour le calcul du débit, ce qui signifie que les messages de signalisation sont ignorés. Le débit de la liaison sélectionnée est mesuré comme suit:

Le mécanisme de routage	Débit pour la taille du msg UDP=25000B	Débit pour la taille du msg UDP=30000B	Débit pour la taille du msg UDP=50000B	Débit pour la taille du msg UDP=60000B
BATMAN	0,817424Mbps	0,7661565Mbps	0,7875945Mbps	0,781097Mbps
AODV	0,817424Mbps	0,097217 Mbps	0,0349105Mbps	0,03755 Mbps
OLSR	0,18724 Mbps	0,2028085Mbps	0,20569 Mbps	0,40603 Mbps

Table 4.5 débit par flux

#### 5.4.2 Interprétation de résultats :

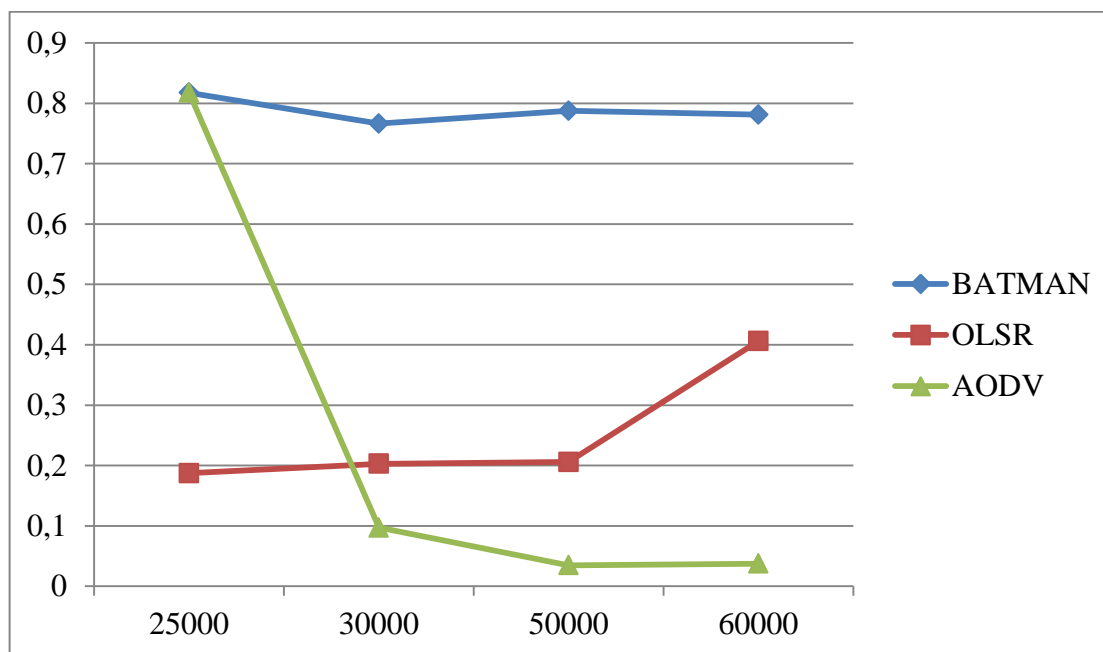


Figure 4.14 débit par flux

Nos résultats de simulation supportent la déclaration générale de la littérature que AODV fonctionne mieux dans les réseaux à faible charge de trafic, mais il provoque des frais généreux et des pertes de paquets, donc il a un débit plus faible pour les charges de trafic

supérieures. Ceci s'explique par le fait, que la perte d'un paquet de taille grande génère une chute sensible du débit.

D'autre part, OLSR achemine efficacement la charge lourde du trafic en raison de mises à jour périodiques et constant de ses tables de routage. Cette même procédure périodique consomme une grande part du débit, ce qui influe considérablement sur la valeur du débit, même si cette dernière croît avec la longueur du message. Du fait que, l'envoi de messages longs entre deux mises à jours successives des tables de routages, serait en faveur du débit.

BATMAN peut maintenir une stabilité à un débit élevé. Du fait qu'il ne génère pas une grande perte de paquet telle que le AODV, de plus, il n'est pas aussi coûteux en mises à jours que le OLSR.

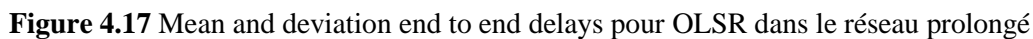
### 5.5.1 Scalabilité:

#### 5.5.1 Configuration de simulation et les résultats

Pour déterminer la compétence de scalabilité d'un mécanisme de routage, la topologie de réseau est altérée avec l'introduction des nœuds supplémentaires. Le but de cette simulation est de récolter des données afin d'évaluer les performances des mécanismes de routage dans un réseau plus large par rapport à l'autre que nous avons utilisé dans les sections précédentes. la topologie proposée avec 29 nœuds est illustrée dans la figure suivante :



**Figure 4.15** Topologie de simulation pour la scalabilité



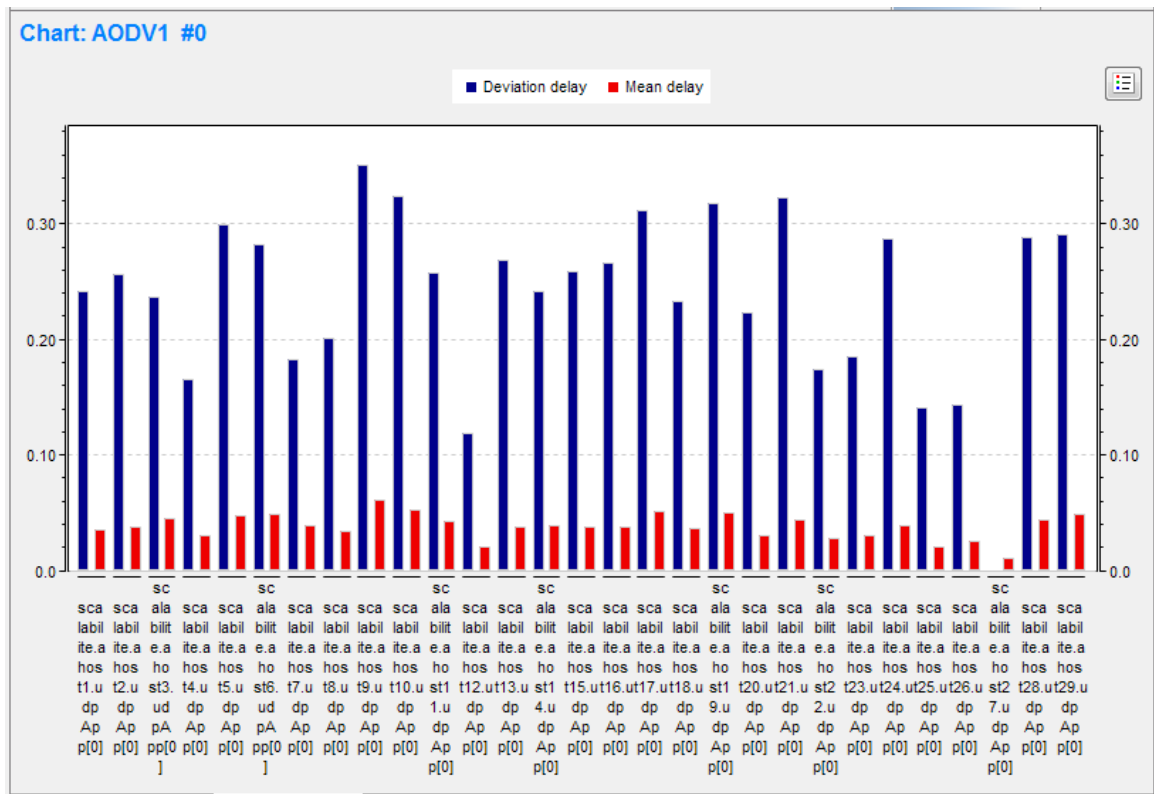


Figure 4.18 Mean and deviation end to end delays pour BATMAN dans le réseau prolongé

Le mécanisme de routage	End-to-End Mean Delay (en Seconds)	End-to-End Deviation Delay (en Seconds)
B.A.T.M.A.N	0,00446662274482	0,007025822790
OLSR	0,002968004779	0,0048233471334
AODV	0,0371328788696	0,24231705522758

Table 4.7 délai de bout en bout



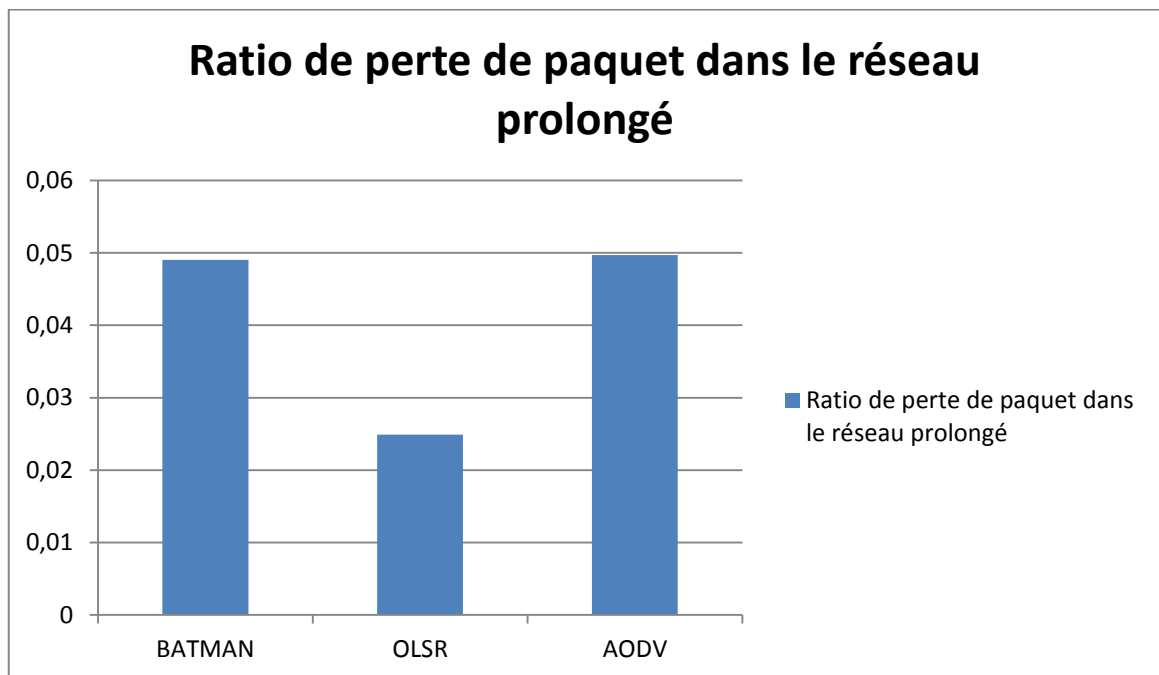
Le mécanisme de routage	l'accroissement du délai moyen de bout en bout	L'accroissement de déviation de délai de bout en bout
BATMAN	0,003259548125	0,005733551
OLSR	0,000662986104	0,003609768
AODV	0,031268319596	0,196212356

**Table 4.7** L'accroissement se moyen et déviation de délai de bout en bout

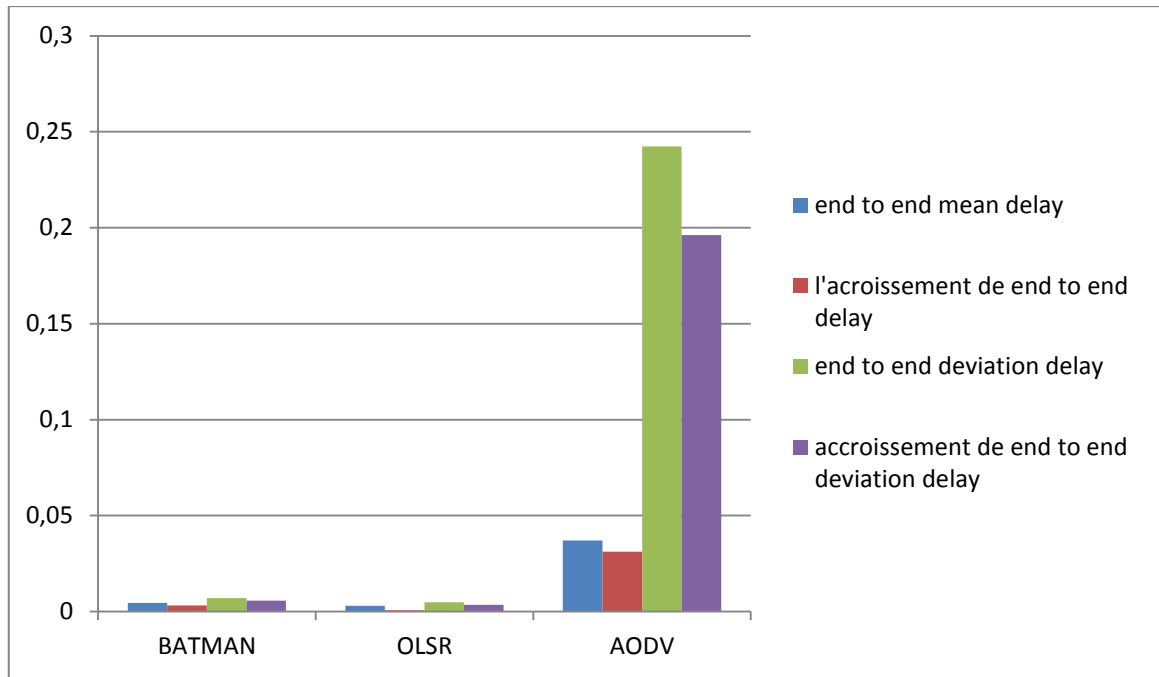
Le mécanisme de routage	Le taux de perte de paquets
BATMAN	0,04902817
OLSR	0,02487641
AODV	0,04967953

**Table 4.8** Taux de perte de paquets

### 5.5.2 Interprétation des résultats



**Figure 4.19** taux de perte de paquets dans la reseau prolongé



**Figure 4.20** les délais de bout en bout et leurs accroissement

L'observation du perte de paquets meme dans ce reseau prolongé révèle que BATMAN a réussi d'atteindre de bons résultats grace au concept d'OGM.

D'après les résultats de délai de bout en bout ,nous constatons que AODV souffre d'atteindre des resultats satisfaisants dans des réseaux plus larges car la metodologie à la demande requis des délais importants.

A l'opposition les mécanismes BATMAN et OLSR offrent des délais avantageux paraport à l'autre mécanisme.Cec trouve son explication dans le maintien des tables de routages et au concept proactif.

### **5.6 Latence de convergence :**

Les latences de convergence sont les durées exactes nécessaires avant que l'ensemble des nœuds du réseau complètent leurs tables de routage. La méthodologie d'évaluation de cette mesure n'est pas applicable pour AODV parce qu'il n'emploie pas une table de routage complète.

Les développeurs de BATAMAN affirment qu'il converge l'établissement des chemins plus vite qu'OLSR et les tests dans le monde réel soutiennent leur affirmation.

### **6 Conclusion :**

Dans ce chapitre nous avons réalisé un réseau maillé sans fils des clients en respectant toutes les contraintes de ce type de réseaux en terme physique et configurations. Puis nous avons effectué des centaines de simulations afin de confirmer les résultats finaux. Les simulations qu'on a faites sur deux réseaux constitués de 19 nœuds respectivement 29 nœuds affirment suffisamment les attentes des développeurs de BATMAN ainsi ils montrent les lacunes de l'AODV surtout dans les réseaux denses.

Donc le BATMAN révèle des résultats prometteuses et comparables à celles du OLSR ce qui signifie qu'il est fourrageur et puisque ses tables de routages sont moins chargées par rapport aux tables d'OLSR du fait de leur concept OGM.