I Introduction :

Les réseaux informatiques connaissent une expansion importante grâce à plusieurs moyens qui ont pu se développer au cours du temps, donc il est très coûteux de déployer un banc d'essai complet contenant plusieurs ordinateurs, des routeurs et des liaisons de données pour valider et vérifier un protocole de réseau ou un certain algorithme spécifique. C'est pour cela les simulateurs de réseaux viennent pour pallier à ce problème.

Les simulateurs du réseau offrent beaucoup d'économie de temps et d'argent pour l'accomplissement des tâches et sont également utilisés pour que les concepteurs des réseaux puissent tester les nouveaux protocoles ou modifier les protocoles déjà existants d'une manière contrôlée et productrice.

La problématique étudiée dans ce chapitre étant la simulation des réseaux sans fil, et en particulier le wifi, la méthode de simulation à événements discrets s'avère la plus adéquate pour notre cas. En effet, il est facile de modéliser un réseau sans fil sous la forme d'entités (les nœuds sans fil) et de modéliser les interactions entre elles aux moyens d'événements, comme l'événement de "*transmission d'un paquet*" ou "*réception d'un paquet*". Nous présentons dans ce qui suit le déroulement des étapes de simulation à événements discrets, que nous avons menée dans ce travail de fin d'études et consistant à simuler l'impact du Direct Link (mécanisme de QoS introduit par le standard 802.11^{e} et qui n'a jamais été simulé auparavant dans les différents modules de simulation du WIFI avec QoS présents dans l'état de l'art). D'où notre intérêt pour évaluer ce mécanisme sur les performances globales du réseau et donc ceci permettra de savoir s'il est intéressant d'implémenter ce mécanisme dans les AP par les constructeurs ou pas, dans la mesure ou leur implémentation apporterait un réel bénéfice dans l'amélioration de la QoS des applications.

II Simulation

Simuler c'est modéliser un système complexe, afin de prévoir son comportement dans le monde réel. Il s'agit d'une approche permettant de représenter le fonctionnement d'un système réel constitué de plusieurs entités, de modéliser les différentes interactions entre elles, et enfin d'évaluer le comportement global du système et son évolution dans le temps.

Le recours à la simulation permet de contourner les limites de la complexité des modèles analytiques. Toutefois, il est nécessaire de bien identifier les caractéristiques du système afin de la représenter, le plus finement possible, par des modèles abstraits.

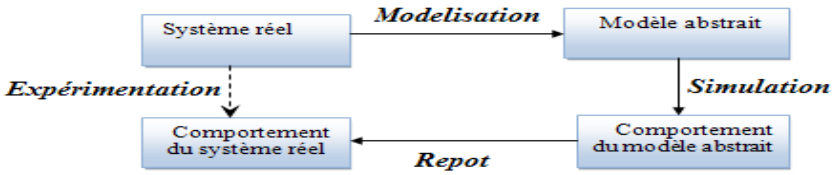

Figure 4.1 : Cycle modélisation-simulation

Si la représentation du système réel par des modèles abstraits est suffisamment réaliste et précise, il est alors possible de reporter les résultats obtenus avec ces modèles sur le système réel. Le cycle correspondant aux étapes de modélisation, simulation et report des résultats est illustré dans la figure 4.1. [25]

III Choix du simulateur

Les simulateurs réseaux sont utilisés par des personnes de différents domaines tels que les chercheurs universitaires, industriels et, d'assurance de qualité (AQ) pour concevoir, simuler, vérifier et analyser les performances des protocoles de différents réseaux. Ils peuvent également être utilisés pour évaluer l'effet des différents paramètres des protocoles étudié. En général, un simulateur de réseau est composé d'un large éventail de technologies et de protocoles réseaux et aide les utilisateurs à construire des réseaux complexes à partir de blocs de construction de base comme des grappes de nœuds et de liens. Avec leur aide, nous pouvons concevoir différentes topologies de réseau en utilisant différents types de nœuds tels que les nœuds terminaux, concentrateurs, ponts de réseau, routeurs, des dispositifs optiques de couche de liaison, et des unités mobiles.

Il existe plusieurs simulateurs réseaux tel que les simulateurs NS-2 et NS-3, OPNET.

Le simulateur NS-2 a été un simulateur populaire pour la recherche et l'éducation sur les systèmes Internet, dont notamment mobiles, les systèmes sans fil. NS-2 bénéficie d'utilisation répandue dans le milieu de la recherche, le code de simulation a été contribué par plus de cent personnes et organisations, et l'utilisation du simulateur est toujours référencé dans de nombreux travaux de recherche en réseau.

Cependant, une lacune majeure de NS-2 est son évolutivité en termes d'utilisation de la mémoire et du temps d'exécution de la simulation. Ceci est particulièrement un problème en ce qui concerne les nouveaux domaines de recherche dans les réseaux

informatiques, tels que les réseaux de capteurs sans fil, les réseaux peer-to-peer ou des architectures maillés qui exigent une simulation de réseaux très larges.

Outre NS-2, plus d'une douzaine de simulateurs des réseaux sont actuellement utilisés dans les universités et l'industrie. Parmi les simulateurs les plus connus nous choisissons le simulateur NS-3 pour réaliser notre travail.

Des travaux de comparaison entre NS-2 et NS-3 ont été réalisés et ont montré que NS-3 est meilleur que NS-2 de plusieurs façons, notamment:

- Un noyau logiciel remanié pour améliorer l'évolutivité et l'extensibilité, y compris le soutien pour les simulations distribuées.
- Une architecture pour soutenir la création de logiciels réseaux open source tels que les machines virtuelles, les piles de protocoles, les démons de routages et de paquets analyseurs de traces.
- La mise à disposition de nouveaux modèles sans fil pour IEEE 802.11, et éventuellement d'autres modèles tels qu'IEEE802.16.
- Une capacité de réseau réorganise l'émulation,
- Une version révisée de recherche et de collecte de statistiques.

En outre, NS-3 regroupe un grand nombre de mécanismes fondés sur le succès et évite les échecs de NS-2. [26], [27]

IV Présentation du Simulateur NS3

NS-3 est conçu pour remplacer le NS-2 courant populaire. Toutefois, NS-3 n'est pas une version mise à jour de NS2. NS-3 est un nouveau simulateur et il n'est pas rétro-compatible avec NS-2.

NS-3 est un simulateur réseau à évènements discrets. Il vise à remplacer son prédécesseur NS-2, écrit en C++ et OTcl (version orientée objet de Tcl), pour tenter de remédier à ses limites (mauvaise gestion des traces ou encore, plus gênant l'utilisation de multiples interfaces sur un noeud…). NS-3 est écrit en C++ et Python, et peut être utilisé sur les plateformes Linux/Unix, OS X (Mac) et Windows.

Son développement a d'abord commencé en Juillet 2006, et devait durer quatre ans, Il est financé par les instituts comme l'Université de Washington, Georgia Institute of Technology et le Centre de l'ICSI pour la recherche sur Internet, la première version majeure publique et stable a été publiée en juin 2008.

Les développeurs de NS-3 ont décidé que l'architecture de simulation devait être remaniée complètement en partant du zéro. Dans cette optique, l'expérience tirée de

NS-2 doit être associé avec les progrès des langages de programmation et du génie logiciel. L'idée de la rétrocompatibilité avec NS-2 a été abandonné dés le départ. Cela libère NS-3 de contraintes héritées de NS-2 et permet la construction d'un simulateur qui est bien conçu depuis le début. [27], [28], [29].

V Terminologie et abstractions

Il est important de bien comprendre le sens des termes employés au sein du simulateur, ainsi que les abstractions qui ont été faites.

NS-3 utilise des termes largement employés dans le domaine des réseaux, mais qui peuvent avoir une signification particulière au sein du simulateur. Voici les principaux :

- **Un nœud *Node :***

Représente tout élément de réseau. La composition d'un Node peut être gérée (ajout de composants, de protocoles, d'applications).

- **Une application *Application* :**

Représente un code exécuté par un utilisateur. Ce code peut être nécessaire au déroulement d'une simulation. L'échange de paquets durant une simulation nécessite par exemple la description d'une Application au sein des nœuds participants (par exemple, *UdpEchoClientApplication* d'un côté et *UdpEchoServerApplication* de l'autre pour réaliser une application en mode client/serveur). NS-3 ne fait pas de distinction entre les "applications système" (souvent exécutées par le noyau) et les applications des utilisateurs (exécutées dans le user-space). Les applications peuvent ensuite être attachées à un Node.

- **Un canal de communication *Channel***

Représente le lien qui relie des nœuds, ou plus exactement les NetDevices installés dans les nœuds. Des spécialisations de cette classe sont définies, comme par exemple *CsmaChannel* pour modéliser un lien Ethernet utilisant CSMA, ou encore *WifiChannel* pour modéliser un lien WiFi.

- **Une interface de communication, ou interface réseau :**

Appelée **NetDevice**, qui modélise à la fois l'équipement (carte réseau) et le pilote dont un ordinateur a besoin pour pouvoir communiquer avec d'autres. Des spécialisations de NetDevice sont fournies, comme par exemple *CsmaNetDevice* qui simule une carte réseau Ethernet et peut être reliée à un *CsmaChannel*, ou encore **WifiNetDevice** pour un lien à un canal de type *WifiChannel*.

Pour établir une connexion entre deux nœuds, il faut alors :

- Équiper chaque nœud de *NetDevice.*
- Configurer la couche protocolaire sur chaque nœud (élément non représenté qui se situe entre les applications et les NetDevice).
- Dans le cas d'une couche protocolaire TCP/IP, configurer les adresses MAC et IP sur chaque NetDevice.
- Créer le canal de communication correspondant.
- Connecter chaque NetDevice au canal de communication.

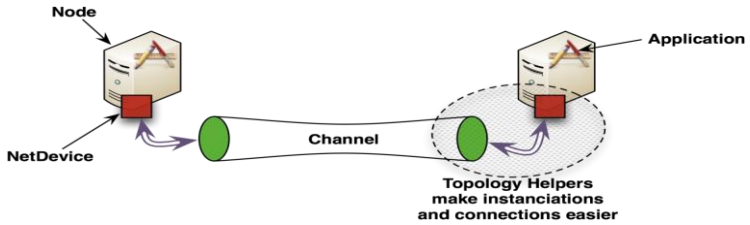

Figure 4.2 : Établissement d'une connexion entre deux nœuds

S'il s'agit de connecter un grand nombre de nœuds les uns avec les autres, ce processus peut être très lourd. NS-3 fournit des **TopologyHelpers** pour faciliter ce genre de tâches. Les classes représentant ces objets sont dans le dossier *NS-3.10/src/helper*. On y trouve par exemple :

- un *CsmaHelper* pour instancier des *CsmaNetDevice* sur un nœud, instancier un *CsmaChannel* et les connecter.
- un *WifiHelper*, qui a la même fonction que le CsmaHelper, mais pour une interface et un canal qui utilisent le *WiFi.*
- un I*nternetStackHelper* pour instancier au sein d'un nœud la couche protocolaire *IP/TCP/UDP* et des fonctions de routage (*IPv4/IPv6*). [30]

VI Modules du simulateur NS3 :

NS-3 fournit différents modules qui peuvent être modifiés et effectivement utilisés. L'organisation du logiciel de NS-3 est illustrée dans la figure 4.4. Voici les modules de base de NS-3 :

- ***Core Module:***

Module de base, il est indépendant, il permet la création d'une structure de classes hiérarchiques dérivant de la classe Object de base. Cette hiérarchie possède plusieurs fonctionnalités conçus spécialement pour répondre aux besoins de la simulation.

Parmi lesquels : l'agrégation d'objets, l'enregistrement actif(TypeId) avec les attributs publiques…etc.

- ***Common Module :***

Ce module gère les actions liées à la génération et la réception des paquets, ce module est centré sur la *classe Packet* utilisé pour la simulation de réseau au niveau packet. L'utilisation de cette dernière classe est conçu de façon à optimiser la gestion de la mémoire en utilisant la méthode (*copy on write*) et en donnant la possibilité de manipuler des paquets vides et dont la taille est le paramètre suffisant et nécessaire pour la simulation. La plupart des autres modules ont utilisé les fonctionnalités de ce module.

- ***Simulator Module*** :

Les Simulations des événements sont gérées par le module de simulation. Il prévoit explicitement la possibilité de planifier des événements à des moments différents et ensuite d'exécuter ces événements. La classe *Time* représente le temps simulé à haute résolution en utilisant un entier de 128 bits. Cette classe est la classe la plus importante du simulateur.

- ***Mobility Module :***

Ce module permet de définir la position des nœuds et associe un modèle de mouvement aux agents de la simulation. NS-3 fournit sept modèles de mobilité.

- ***Node Module***

La classe Node peut contenir plusieurs *NetDevices*. Chaque *NetDevice* est attaché à un *channel* par lequel il envoie et reçoit des paquets.

Un nœud peut contenir plusieurs gestionnaires de protocole, qui acceptent les paquets reçus par le NetDevice. Pour lancer la transmission des paquets, chaque nœud peut également contenir une liste d'applications.

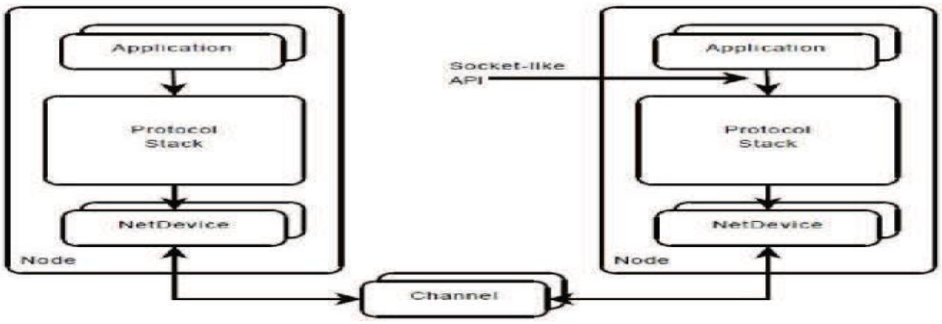

Figure 4.3 : Architecture du nœud NS-3

- ***Helper Module :***

Ce modèle peut être considéré comme un emballage de haut niveau. Il facilite la construction des scénarios complexes de simulation et l'installation des différents modules dans des agents différents.

- ***Application Module :***

Certaines applications sont intégrées et fournies par NS-3. Elles sont installées dans les nœuds et peuvent être démarrées/arrêtées à des moments précis dans la simulation.

- ***InternetStack Module :***

Les classes de ce module définissent les protocoles TCP/IP des couches réseaux trois et quatre (TCP/IP).

- ***Devices Module :***

Les composants de ce type représentent des périphériques réseaux et transmettent des paquets via un canal virtuel à d'autres instances de la même classe *NetDevice*.

- ***Routing Module :***

Deux algorithmes de routage sont disponibles dans NS-3. Le premier appelé *GlobalRouter*, utilise des routes statiques et le deuxième met en œuvre le protocole *OLSR* pour les réseaux dynamiques ad-hoc.

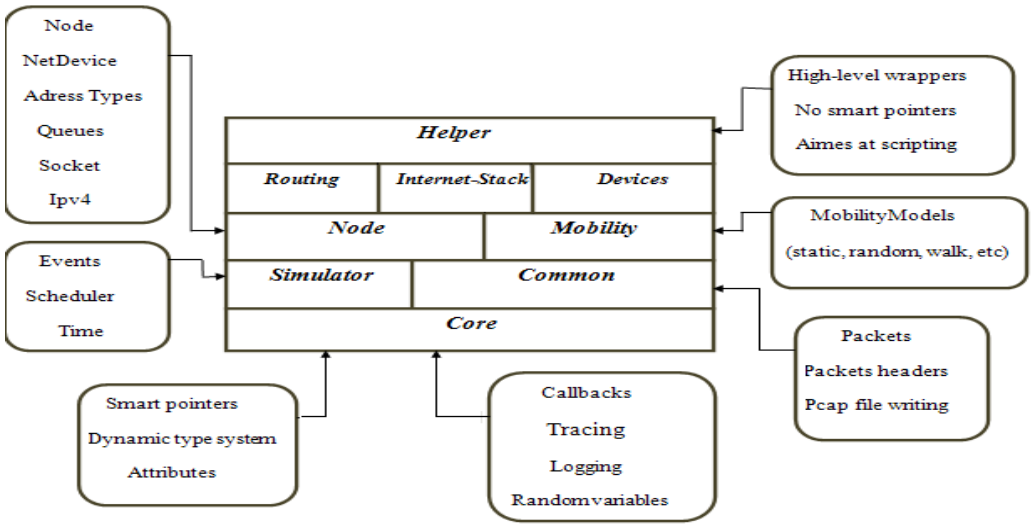

Figure 4.4:Les principaux modules de NS3

VII Modèles et mise en œuvre de réseaux Wifi dans ns-3

Pour mettre en œuvre des réseaux Wifi (802.11) dans NS-3, tout doit être décrit, des couches les plus basses (canal de communication) aux éléments à mettre en œuvre dans les nœuds (interfaces, couches physique et MAC qu'elles utilisent).

Nous avons détaillé quelques classes de base et les modèles fournis dans NS-3.

VII.1 Classes de base :

- **WifiNetDevice :**

Tout d'abord, un nœud souhaitant communiquer en Wifi doit disposer d'une **WifiNetDevice.** Cette classe, qui hérite de la classe générique **NetDevice**, est définie dans le fichier */src/devices/wifi/wifi-net-device.h* et implémentée dans le fichier /src/devices/wifi/wifi-net-device.cc. En plus des éléments caractérisant une **NetDevice** (notamment le nœud auquel elle est attachée), elle est caractérisée par :

- Un modèle de la couche physique **WifiPhy**
- Un modèle de la couche MAC **WifiMac**
- Un **WifiRemoteStationManager**, qui maintient une liste des stations connectées au réseau sans fil, des informations sur leur état, et qui incarne un algorithme d'adaptation du débit.

- **WifiChannel :**

Cette **WifiNetDevice** doit être connectée à un **WifiChannel**. La classe **YansWifiChannel** est la seule implémentation d'un modèle de canal wifi. Elle est caractérisée par :

- Une liste contenant des pointeurs vers les modèles de couche physique de chaque nœud connecté, qui doivent être du type **YansWifiPhy.**
- Un modèle de **perte/atténuation** lors de la propagation.
- Un modèle de **délai de propagation**.

La classe **WifiChannel** peut être utilisé pour relier un ensemble d'interfaces réseau **WifiNetDevice**. La classe **WifiPhy** est la partie dans le **WifiNetDevice** qui reçoit les bits du canal. Le **WifiPhy** modélise un canal 802.11 en termes de fréquences, de modulation, de débit binaire et interagit avec le **PropagationLossModel** et **PropagationDelayModel** trouvés dans le canal. La couche physique peut être dans l'un des trois états : **TX (**Transmission**), RX (**Reception**)** ou **IDLE (**neutre**).**

VII.2 Les modèles :

- **Attachés à une WifiNetDevice :**

i. Modèle de couche physique :

Le modèle de couche physique qu'utilise une **WifiNetDevice** conditionne sa capacité à envoyer ou recevoir des signaux et la façon dont elle le fait. Ce modèle a pour vocation de traiter des considérations telles que la puissance d'émission/réception

des interfaces, la bande de fréquence utilisée selon le standard 802.11 choisi, les interférences, etc.

ii. Modèle de couche MAC :

Le modèle de couche MAC gère l'accès au canal, les éléments d'identification du réseau sans fil (**SSID**), l'association/désassociassions des nœuds et leur adressage physique (adresse MAC). Le modèle de couche MAC est décrit dans NS-3 par la classe **WifiMac.** Le fichier */src/devices/wifi/wifi-mac.cc* contient l'implémentation d'un certain nombre de méthodes renvoyant des valeurs par défaut et de **callbacks** (**TraceSource** notamment), mais ne propose pas d'implémentation du modèle en soit. La classe **RegularWifiMac** est une implémentation générique de ce modèle de couche MAC qui traite de la majorité des fonctionnalités de cette couche.

Il existe actuellement six modèles Mac de haut niveau, trois pour le Mac sans gestion de la QoS et trois avec la gestion de QoS, dans chaque groupe (avec et sans QoS) on trouve les modèles suivants :

- **AdhocWifiMac :**

Surcharge quelques méthodes pour représenter le fonctionnement des nœuds en mode **AdHoc**.

- **ApWifiMac :**

Implémente les méthodes permettant l'envoi de beacons, l'association/désassociassions et l'authentification des stations auprès d'un point d'accès.

- **StaWifiMac :**

Implémente les méthodes qui permettent la réception de beacons émis par des points d'accès et la gestion de l'état d'une station en termes d'association.

Avec les modèles Mac QoS, il est possible de travailler avec un trafic appartenant à quatre classes d'accès différentes :

AC_VO	Pour le trafic voix
AC_VI	Pour le trafic vidéo
AC_BE	Pour le trafic best-effort
AC_BK	Pour le trafic d'arriere-plan

Figure 4.5 : Classe de trafic

Afin de déterminer la classe d'accès, chaque paquet provenant d'un Mac autre que le wifi et transmis vers un Mac Wifi doit être marqué au moyen d'un objet **QoSTag**

afin d'associer un **TID** (trafic id) pour ce paquet. Dans le cas où un **QoSTag** est *absent* le paquet sera considéré comme appartenant à la classe d'accès *AC_BE*.

- **Attachés au WifiChannel :**

L'implémentation fournie par la classe **YansWifiChannel** admet l'utilisation de modèles permettant de simuler la perte de puissance (atténuation) d'un signal et le délai de propagation sur le canal.

Nous avons détaillés les modèles proposés dans NS-3.

i. Modèle de perte/atténuation lors de la propagation :

Ce modèle permet de simuler la perte de puissance (atténuation) d'un signal évoluant sur un canal de transmission. Il permet en fait de calculer la puissance de réception d'un nœud destination, ce qui permet de déterminer s'il est susceptible de recevoir le signal. Ce calcul repose sur la puissance d'émission du nœud source et la position des nœuds source et destination (qui dépend d'un modèle de mobilité).

ii. Modèle de délai de propagation :

Ce modèle permet de simuler le délai de propagation sur le canal de transmission. Le calcul de ce délai repose sur la position des nœuds source et destination (qui dépend d'un modèle de **mobilité**).

VII.3 Les helpers :

Les helpers comme pour tout ce que l'on peut définir dans NS-3, les instanciations puis les configurations des éléments instanciés se font principalement en passant par les helpers fournis. Concernant la mise en œuvre de réseaux Wifi, tout commence par l'utilisation d'un premier helper : **Le WifiHelper.**

- **WifiHelper :**

Ce helper sert à la création et à l'installation de **WifiNetDevice pré-configurées** sur différents nœuds.

- **WifiPhyHelper :**

Ce helper a pour vocation de permettre la création et la configuration de la couche physique à mettre en œuvre dans les **WifiNetDevice**.

- **WifiMacHelper :**

Ce helper a pour vocation de permettre la création et la configuration de la couche MAC à mettre en œuvre dans **lesWifiNetDevice.** Seulement, aucune implémentation n'est directement fournie par cette classe. Par contre, deux classes en héritent et fournissent une implémentation :

i. La classe NqosWifiMacHelper :

Permet d'instancier un modèle de couche MAC simple, qui ne gère pas la qualité de service(QoS).

ii. La classe QosWifiMacHelper :

Permet d'instancier un modèle de couche MAC qui gère la qualité de service : définition de classes de service, gestion des files d'attentes associées etc. Il faut donc utiliser, au choix, un de ces deux helpers pour instancier et configurer un modèle de couche MAC.

Les constructeurs de ces deux classes n'instancient rien du tout. Une méthode **Default ()** définie dans chacun de ces helpers initialise à la fois le modèle de couche MAC à utiliser et le fait qu'il supporte ou non la QoS :

- ✓ **pour un NqosWifiMacHelper :**

Le modèle est initialisé à *AdhocWifiMac* et l'attribut **QosSupported** à *false*.

- ✓ **pour un QosWifiMacHelper :**

Le modèle est initialisé à *StaWifiMac* et l'attribut **QosSupported** à *true*.

Un des intérêts de NS-3, c'est de pouvoir faire aux nœuds un certain nombre de traitements, que ce soit pour tester des protocoles ou pour évaluer des approches à plus haut niveau. Ces traitements doivent être développés en utilisant le concept d'application. Nous avons optés pour les applications de type OnOffApplication dans nos expérimentations afin d'étudier et de tester le protocole de lien directe. [30], [32]

VIII Notre proposition

Dans notre travail nous avons utilisé une machine virtuelle sur laquelle nous avons installé le système linux fédora.

Nos simulations ont été faites sous NS3 qui possède un module qui permet de simuler le réseau Wifi avec une variété de mécanismes de qualités de service.

Nous avons ainsi utilisés le logiciel Wireshark pour l analyse de nos fichiers trace et le logiciel Matlab pour la représentation graphique de nos résultats.

Notre application est réalisée selon les deux scripts suivants :

- ***Le premier script***

Dans notre premier script et dans un premier cas nous avons créé un réseau wifi composé de deux stations et un point d'accès qui supportent la QoS avec un adressage aléatoire où la station de base du réseau n'est pas spécifiée. Donc dans ce cas là les stations vont communiquer via le point d'accès.

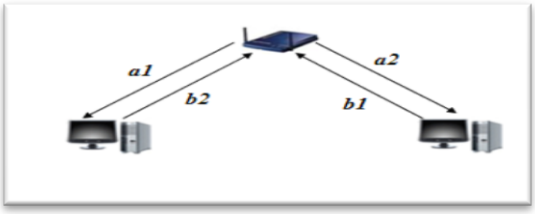

Figure 4.6 : Réseau wifi test

Dans le deuxième cas nous essayons d'augmenter le nombre des stations communiquant via le point d'accès jusqu'à se qu'on arrive à une saturation du point d'accès du réseau, ce qui nous permet de découvrir le nombre de stations supporté par ce dernier.

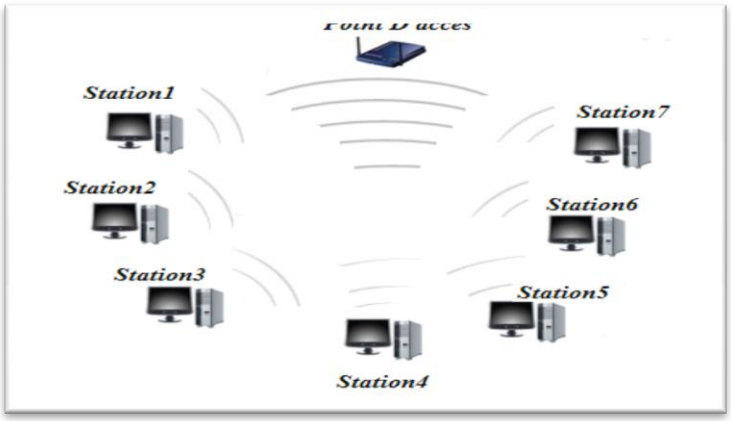

Figure 4.7 : Augmentation du nombre des stations

- ***Le deuxième script :***

Dans ce deuxième script nous avons créé une topologie réseau mixte contenant une infrastructure wifi, utilisant un AP relié à un réseau LAN Ethernet filaire.

Dans cette topologie la communication se fait entre le premier nœud du réseau infrastructure wifi et le dernier nœud du réseau LAN tout en passant par le point d'accès du réseau infrastructure wifi. Nous avons considéré que les stations sont dans le même BSS donc les stations vont communiquer directement, et le point d'accès n'interviendra que dans le cas d'une communication entre une station wifi et une autre station dans un autre réseau. (Dans notre cas c'est le réseau LAN).

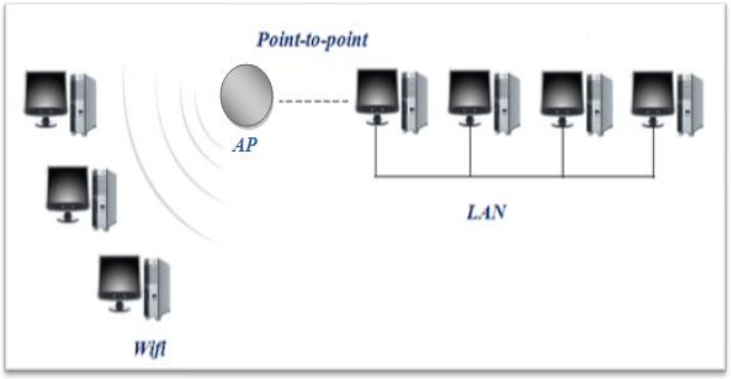

Figure 4.8 : Topologie réseau mixte test

Puis nous commençons à augmenter le nombre des stations du réseau wifi et nous ajoutons des applications entre elles.

Ensuite nous augmentons le nombre d'applications entre les stations du réseau wifi et les stations dans les autres réseaux et nous testons le nombre de clients que le point d'accès peut supporter. Si le nombre des stations trouvés dans le deuxième script est plus grand que celui du premier script alors le DLP apporte un avantage très intéressant dans le domaine du réseau, et cela prouve qu'il est un mécanisme de qualité de service très important.

IX Résultats de simulation

On a testé plusieurs paramètres de qos pour voir l'influence du mécanisme DLP sur ces dits paramètres comme suit :

- **Débit utile :**

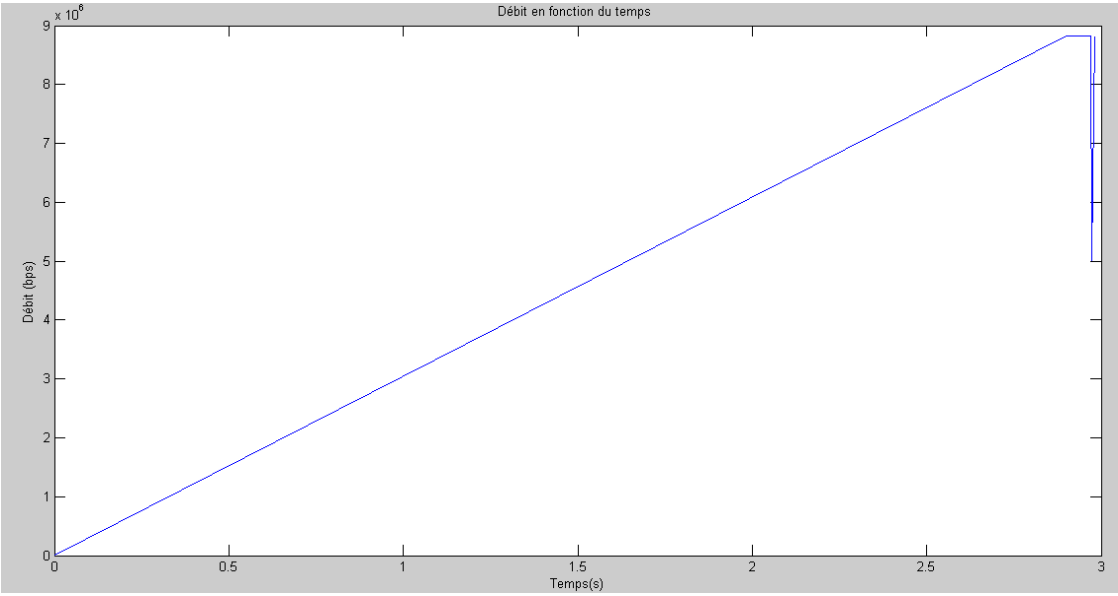

Figure 4.9 : Débit sans DLP

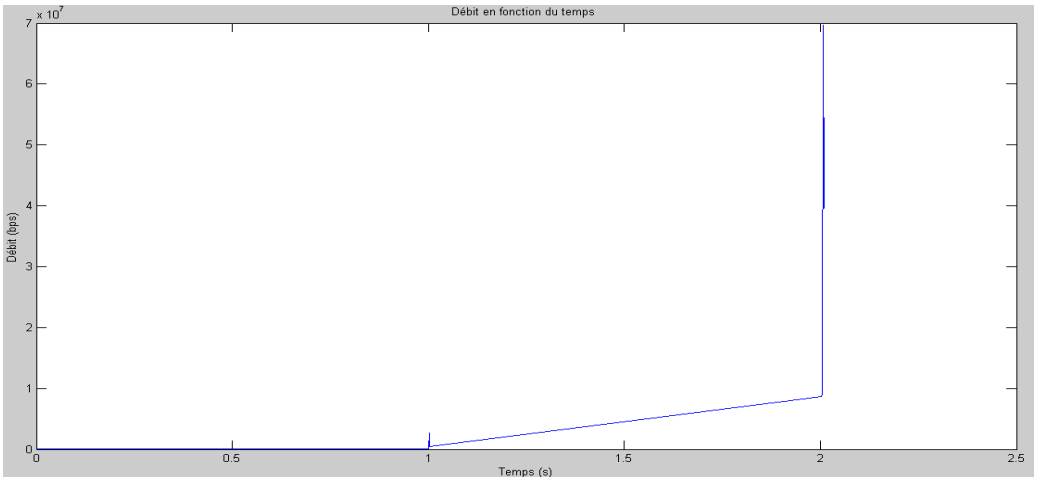

Figure 4.10 : débit avec DLP

L'analyse des deux courbes précédentes montre bien que le débit avec le mécanisme DLP est plus grand que dans le deuxième cas.

- **Délai :**

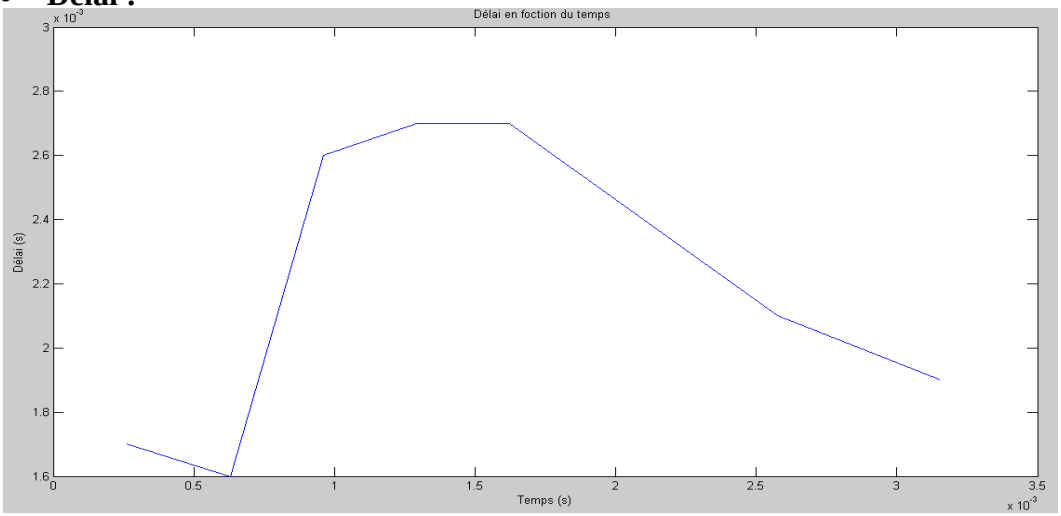

Figure 4.11: Délai sans DLP

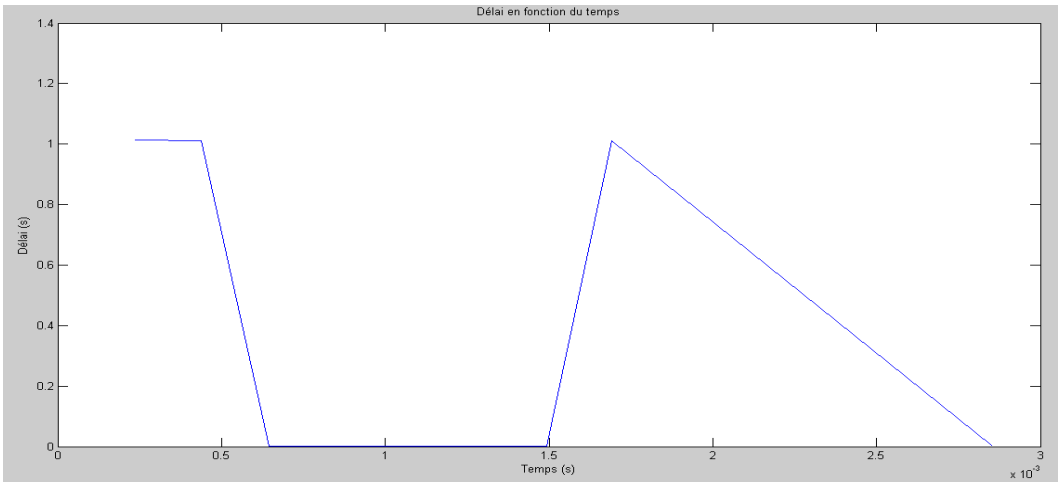

Figure 4.12: Délai avec DLP

L'analyse des deux courbes précédentes montre bien que le délai sans le mécanisme DLP est plus grand que dans le cas avec DLP.

- **Latence :**

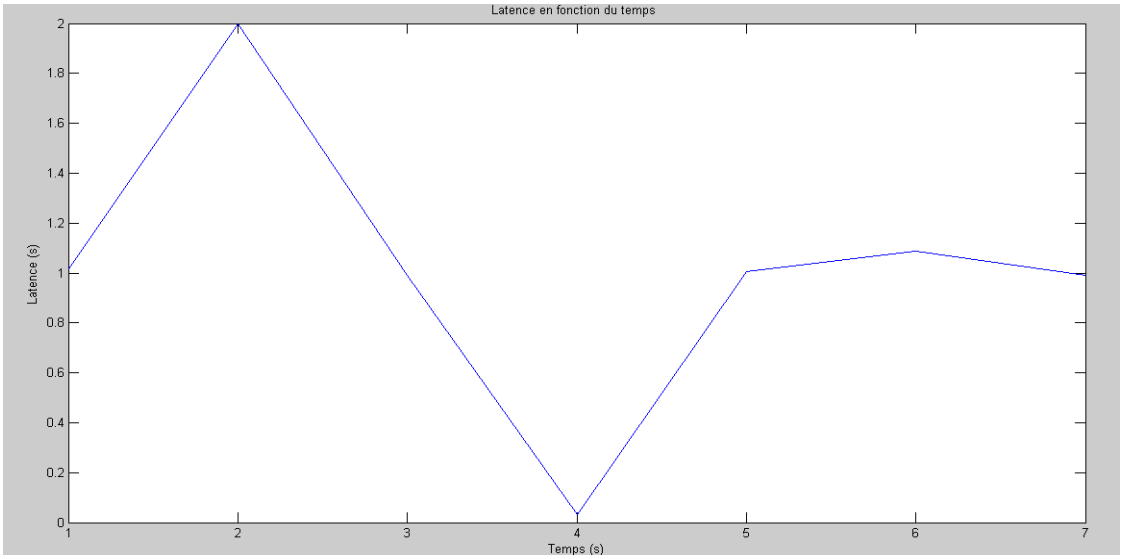

Figure 4.13 : Latence sans DLP

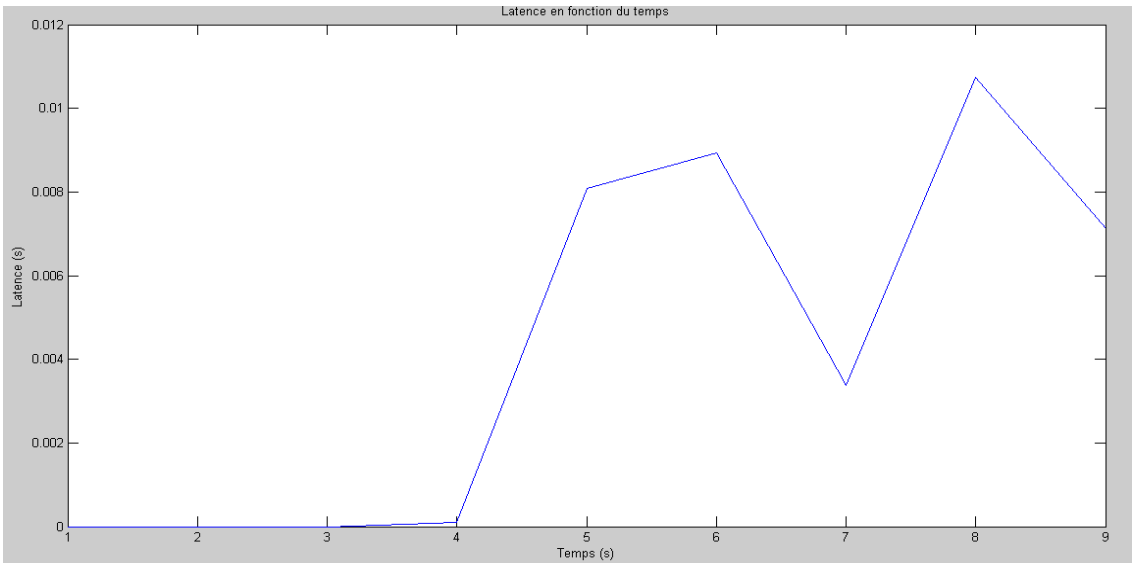

Figure 4.14: Latence avec DLP

La valeur maximal de la latence pour le premier script est 2mn par contre pour le deuxième script cette valeur ne dépasse pas 0.012mn se qui signifie que les paquets d'une même application arrive a la destination avec une différence de temps négligeable.

- **Nombre d'applications :**

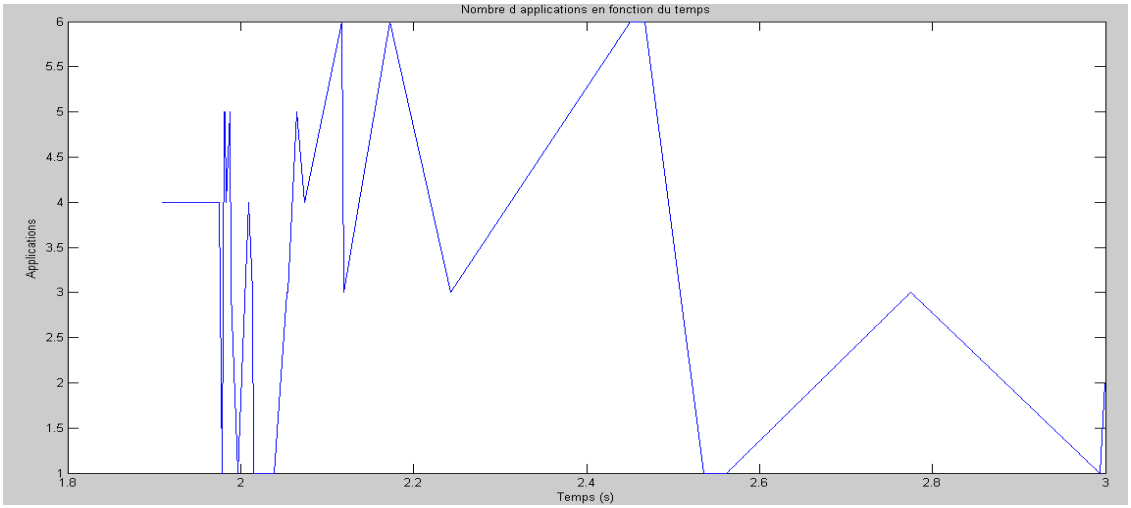

Figure 4.15: Nombres d'applications sans DLP

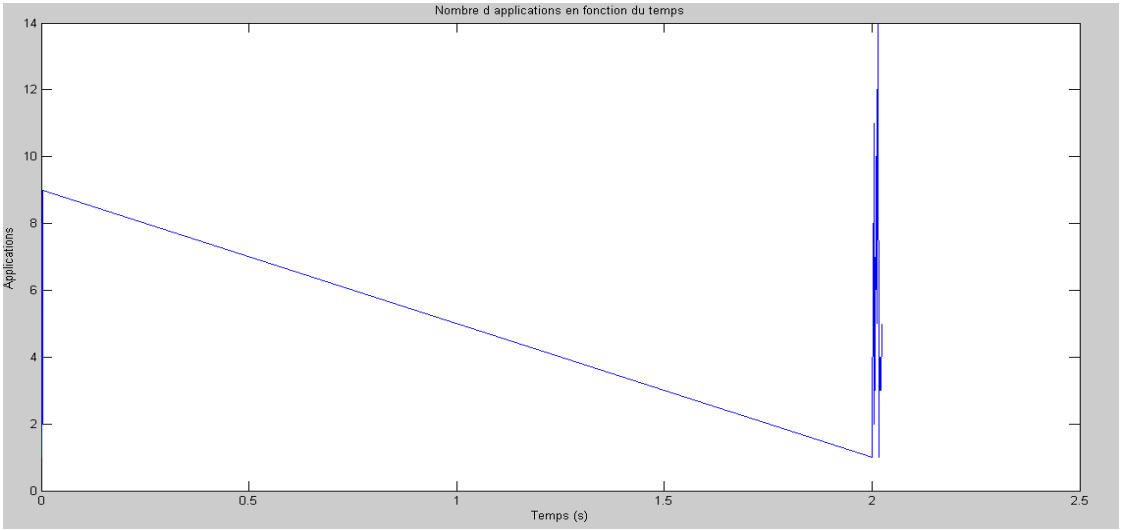

Figure 4.16 : Nombres d'applications avec DLP

Nous remarquons que le point d'accès se saturera avec six applications seulement dans le premier script par contre dans le deuxième se nombre peut arriver jusqu'à **quatorze** applications.

- **Nombre de paquets envoyés :**

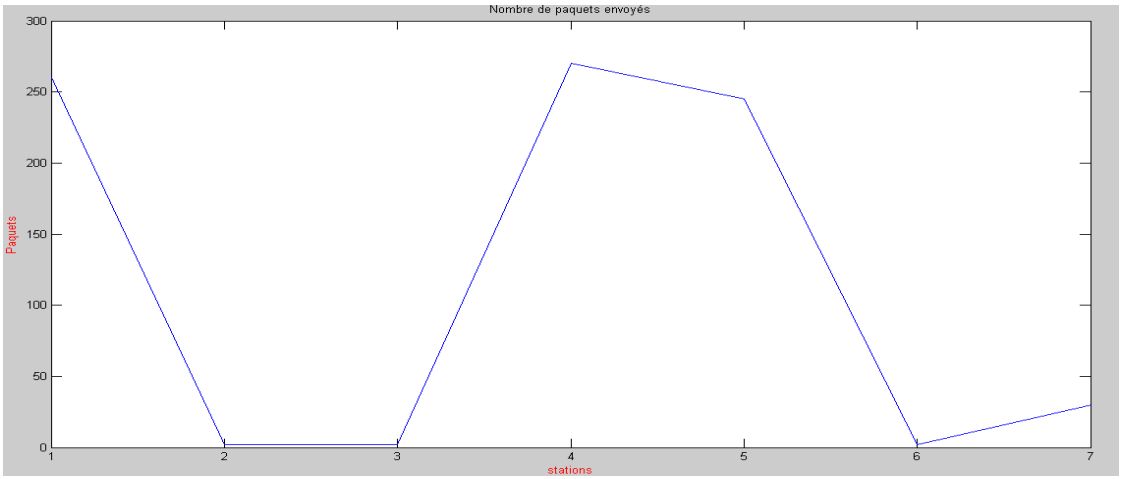

Figure 4.17: Nombre de paquets envoyés sans DLP

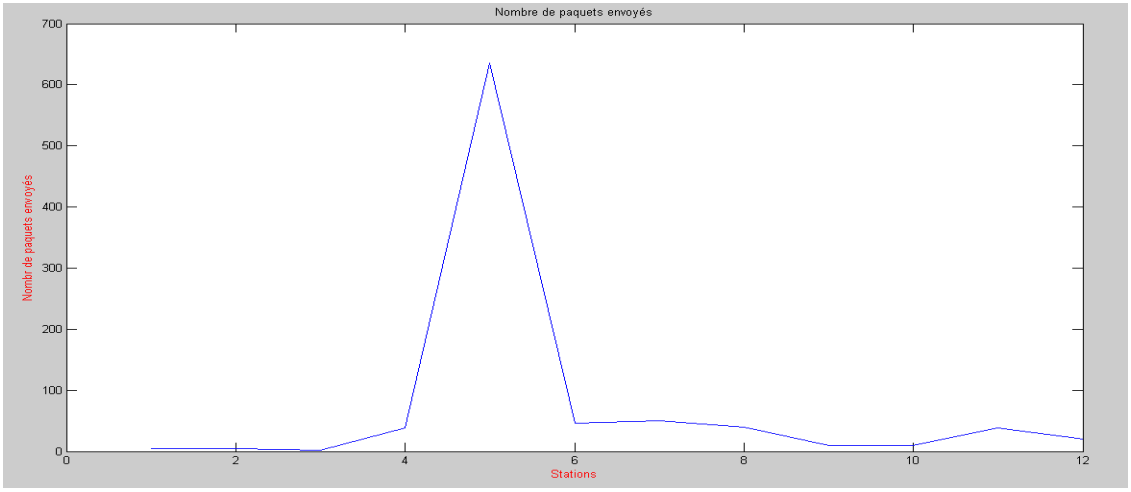

Figure 4.18: Nombre de paquets envoyés avec DLP

Le nombre de paquets maximum qui peut être envoyés dans le premier script est **250** paquets se nombre peut atteindre **600** paquets dans le deuxième script mais dans les deux cas aucun paquet n'est perdu.

X Conclusion :

Direct Link Protocol comme son nom l'indique est un protocole permettant la communication directe entre deux stations d une même cellule et se trouvant l'un à la portée de l'autre ; ce protocole permet en fait l'envoi de message dans un pseudo mode ad hoc, c a d sans passer par le point d'accès ce qui permet d augmenter le nombre de clients de ce dernier. On a réussi à tester le bon fonctionnement du DLP et démontrer l'avantage certain qu'apporte ce mécanisme dans le gain de ressources du réseau.