TABLE DES MATIERES

LISTE DES ABREVIATIONS

A

AQ : Assurance de Qualité

ACK: Acknowledgement

AES: Advanced Encryption Standard

AP: Access Point

AC: Access Caregory

ARF: Auto Rate Fallback

B

BLR: Boucle Locale Radio.

BPSK: Binary Phase Switch Keying

BSS: Basic Service Set

BE: Best Effort

BK: BackGround

BSSID: Basic Service Set Identifier

C

CRC: Cyclic Redundancy Chek.

CSMA/CA: Carrier Sense Multiple Access/Collision Avoidance.

CSMA/CD: Carrier Sense Multiple Access/Collision Detection.

CTS: Clear To Send.

CW : Contention Windows.

CBR : Constant Bit Rate.

CBQ : Class Based Queuing.

CSMA : Carrier Sense Multiple Access

D

DCF: Distributed Coordination Function

DFS: Dynamic Frequency Selection

DIFS: DCF Inter-Frame Spacing

DS: Distribution System

DSSS: Direct Sequence Spread Spectrum

DLP: Direct Link Protocol.

DRR: Decit Round-Robin.

E

EBSS: Extended Basic Service Set

EIFS: Extended Inter-Frame Spacing

ETSI: European Telecommunications Standard Institute

F

FHSS: Frequency Hopping Spread Spectrum

FCS: Frame Check Sequence

FQ: Fair Queuing.

FTP: File Transfer Protocol.

G

GMSK: Gaussian Minimum Shift Keying

GSM: Global System for Mobile communication

GPRS : General Packet Radio Service.

I

IAPP: Inter-Access Point Protocol

IP: Internet Protocol

IBSS: Independent Basic Service Set

IFS: Inter-Frame Space

IR: Infra Rouge

ISM: Industriel, Scientifique et médical.

IEEE: Institute of Electrical and Electronics Engineers

L

LAN: Local Area Network

LLC: Logical Link Control

M

MAC: Medium Access Control

MMS : Multimedia Message Service.

N

NAV: Network Allocation Vector

NS : Network Simulator

NAM : Network AniMator.

O

OFDM: Orthogonal Frequency Division Multiplexing.

OSI: Open System Interconnection.

OTCL : Object-oriented Tool Command Language.

P

PCF: Point Coordination Function.

PIFS: PCF Inter-Frame Spacing.

PTP : point à point.

PTMP : point à multipoint.

PLCP: Physical Layer Convergence Procedure.

PLW: PSDU Length Word.

PMD: Physical Media Dependant.

PN: Pseudo-random Noise.

PPM: Pulse Position Modulation.

PSK: Phase Shift Keying.

Q

QOS: Quality of Services

QPSK: Quadrature Phase Switch Keying

R

RF : Radio Frequency.

RTS: Request To Send

RED: Random Early Detection.

Rapport-Gratuit.com

S

SFD: Start Frame Delimiter.

SIFS: Short Inter-Frame Spacing.

SFQ: Stochastic Fair Queuing.

SMS: Short Message Service.

SSID: Service Set Identifier

T

TCL: Tool Command Language.

TCP: Transmission Control Protocol

TXOP: Transmission OPportunity

TID: Trafic Identifier

U

UMTS: Universal Mobile Telecommunications System.

UDP: User Datagram Protocol.

V

VCS: Virtual carrier Sense

W

WEP: Wired Equivalent Privacy.

WIFI: Wireless Fidelity.

WLAN: Wireless Local Area Network.

WPAN: Wireless Personal Area Network.

WMAN: Wireless Metropolitan Area Network.

WWAN: Wireless Wide Area Network.

WECA : Wireless Ethernet Compatibility Alliance.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

Introduction générale

Des nos jours, les consommateurs demandent des techniques de communications puissantes et adéquates pour combler leur envie de communication et d'échange d'information. Les techniques vidéo et audio rendent des services tels que le chat online et la vidéoconférence qui s'imposent sur le domaine des télécommunications. Ces différentes techniques demandent des propriétés bien définies de service, comme un délai de transmission minime, un débit utile grand, ou un faible temps d'inter arrivé.

Par conséquent, la qualité de service (*QoS*) est devenue une orientation de développement actuelle et future.

Par ailleurs, la communication sans fil demeure un nouveau centre d'intérêt. Elle s'incruste dans notre vie quotidienne au niveau social, professionnel et scientifique. Une grande variété de standards ont été développés pour satisfaire le besoin des utilisateurs en terme de débit des données. Dans ce PFE, on se propose de simuler un mécanisme de qualité de service qui est le « Direct Link », qui consiste à faire communiquer des stations directement sans passer par le point d'accès, en vue d'économiser les ressources de l'AP. Ce mécanisme n'était pas présent dans NS2, car considéré comme un mécanisme secondaire. On a l'ambition de démontrer le contraire justement. Cependant la simulation sera effectuée sous NS3, qui est prévu pour remplacer NS2, tout en étant plus simple.

Notre mémoire s'articule autour de quatre chapitres. Dans le premier, nous donnerons une présentation générale des réseaux sans fil. Dans le deuxième chapitre, nous présenterons les spécificités générales de la norme IEEE 802.11 : nous commencerons par décrire les différents modes topologiques des réseaux sans fil 802.11, puis nous nous intéresserons à l'architecture logique de la norme. Dans le troisième chapitre, nous commencerons par exposer la problématique ainsi que la dégradation des performances en matière de qualité de service (QoS). Ensuite, nous exposerons des solutions pour améliorer le comportement des réseaux congestionnés tout en s'intéressant et focalisant sur le protocole DLP pour pouvoir le tester dans le dernier chapitre. Dans Le dernier chapitre nous rendrons compte des aspects liés à l'aspect purement des simulations et à la programmation puis nous présenterons les résultats de notre simulation.

Finalement ce mémoire est clôturé par une conclusion générale résumant les idées fondamentales que nous a apportées ce travail.

I. Introduction :

L'essor des technologies sans fil offre aujourd'hui de nouvelles perspectives dans le domaine des télécommunications. L'évolution récente des moyens de la communication sans fil (communication entre machines sans besoin de liaisons filaires) a permis la manipulation de l'information à travers des unités de calculs dynamiques qui ont des caractéristiques particulières (une faible capacité de stockage, une source d'énergie autonome…) et accèdent au réseau à travers une interface de communication sans fil, d'où la naissance d'un nouvel environnement de communication appelé Environnement mobile sans fil.

Dans ce chapitre nous verrons l'intérêt des réseaux sans fil, ainsi que les différentes technologies tel que les réseaux personnels sans fil, les réseaux métropolitains, les réseaux locaux et les réseaux étendus qui seront classés selon leur zone de couverture, et après nous citons quelques avantages des réseaux sans fil.

II. Définition d'un réseau sans fil:

Un **réseau sans fil** (en anglais *wireless network*) est, comme son nom l'indique, un réseau dans lequel au moins deux terminaux (*ordinateur portable, PDA, etc.*) peuvent communiquer sans liaison filaire.

Grâce aux réseaux sans fil, un utilisateur a la possibilité de rester connecté tout en se déplaçant dans un périmètre géographique plus ou moins étendu, c'est la raison pour laquelle on entend parfois parler de "mobilité".

Les réseaux sans fil sont basés sur une liaison utilisant des ondes radio-électriques (radio et infrarouges) en lieu et place des câbles habituels. Il existe plusieurs technologies se distinguant d'une part par la fréquence d'émission utilisée ainsi que le débit et la portée des transmissions.

Les réseaux sans fil permettent de relier très facilement des équipements distants d'une dizaine de mètres à quelques kilomètres. De plus l'installation de tels réseaux ne demande pas de lourds aménagements des infrastructures existantes comme c'est le cas avec les réseaux filaires (creusement de tranchées pour acheminer les câbles, équipements des bâtiments en câblage, goulottes et connecteurs), ce qui a valu un développement rapide de ce type de technologies.

En contrepartie se pose le problème de la réglementation relative aux transmissions radio-électriques. En effet, les transmissions radio-électriques servent pour un grand nombre d'applications (militaires, scientifiques, amateurs, ...), mais sont sensibles aux

interférences, c'est la raison pour laquelle une réglementation est nécessaire dans chaque pays afin de définir les plages de fréquence et les puissances auxquelles il est possible d'émettre pour chaque catégorie d'utilisation.

De plus les ondes hertziennes sont difficiles à confiner dans une surface géographique restreinte, il est donc facile pour un pirate d'écouter le réseau si les informations circulent en clair (c'est le cas par défaut). Il est donc nécessaire de mettre en place les dispositions nécessaires de telle manière à assurer une confidentialité des données circulant sur les réseaux sans fil. **[1]**

III. Intérêt du «sans fil» :

Un réseau sans fil peut servir plusieurs buts distincts :

- Utilisation croissante des terminaux portables en milieu industriel et logistique ;
- Besoin d'un accès permanent des populations nomades au système d'information de l'entreprise ;
- Pour transmettre :
    - Des messages courts ;
    - bips, numériques, alphanumériques ;
    - La voix ;
    - Des données informatiques ;
        - fax, fichiers, textes, images.
- Réaliser des installations temporaires ;
- Mettre en place des réseaux en un temps très court ;
- Eviter le câblage de locaux, de liaisons inter-bâtiments ;
- Créer une infrastructure dans des bâtiments classés ;
- Maturité des technologies sans fil:
    - maîtrise de la téléphonie cellulaire sur une large échelle ;
    - numérisation des communications, miniaturisation des interfaces ;
- Assouplissement des règlementations :
    - disponibilité de nouvelles fréquences ;
- Mise en place d'une standardisation européenne :
    - au niveau des infrastructures (norme ETS300/328) ;
    - pour l'attribution des bandes de fréquences (bande des 2.4 Ghz) ;

- Normalisation IEEE802.11 ;
- Technologies :
    - spectre radio ;
    - infrarouge ;
    - optique (laser). **[2]**

IV. Les technologies sans fil:

Les technologies dites « sans fil », la norme 802.11 en particulier, facilitent et réduisent le coût de connexion pour les réseaux de grande taille. Avec peu de matériel et un peu d'organisation, de grandes quantités d'informations peuvent maintenant circuler sur plusieurs centaines de mètres, sans avoir recours à une compagnie de téléphone ou de câblage.

Ces technologies peuvent être classées en quatre parties :

- Les réseaux personnels sans fil : WPAN.
- Les réseaux locaux sans fil : WLAN.
- Les réseaux métropolitains sans fil : WMAN.
- Les larges réseaux sans fil : WWAN. **[3]**

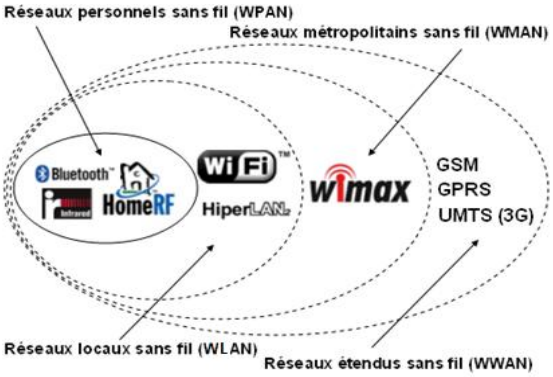

Figure 1-1: Catégories des réseaux sans fil [4]

IV.1 Réseaux personnels sans fil (WPAN) :

Le *réseau personnel sans fil* (appelé également *réseau individuel sans fil* ou *réseau domestique sans fil* et noté **WPAN**) concerne les réseaux sans fil d'une faible portée : de l'ordre de quelques dizaines de mètres. Ce type de réseau sert généralement à relier des périphériques (*imprimante, téléphone portable, appareils domestiques, ...)* ou un assistant personnel (*PDA*) à un *ordinateur* sans liaison filaire ou bien à permettre la liaison sans fil entre deux machines très peu distantes. Il existe plusieurs technologies utilisées pour les WPAN :

IV.1.1 Bluetooth :

C'est la principale technologie *WPAN*, lancée par Ericsson en 1994, proposant un débit théorique de **1 Mbps** pour une portée maximale d'une trentaine de mètres. Bluetooth, connue aussi sous le nom ***IEEE 802.15.1,*** possède l'avantage d'être très peu gourmande en énergie, ce qui la rend particulièrement adaptée à une utilisation au sein de petits périphériques.

Figure 1-2 : Logo de Bluetooth

IV.1.2 HomeRF :

Lancée en 1998 par le **HomeRF Working Group** (formé notamment par les constructeurs Compaq, HP, Intel, Siemens, Motorola et Microsoft) propose un débit théorique de **10 Mbps** avec une portée d'environ **50 à 100 mètres** sans amplificateur. La norme HomeRF soutenue notamment par Intel, a été abandonnée en Janvier 2003, notamment car les fondeurs de processeurs misent désormais sur les technologies Wi-Fi embarquée (via la technologie *Centrino*, embarquant au sein d'un même composant un microprocesseur et un adaptateur Wi-Fi).

Figure 1-3 : Logo de HomeRF

IV.1.3 ZigBee :

(Aussi connue sous le nom *IEEE 802.15.4*) permet d'obtenir des liaisons sans fil à très bas prix et avec une très faible consommation d'énergie, ce qui la rend particulièrement adaptée pour être directement intégrée dans de petits appareils électroniques (*appareils électroménagers, hifi, jouets, ...).* La technologie Zigbee, opérant sur la bande de fréquences des **2,4 GHz** et sur **16 canaux**, permet d'obtenir des débits pouvant atteindre **250 Kb/s** avec une portée maximale de **100 mètres** environ.

Figure 1-4 : Logo de *ZigBee*

IV.1.4 *Infrarouges :*

Permettent de créer des liaisons sans fil de quelques mètres avec des débits pouvant monter à quelques mégabits par seconde. Cette technologie est largement utilisée pour la domotique (télécommandes) mais souffre toutefois des perturbations dues aux interférences lumineuses. **[5]**

IV.2 Réseaux locaux sans fil (WLAN) :

Le *réseau local sans fil* (noté **WLAN**) est un réseau permettant de couvrir l'équivalent d'un réseau local d'entreprise, soit une portée d'environ une centaine de mètres. Il permet de relier entre-eux les terminaux présents dans la zone de couverture. Il existe plusieurs technologies concurrentes :

IV.2.1 Wifi : (ou IEEE 802.11) :

Est un standard international décrivant les caractéristiques d'un réseau local sans fil WLAN, elle est soutenu par l'alliance **WECA**.

Figure 1-5 : Logo de Wi-Fi [6]

Grâce au Wi-Fi, il est possible de créer des réseaux locaux sans fils à haut débit pour peu que l'ordinateur à connecter ne soit pas trop distante par rapport au point d'accès. Dans la pratique, le WiFi permet de relier des ordinateurs portables, des ordinateurs de bureau, des assistants personnels *(PDA*) ou tout type de périphérique à une liaison haut débit (**11 Mbps** ou supérieur) sur un rayon de plusieurs dizaines de mètres en intérieur (généralement entre une vingtaine et une cinquantaine de mètres) à plusieurs centaines de mètres en environnement ouvert. **[7]**

IV.3 Réseaux métropolitains sans fil (WMAN) :

Le *réseau métropolitain sans fil* (**WMAN**) est connu sous le nom de **BLR**. Les WMAN sont basés sur la norme *IEEE 802.16.* La boucle locale radio offre un débit utile de **1 à 10 Mbit/s** pour une portée de **4 à 10 kilomètres**, ce qui destine principalement cette technologie aux opérateurs de télécommunication.

IV.3.1 WiMAX :

C'est La norme de réseau métropolitain sans fil la plus connue permettant d'obtenir des débits de l'ordre de 70 Mbit/s sur un rayon de plusieurs kilomètres.

Elle permet de fournir un accès internet rapide à certaine zones rurales qu'il couterait trop cher d'équipé en ADSL classique. **WiMAX** utilise des bandes de très hautes fréquences, situées entre **2** et **66 GHz.**

C'est une technologie de réseau sans fil fixe et non mobile. Elle nécessite que les antennes émettrices et réceptrices soient situés l'une en face de l'autre pour que les transmissions passent.

Connue sous le nom officiel *802.16*, cette technologie est très utile pour éviter les couteuses liaisons câblés qui étaient jusques là nécessaires pour apporter l'internet à haut débit dans les régions moins peuplées. **WiMAX** peut être utilisé en complément de Wi-Fi pour relier deux réseaux trop éloignés l'un de l'autre, par exemple deux bâtiments d'une même entreprise. **[8]**

Figure 1-6 : Logo de WiMAX

IV.4 Réseaux étendus sans fil (WWAN) :

Le réseau étendu sans fil (**WWAN**) est également connu sous le nom de réseau cellulaire mobile. Il s'agit des réseaux sans fil les plus répandus puisque tous les téléphones mobiles sont connectés à un réseau étendu sans fil. Les principales technologies sont les suivantes : **[9]**

IV.4.1 *GSM :*

Le réseau **GSM** constitue au début du 21ème siècle le standard de téléphonie mobile le plus utilisé en Europe. Il s'agit d'un standard de téléphonie dit « de seconde génération » (**2G**) car, contrairement à la première génération de téléphones portables, les communications fonctionnent selon un mode entièrement numérique.

La norme **GSM** autorise un débit maximal de **9,6 kbps**, ce qui permet de transmettre la voix ainsi que des données numériques de faible volume, par exemple des messages textes (**SMS**) ou des messages multimédias (**MMS**). **[10]**

IV.4.2 GPRS :

Le standard **GPRS** est une évolution de la norme GSM, ce qui lui vaut parfois l'appellation **GSM++** (ou **GMS 2+).** Etant donné qu'il s'agit d'une norme de téléphonie de seconde génération permettant de faire la transition vers la troisième génération (3G). Le GPRS permet d'étendre l'architecture du standard GSM, afin d'autoriser le transfert de données par paquets, avec des débits théoriques maximums de l'ordre de **171,2 kbit/s** (en pratique jusqu'à **114 kbit/s**). Grâce au mode de transfert par paquets, les transmissions de données n'utilisent le réseau que lorsque c'est nécessaire. Le standard GPRS permet donc de facturer l'utilisateur au volume échangé plutôt qu'à la durée de connexion, ce qui signifie notamment qu'il peut rester connecté sans surcoût.

Rapport-Gratuit.com

Ainsi, le standard GPRS utilise l'architecture du réseau GSM pour le transport de la voix, et propose d'accéder à des réseaux de données (notamment internet) utilisant le **protocole IP** ou le protocole **X.25.**

Le GPRS permet de nouveaux usages que ne permettait pas la norme GSM, généralement catégorisés par les classes de services suivants :

- Services **PTP**, c'est-à-dire la capacité à se connecter en mode client-serveur à une machine d'un réseau IP,
- Services **PTMP**, c'est-à-dire l'aptitude à envoyer un paquet à un groupe de destinataires (***Multicast*)**.
- Services de messages courts (**SMS**). **[11]**

V. Les avantages et les inconvénients des réseaux sans fil :

Voici les principaux avantages et inconvénients à déployer un réseau sans fil **:**

V.1 Les avantages du réseau sans fil :

V.1.1 Financier :

Le réseau sans fil permet d'éviter l'obligation d'un câblage coûteux qui peut se révéler rapidement obsolète ou inutile en cas de déménagements de locaux.

V.1.2 Facilité et flexibilité :

Dans le contexte d'un réseau temporaire, pour des formations, des expositions ou autre chantiers, pour couvrir des zones difficiles d'accès aux câbles, et relier des bâtiments distants.

V.1.3 Mobilité :

Par exemple, tous les participants d'une réunion sont automatiquement interconnectés sans avoir besoin de perdre du temps en début de réunion pour relier chaque personne.

V.2 Les inconvénients du réseau sans fil :

V.2.1 Qualité et continuité du signal :

Ces notions ne sont pas garanties du fait des problèmes pouvant venir des interférences du matériel et de l'environnement.

V.2.2 Problèmes de sécurité :

Les données échangées sont transmises par voie aérienne et couvrent de grandes distances dans le cadre d'une liaison hertzienne donc il est possible en théorie de récupérer ses données même si celles-ci sont cryptées et confidentielles. **[12]**

VI. Conclusion :

Ces dernières années, les réseaux sans fil ont connues un essor considérable et ceci revient aux multiples avantages qu'elles offrent (mobilité, fiabilité, etc.) mais grâce à l'extrême d'usage libre que le wifi offre aux utilisateurs « nomades » en assurant une continuité des services à la fois performante et économique via des terminaux adaptés, fiables et relativement peu coûteux (PC portable, PDA, téléphone mobile,…),le Wifi est arrivée à surpassé ses autres concurrents.

Donc le WIFI est devenu un moyen dominant permettant de fournir une architecture de réseaux locaux sans fils.

Dans le chapitre qui suit nous allons se focaliser sur l'étude de la norme 802.11 du standard IEEE.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

I. Introduction :

En ce début du 21eme siècle, les réseaux locaux informatiques connaissent deux évolutions importantes. D'une part, l'utilisation courante du réseau local chez les particuliers, due en grande partie à internet, et d'autre part, l'arrivée en masse des ordinateurs et autres matériels mobiles. Pour cela il faut trouver une technologie permettant de simplifier le câblage du réseau chez un particulier et de préserver la mobilité des produits portables. Un seul principe permet de concilier les deux, le sans fil. [13]

IEEE 802.11 est un standard de réseau sans fil local proposé par l'organisme de standardisation Américain IEEE. La technologie 802.11 est généralement considérée comme la version sans fil de 802.3 (Ethernet). [14]

L'objectif de ce chapitre est de présenter en détail le standard 802.11 qui est le plus utilisé dans les réseaux locaux sans fil. Pour cela, nous commencerons dans une première partie par décrire les topologies suivant lesquels les WLAN 802.11 fonctionnent. Ensuite, nous présenterons les différentes versions du standard et les caractéristiques liées à l'architecture logique de la norme (couche physique et couche MAC).

II. Le standard IEEE 802.11 :

II.1 Généralités :

La première version de la norme IEEE 802.11 est définie en 1997. Des transmissions infrarouges étaient envisagées, les versions les plus récentes du standard sur la base desquelles sont construites l'essentiel des cartes d'interface commercialisées, s'adressent principalement à des transmissions radiofréquences.

Pour définir cette norme, les concepteurs ont pris en considération les points suivants :

- Robustesse et simplicité de la technologie contre les défauts de communication, de pouvoir transmettre dans les meilleures conditions, tenant compte des considérations que le canal de transmission, en l'occurrence l'air, n'est pas aussi fiable que le câble, et qu'il est plus difficile à gérer. Ces caractéristiques ont été vérifiées par l'utilisation d'une approche distribuée du protocole de la couche MAC.
- Utilisation du WLAN mondialement. C'est-à-dire le respect des différentes règles en usage dans les différents pays du monde.

- Totale compatibilité avec les anciens produits et les produits actuels qui composent les réseaux LAN. C'est-à-dire que le passage du WLAN au LAN et vice-versa devra être transparent à l'utilisateur.
- Une sécurité acceptable pour le passage de l'information dans l'air. (WEP).

Cette technologie très intéressante pourra prendre la relève des LAN au sein des entreprises, mais seulement le principal problème vient de la qualité de transmission, puisque le problème de capacité tend de plus en plus à être réduit, par l'augmentation des débits de transmission. Ce problème vient du fait que le canal de transport du WLAN n'est autre que l'air et il va être étudié d'une manière détaillé dans le chapitre suivant. [16], [19]

II.2 La famille IEEE 802 et les standards 802.11:

Le 802.11 est issu de la famille 802, qui est une série de spécifications pour les réseaux locaux. La figure montre la relation entre les différents composants de la famille 802 et leurs emplacements dans le modèle OSI.

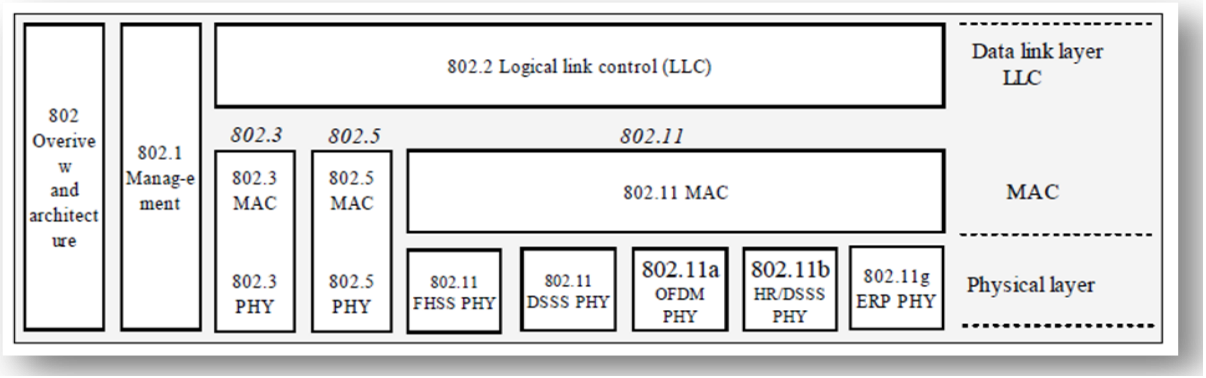

Figure 2.1 : la famille IEEE 802.11

Comme les spécifications 802, le standards IEEE 802.11 couvre les deux couches inferieures du modèle OSI : la couche liaison et la couche physique. La couche MAC définit un ensemble de règles permettant d'accéder au médium et d'envoyer des données, les détails de la réception et de la transmission, sont traiter au niveau de la couche physique. [13]

Actuellement au sein du 802.11 plusieurs groupes de travail ont été crées afin d'améliorer ou de proposer des nouveaux mécanismes régissant divers aspect. Des révisions donc ont été apportées à la norme originale (avec un débit de 1 ou 2 Mbps) afin d'optimiser le débit (c'est le cas des normes 802.11 physiques à savoir les normes 802.11a, 802.11b, 802.11g) ou bien préciser des éléments afin d'assurer une meilleure sécurité ou une meilleure interopérabilité. [17]

Les différentes révisions de la norme 802.11 sont citées ici : [14], [18]

- **802.11 (norme initiale) :** Dans sa version initiale de 1997, 802.11 proposait trois couches physiques : Radio a étalement de spectre par utilisation de séquences directes (DSSS3), débit bande de base 1 Mbits/s et 2 Mbits/s, Radio a étalement de spectre par utilisation de sauts de fréquences (FHSS3) a 1,6 Mbits/s, Infrarouge, 1 ou 2 Mbits/s.
- **802.11 a :** propose 8 canaux dans la bande des 5 GHz. Cette proposition permet d'atteindre un débit bande de base de 54 Mbits/s sur une portée d'une vingtaine de mètres environ.
- **802.11 b :** propose une amélioration de la norme initiale en introduisant la modulation CCK3 dans la bande des 2,4 GHz. Deux nouveaux débits sont alors disponibles : 5,5 Mbits/s et 11 Mbits/s sur une portée de quelques dizaines de mètres environ. Ratifiée en septembre 1999, 802.11b est l'amendement de 802.11 qui a donné sa popularité au Wifi. Bien que 802.11b soit encore largement utilisé, il est maintenant supplanté par 802.11g.
- **802.11 c :** propose une modification de la norme 802.1d existante pour les réseaux filaires afin de la transposer a 802.11. Elle permet une normalisation de l'interconnexion de niveau 2 (pont) entre un réseau filaire et un réseau Wifi.
- **802.11 d :** propose un protocole d'échange d'informations sur les fréquences et les puissances d'´emission en vue d'une utilisation dans chaque région du monde, quelque soit le pays d'origine du matériel.
- **802.11 e :** propose des outils de Qualité de Service. Les travaux spécifiques de ce groupe de travail seront détaillés et cette norme sera étudiée plus loin dans le chapitre 3.
- **802.11 f :** est une recommandation qui propose une extension pour la communication entre points d'accès compatibles 802.11 par le protocole IAPP en introduisant des capacités de changement de cellules et d'équilibrage des charges (load-balancing).
- **802.11 g :** constitue une amélioration directe de 802.11b en proposant un débit bande de base de 54 Mbits/s sur la bande des 2,4 GHz. Ce gain en débit est réalisé en reprenant le concept de l'´etalement de spectre par OFDM utilisé dans 802.11a. Toutefois, 802.11g garde une compatibilité avec 802.11b, ce qui

signifie que des matériels conformes à la norme 802.11g peuvent fonctionner en 802.11b.

- **802.11 h :** propose des améliorations pour pallier au futur problème de la sur-utilisation des fréquences dédiées à 802.11. Ce groupe de travail propose d'une part une possibilité de sélection dynamique de fréquence appelée DFS, qui permet de choisir le canal le moins perturbé, et d'autre part le contrôle de puissance TP pour Transmit Power Control, qui permet a l'émetteur de réduire sa puissance d'´emission au minimum nécessaire.
- **802.11 i :** met en place les mécanismes afin de garantir la sécurité. Cette norme définit des techniques de chiffrage telles que l'AES.
- **802.11 n :** son but est d'étendre le standard 802.11 pour atteindre un débit de 540 Mbit/s tout en assurant une rétrocompatibilité avec les trois précédents amendements (a, b et g). Sa porté est d'une centaine de mètre.il utilise les deux bandes 2.4 et 5GHz.
- **802.11 x :** sécurisation de divers médias y compris le lien sans fil par le biais de mécanismes d'authentification fort et de serveur RADIUS avec une distribution dynamique des clés.

II.3 Topologies: [14], [16]

Le réseau sans fil utilisant la norme 802.11 peut être déployé de deux manières différentes : Avec infrastructure ou sans infrastructure (mode Ad Hoc).

II.3.1 Réseaux WLAN avec Infrastructure :

Le réseau à infrastructure comprend des points d'accès ou Access Point qui gèrent l'ensemble des communications dans une même zone géographique sous la forme de cellule. Ce mode de gestion géographique ressemble un peu au modèle GSM ou UMTS. D'ailleurs il fonctionne de façon presque similaire, car les stations munies de carte WLAN peuvent se déplacer dans la zone de couverture de l'AP et effectuer un roaming entre les différents AP si la topologie le permet (chevauchement des cellules). Il faut remarquer que chaque AP possède une connexion LAN, ou un autre type de connexion lui assurant la connexion avec le réseau fixe.

Le réseau est alors formé de plusieurs BSS qui forment ensemble un unique EBSS.

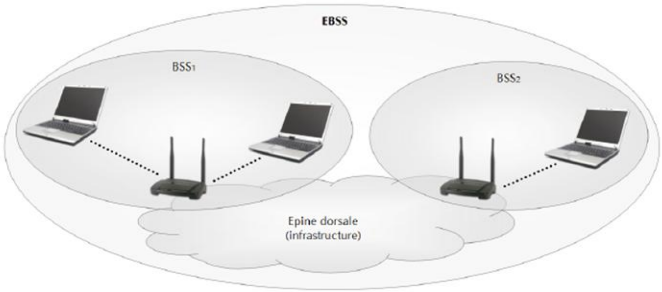

Figure 2.2 : Réseau WLAN avec infrastructure

II.3.2 Réseau WLAN Ad Hoc :

Un réseau Ad Hoc ou encore IBSS (Independent Basic Service Set) est un ensemble de stations possédant une carte WLAN sans la présence d'un AP. Contrairement au réseau à infrastructure, les stations dans un réseau Ad Hoc communiquent directement entre elles.

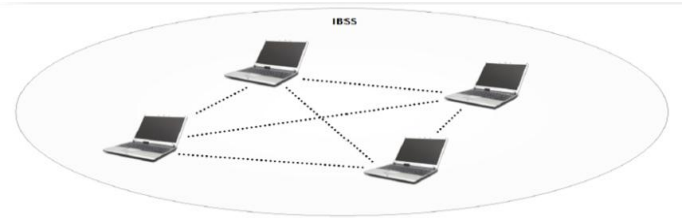

Figure 2.3: Réseau WLAN Ad Hoc

II.4 Architecture de la norme IEEE 802.11: [15], [22]

La norme IEEE 802.11 définit les deux premières couches (basses) du modèle OSI, à savoir la couche physique et la couche liaison de données. Elle introduit des modifications sur la couche basse du niveau lien (donc niveau MAC) et sur le niveau physique avec le support de plusieurs méthodes d'accès radio (donc la définition de plusieurs couches physiques). Il est à noter que la nouvelle couche MAC est commune à toutes les couches physiques. La figure2.4 illustre l'architecture en couches de la norme IEEE 802.11.

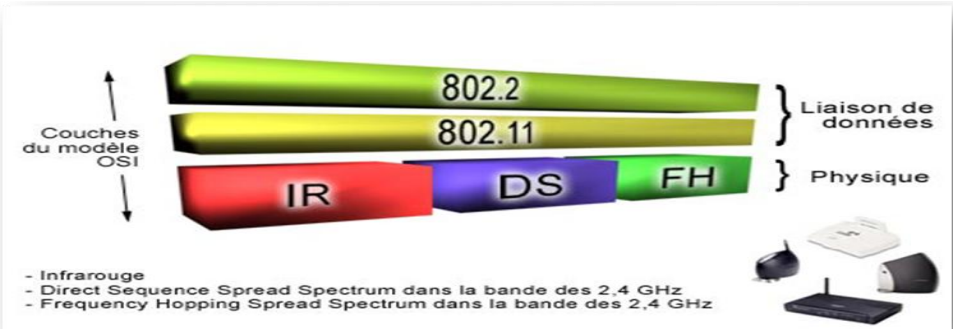

Figure 2.4 : Description des couches IEEE 802.11

La couche physique définit la modulation des ondes radioélectriques et les caractéristiques de la signalisation pour la transmission de données, tandis que la couche liaison de données définit l'interface entre le bus de la machine et la couche physique, notamment une méthode d'accès proche de celle utilisée dans le standard Ethernet et les règles de communication entre les différentes stations. [20]

II.4.1 La couche physique :

La norme IEEE 802.11 définit deux sous-couches physiques :

- PMD (Physical Media Dependant) : gère l'encodage des données et la modulation.
- PLCP (Physical Layer Convergence Procedure): s'occupe de l'écoute du support et est directement reliée à la couche MAC pour lui signifier que le support de transmission est libre.

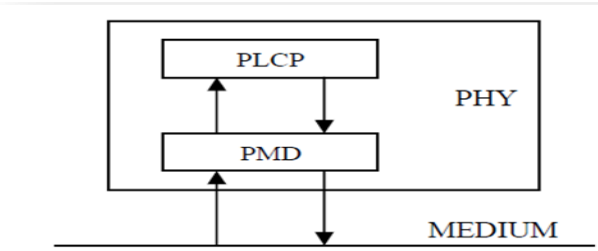

Figure 2.5 : Les deux sous couches physique du standard 802.11

Le standard 802.11 d'origine a défini trois couches physiques de base, FHSS, DSSS, IR, auxquelles ont été rajoutées trois nouvelles couches physiques Wifi (avec deux variantes au sein de la solution 802.11b) et Wi-Fi5 (802.11a/g).la figure suivante illustre ça :

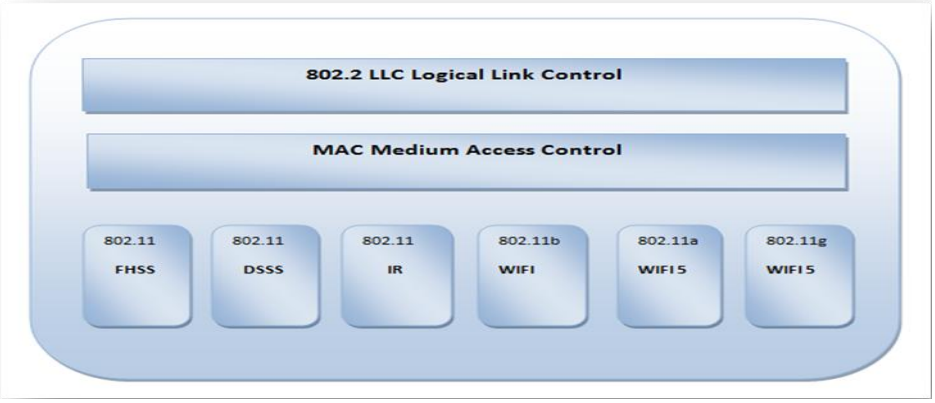

Figure 2.6 : Les couches physique du standard 802.11

i. Les couches physiques de base :

- **FHSS (Frequency Hopping Spread Spectrum):** [15] , [20], [22], [23]

La technique **FHSS** (Frequency Hopping Spread Spectrum, en français étalement de spectre par saut de fréquence ou étalement de spectre par évasion de fréquence) consiste à découper la large bande de fréquence en un minimum de 75 canaux (hops ou sauts d'une largeur de 1MHz), puis de transmettre en utilisant une combinaison de canaux connue de toutes les stations de la cellule. Dans la norme 802.11, la bande de fréquence 2.4 - 2.4835 GHz permet de créer 79 canaux de 1 MHz.

La transmission est ainsi réalisée en émettant successivement sur un canal puis sur un autre pendant une courte période de temps (d'environ 400 ms), ce qui permet à un instant donné de transmettre un signal plus facilement reconnaissable sur une fréquence donnée. L'émetteur et le récepteur s'accordent sur un schéma de saut, et les données sont envoyées sur une séquence de sous-canaux. Chaque conversation sur le réseau 802.11 s'effectue suivant un schéma de saut différent, et ces schémas sont définis de manière à minimiser le risque que deux expéditeurs utilisent simultanément le même sous-canal.

L'étalement de spectre par saut de fréquence a originalement été conçu dans un but militaire afin d'empêcher l'écoute des transmissions radio. En effet, une station ne connaissant pas la combinaison de fréquence à utiliser ne pouvait pas écouter la communication car il lui était impossible dans le temps imparti de localiser la fréquence sur laquelle le signal était émis puis de chercher la nouvelle fréquence. Aujourd'hui les réseaux locaux utilisant cette technologie sont standards ce qui signifie que la séquence de fréquences utilisées est connue de tous, l'étalement de spectre par saut de fréquence n'assure donc plus cette fonction de sécurisation des échanges.

FHSS est désormais utilisé dans le standard 802.11 de telle manière à réduire les interférences entre les transmissions des diverses stations d'une cellule.

Les techniques FHSS simplifient -- relativement -- la conception des liaisons radio, mais elles sont limitées à un débit de 2 Mbps, cette limitation résultant essentiellement des réglementations de l'ETSI qui restreignent la bande passante des sous-canaux à 1 MHz. Ces contraintes forcent les systèmes FHSS à s'étaler sur l'ensemble de la bande des 2,4 GHz, ce qui signifie que les sauts doivent être fréquents et représentent en fin de compte une charge importante.

En mode FHSS les données sont émises au moyen d'une modulation GMSK.

L'un des avantages du FHSS est qu'il permet, théoriquement, de faire fonctionner simultanément 26 réseaux 802.11 FHSS (correspondant aux 26 séquences) dans une même zone, chaque réseau utilisant une des séquences prédéfinies.

Un autre avantage du FHSS est sa résistance face aux interférences, comme le système saute toutes les 300 ms d'un canal à un autre sur la totalité de la bande, si des interférences surviennent sur une partie de la bande ISM (un ou plusieurs canaux), cela n'engendre pas de trop importantes pertes de performances.

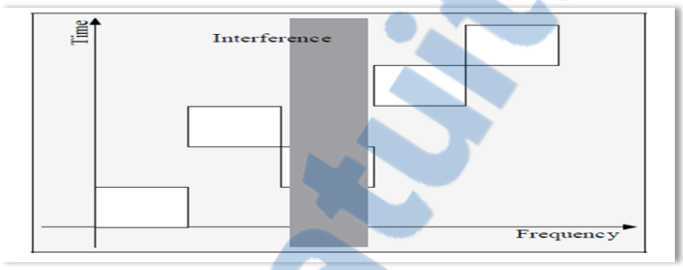

Figure 2.7 : Changement de fréquence dans FHSS

Une trame au niveau physique est composée de trois parties. Elle débute par un préambule, suivi d'un entête et terminée par la partie donnée :

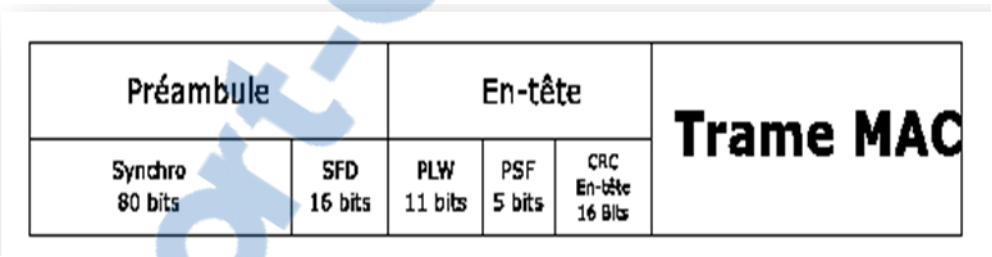

Figure 2.8 : Structure de la trame 802.11 au niveau physique pour le FHSS

Chaque champ de chaque partie possède un rôle spécifique :

- **Le préambule :**
  - La synchro est une séquence de synchronisation qui est composée d'une suite de 80 bits constitués en alternance de 0 et de 1. Elle permet à la couche physique de détecter la réception d'un signal. Elle permet accessoirement aussi, de choisir la meilleure antenne de réception si le choix existe.
  - Le Start Frame Delimiter (SFD) est l'identificateur de trame. Il est constitué par la suite de bits suivants : 0001100101101101.
- **L'entête :**
  - Le PSDU Length Word (PLW) est un paramètre passé par la couche MAC qui indique la longueur de la trame. C'est donc la longueur de la partie de donnée dans cette trame.

- Le PSF est un champ sur 5 bits qui permet de définir la vitesse de transmission. Le premier bit est toujours à 0. Les bits 1, 2 et 3 sont réservés et définis par défaut à zéro. Le 4ème et dernier bit, indique la vitesse de transmission. A 1Mb/s s'il est à 0 et à 2Mb/s s'il est à 1.
- Le CRC de l'entête est le champ de contrôle d'erreur de l'entête, composé de 16bits.

▪ **La partie donnée :** La Trame MAC contient les données relatives à la couche MAC.

- **DSSS (Direct Sequence Spread Spectrum):** [13], [15], [22], [20], [23]

Dans le but de lutter contre les interférences importantes mais n'affectant que des plages de fréquences assez étroites, il existe la technique de l'étalement de spectre.

Comme le FHSS, le DSSS divise la bande ISM en sous bandes. Cependant la division se fait ici en 14 canaux de 20 MHz chacun. La transmission ne se fait que sur un canal donné. La largeur de la bande ISM étant égale à 83.5 MHz, il est impossible d'y placer 14 canaux adjacents de 20 MHz. Les canaux se recouvrent donc, comme illustré à la figure suivante.

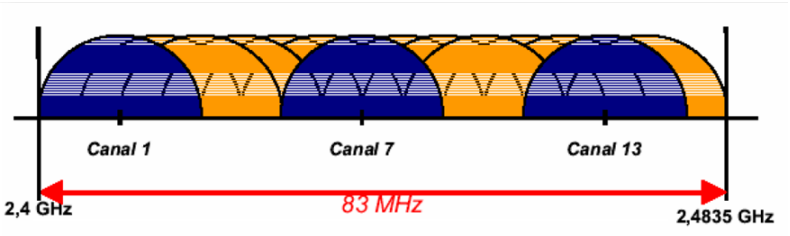

Figure 2.9 : Décomposition de la bande ISM en sous canaux

Comme le montre le tableau suivant, les fréquences centrales de chaque sous-canal sont espacées de 5 MHz.

Canal	Fréquence centrale (GHz)	Canal	Fréquence centrale (GHz)
1	2.412	8	2.447
2	2.417	9	2.452
3	2.422	10	2.457
4	2.427	11	2.462
5	2.432	12	2.467
6	2.437	13	2.472
7	2.442	14	2.477

Tableau 2.1: Fréquences centrales des sous canaux du mode DSSS

Comme la transmission ne se fait que sur un canal, les systèmes DSSS sont plus sensibles aux interférences que les systèmes FHSS, qui utilisent toute la largeur de bande.

L'utilisation d'un seul canal pour la transmission est un inconvénient si différents réseaux 802.11 DSSS se superposent.

Lorsqu'un canal est sélectionné, le spectre du signal occupe une bande comprise entre 10 et 15 MHz de chaque coté de la fréquence centrale. La valeur 15 MHz provient de la décroissance non idéale des lobes secondaires de la modulation utilisée. Il n'est donc pas possible d'utiliser dans la même zone géographique les canaux adjacents à ce canal.

Pour permettre à plusieurs réseaux d'émettre sur une même cellule, il faut allouer à chacun d'eux des canaux appropriés, qui ne se recouvrent pas. Par exemple, considérons deux réseaux utilisant DSSS. Si l'un d'eux utilise le canal 6, le canal 5 et 7 ne peut pas être utilisé par le deuxième réseau, car trop proche. Il en va de malheureusement de même pour les canaux 2, 3,4, 8, 9 et 10, qui ne peuvent non plus être alloués du fait de l'étalement de la bande passante du canal 6. Les canaux qui peuvent être utilisés sont les canaux 1, 11, 12, 13 et 14. Sachant que la largeur de bande n'est que de 83.5 MHz, il ne peut donc y avoir au maximum que trois réseaux 802.11 DSSS émettant sur une même cellule sans risque d'interférences.

Dans le standard 802.11 DSSS, La technique du « chipping sur 11 bits » aide à compenser le bruit généré par un canal donné, cette technique consiste à transmettre pour chaque bit une séquence Barker (parfois appelée bruit pseudo-aléatoire, noté PN) de bits. Ainsi chaque bit valant 1 est remplacé par une séquence de bits et chaque bit valant 0 par son complément. La couche physique de la norme 802.11 définit une séquence de 11 bits (*10110111000*) pour représenter un 1 et son complément (*01001000111*) pour coder un 0. On appelle *chip* ou chipping *code* (en français *puce*) chaque bit encodé à l'aide de la séquence *Barker*.

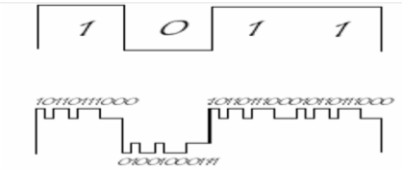

Figure 2.10 : Technique du chipping

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

Grâce au chippin*g*, de l'information redondante est transmise, ce qui permet d'effectuer des contrôles d'erreurs sur les transmissions.

Pour supporter les environnements plus bruyants et étendre la portée des équipements, les WLAN 802.11b utilisent la variation dynamique du débit, qui permet d'ajuster les taux de transmission automatiquement pour compenser les variations du canal radio. Dans une situation idéale, les utilisateurs se connectent à un taux de 11 Mbps plein.

Les caractéristiques du DSSS varient selon chaque pays, notamment ce qui concerne le nombre de sous canaux utilisés, ce qui peut remettre en cause la superposition de réseaux. Le tableau suivant montre ça :

Pays	**Etats-Unis**	**Europe**	**Japon**	**France**
Nombres de sous canaux utilisés	1 à 11	1 à 13	14	10 à 13

Tableau 2.2 : Nombre de canaux disponibles pour le DSSS en fonction du pays

Une trame au niveau physique est composée, comme pour la technique précédente, de trois parties : un préambule, puis un entête et enfin la partie données :

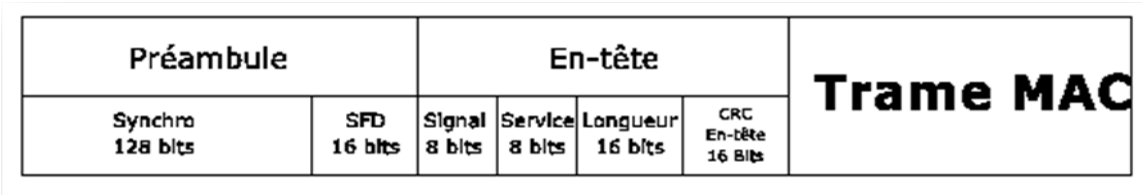

Figure 2.11 : Structure de la trame 802.11 au niveau physique pour le DSSS

- **Le préambule :**
  - La synchro est une séquence de synchronisation pseudo-aléatoire. Elle sert à la synchronisation au niveau récepteur.
  - Le Start Frame Delimiter (SFD) permet au récepteur de détecter le début de la trame. ce champ de deux octets vaut en hexadécimal F3A0.
- **L'entête :**
  - Le signal permet d'indiquer la vitesse de transmission sélectionnée. Si la valeur de ce champ est à 0A (en hexadécimal) la transmission se déroulera à 1Mb/s et si celle ci est à 14 (en hexadécimal), la transmission se déroulera à 2Mb/s. Il faut savoir qu'en fonction de la vitesse de transmission, une modulation différente est appliquée. Le differential binary phase shift

keying est utilisé lors d'une transmission à 1Mb/s et en opposition au Differential quadrature phase shift keying lors d'une transmission en 2Mb/s.

- Le service est réservé pour un usage futur La valeur 00 signifie que le transmetteur est conforme à la norme IEEE 802.11.
- La longueur indique la valeur de la longueur de la partie de données. Sa valeur peut varier entre 4 et 2^{16}.
- Le CRC de l'entête : est le champ de contrôle d'erreur de l'entête.

- **La partie donnée :**
  - La Trame MAC contient les données de la trame physique. Elles sont transmises selon la modulation sélectionnée dans le champ signal.
  - **IR (Infra Rouge):**

Le standard IEEE 802.11 prévoit également une alternative à l'utilisation des ondes radio : la lumière infrarouge. La technologie infrarouge a pour caractéristique principale d'utiliser une onde lumineuse pour la transmission de données. [21]

Ainsi les transmissions se font de façon unidirectionnelle, soit en "vue directe" soit par réflexion. Le caractère non dissipatif des ondes lumineuses offre un niveau de sécurité plus élevé.

Il est possible grâce à la technologie infrarouge d'obtenir des débits allant de 1 à 2 Mbit/s en utilisant une modulation appelé **PPM**.

La modulation *PPM* consiste à transmettre des impulsions à amplitude constante, et à coder l'information suivant la position de l'impulsion. Le débit de 1 Mbps est obtenu avec une modulation de *16-PPM*, tandis que le débit de 2 Mbps est obtenu avec une modulation *4-PPM* permettant de coder deux bits de données avec 4 positions possibles

La méthode IR se base sur la diffusion d'une lumière infrarouge de longueur d'onde comprise entre 850 et 950 nm (nanomètres). Grâce aux caractéristiques réflectives de l'infrarouge, les stations appartenant au réseau ne doivent pas nécessairement être dirigées les unes vers les autres. Cependant, vu la portée très faible de l'infrarouge, les stations ne peuvent être éloignées les unes des autres de plus d'une dizaine de mètres. Un réseau 802.11 IR ne peut donc être déployé que dans un espace ayant la dimension d'une pièce. [22]

ii. La modulation :

- **PSK (Phase Shift Keying):**

Cette technique est utilisée par la norme 802.11b. Chaque bit produit une rotation de phase. Une rotation de 180° permet de transmettre des débits peu élevés (technique appelée BPSK) tandis qu'une série de quatre rotations de 90° (technique appelée QPSK) permet des débits deux fois plus élevés grâce à l'optimisation de l'utilisation de la bande radio. [20]

- **OFDM (Orthogonal Frequency Division Multiplexing):**

OFDM est une méthode de codage appliquée aux normes 802.11a et g qui permet d'obtenir une meilleure bande passante. De ce fait, OFDM divise la bande de fréquence en bandes secondaires qui transmettent simultanément des fractions de données. Plus le nombre de canaux est élevé, plus les données transmises en parallèle sont nombreuses, plus la bande passante est élevée. Selon les conditions de bande passante, OFDM peut utiliser des méthodes de modulation de phase et d'amplitude.

OFDM est plus efficace que DSSS à savoir : en fonctionnant avec une même bande de fréquences (2,4000 - 2,4835 GHz), 802.11g a une bande passante de 54 Mbps avec OFDM, alors que 802.11b monte seulement jusqu'à 11 Mbps avec DSSS.

Le tableau suivant présente les différentes méthodes de codage pour le 802.11 a, b et g :

Paramètres	Standards		
	802.11a	802.11b	802.11g
Bande de fréquence (GHz)	5.15-5.35 5.725-5.825	2.4000-2.4835	2.4000-2.4835
Méthode d'encodage	OFDM	DSSS	OFDM (et DSSS pour une compatibilité avec 802.11b)
Bande passante maximale	54 Mbps	11 Mbps	54 Mbps

Tableau 2.3: Méthodes de codages pour le 802.11 a, b et g

II.4.2 La couche liaison de données :

La couche Liaison de données de la norme 802.11 est composée de deux sous-couches : la couche de contrôle de la liaison logique (LLC) et la couche de contrôle d'accès au support (MAC) [22]

En plus des fonctions habituellement rendues par la couche MAC, la couche MAC 802.11 offre d'autres fonctions qui sont normalement confiées aux protocoles supérieurs, comme :

- la fragmentation et le réassemblage des trames
- le contrôle d'accès au support

- l'adressage et le formatage des trames.
- le contrôle d'erreur sur la trame, à partir d'un CRC (Cyclic Redundancy Chek)
- **la qualité de service**
- la gestion de l'énergie
- la gestion de la mobilité
- la sécurité

Le contrôle d'accès au support est une fonctionnalité qui nous intéresse ici particulièrement, se fait suivant deux méthodes (DCF et PCF) qui seront étudiées ultérieurement.

i. La trame MAC 802.11 : [22]

La norme a défini 3 types de trames MAC :

- Les trames de données : pour véhiculer les données à transmettre.
- Les trames de contrôle : utiles dans la procédure d'accès au canal (RTS, CTS, ACK).
- Les trames de gestion : contiennent des informations de gestion et ne sont pas remontées au niveau OSI supérieur (trames Beacon contenant les informations de synchronisation).

- **La trame de données MAC 802.11 :**

Comme l'illustre la figure ci dessous, une trame de données MAC 802.11 est constituée de trois parties :

- Entête MAC.
- Données MAC : Données reçues des couches supérieures et à encapsuler.
- CRC: champ de 32 bits contenant la somme de contrôle de la trame.

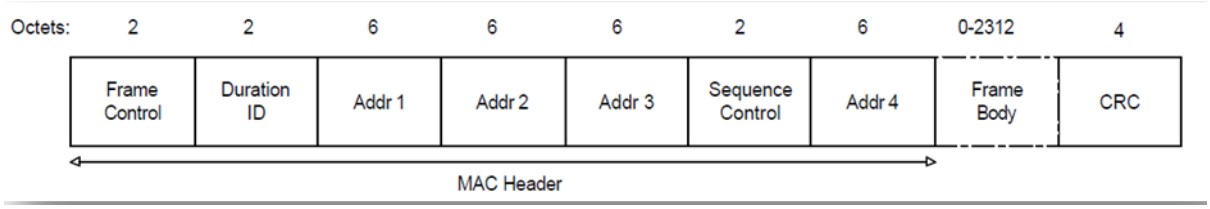

Figure 2.12 : Format de la trame MAC 802.11

L'entête MAC comporte les 7 champs suivants :

- ***Frame Control*** (2 octets) : renferme des informations de contrôle concernant la trame : version utilisée, type de la trame, mode de gestion de puissance, type de cryptage …

Ce champ contient en particulier les sous-champs FromDS qui indique nt si la trame est reçue de la part d'un DS (système de distribution), et ToDS qui indique nt si la trame est destinée à un DS.

- ***Duration ID*** (2 octets) : Indique la durée calculée pour le NAV (Network Allocation Vector).
- ***Adresse 1*** (6 octets) : Adresse de la station réceptrice. Si ToDS =1, alors c'est l'adresse de l'AP correspondant.
- ***Adresse 2*** (6 octets) : Adresse de l'émetteur. Si FromDS =1, c'est l'adresse du point d'accès
- ***Adresse 3*** (6 octets) : Adresse perdue, par exemple si FromDS = 1, Adresse 2 contient l'adresse du point d'accès et Adresse 3 celle de la station source d'origine
- ***Sequence Control*** (2 octets) : représente l'ordre des différents fragments d'une même trame. Ce champ permet aussi de reconnaître la duplication des paquets. Il est constitué de deux sous champs : *Fragment Number* et *Sequence Number* pour respectivement la trame et l'indice du fragment dans cette trame.
- ***Adresse 4*** (6 octets) : Utilisée dans des cas spéciaux tels que la transmission entre points d'accès, quand les ToDS et FromDS sont à 1.

- **Les trames de contrôle MAC 802.11**

La norme a prévu d'autres formats pour les trames de contrôle, en particulier les trames RTS,CTS et ACK.

- Les trames RTS et CTS sont utilisées pour la réservation virtuelle des ressources dans le cadre de la procédure d'accès au support physique.
- La trame ACK est utilisée pour acquitter les transmissions réussies. Elle est envoyée par une station réceptrice, ayant correctement reçu une trame de données, à la station source.

Les trames RTS, CTS et ACK sont constituées chacune par un et un entête MAC. L'entête MAC comporte quelques différences suivant qu'il s'agisse de trames RTS, CTS ou ACK :

- L'entête de la trame RTS comprend les champs suivant :
  - Frame Control : analogue au champ de la trame de données MAC.
  - Duration : Durée à réserver.
  - RA : Adresse de la station réceptrice.

- TA : Adresse de la station émettrice.

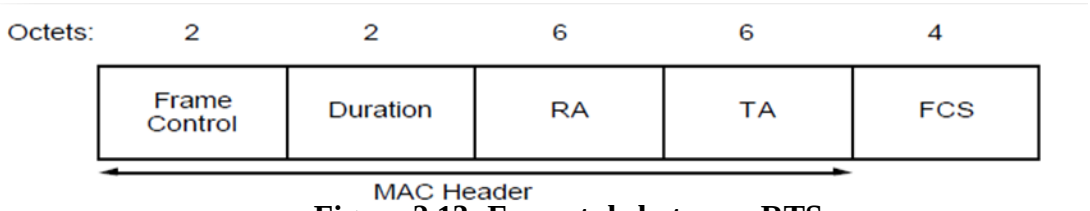

Figure 2.13: Format de la trame RTS

- L'entête de la trame CTS comprend les même champs que celui de RTS, hormis le champ TA. Le champ RA étant recopié à partir du champ TA de la trame RTS reçue.

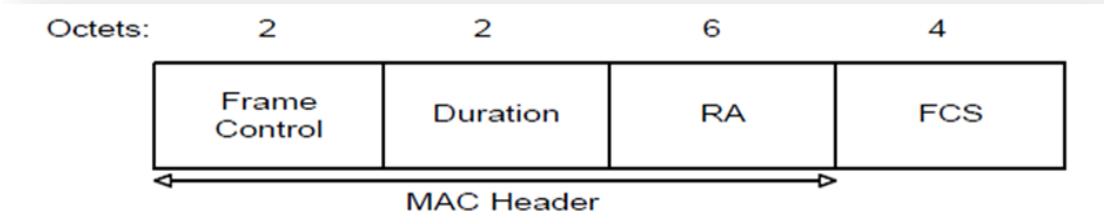

Figure 2.14: Format de la trame CTS

- L'entête de la trame ACK possède un format similaire à celui de CTS. L'adresse RA est recopiée à partir du champ Adresse 2 de la trame MAC à acquitter.

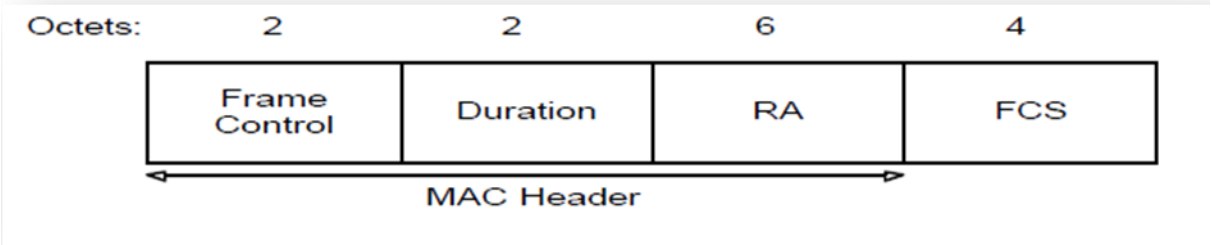

Figure 2.15 : Format de la trame ACK

iii. Protocoles d'accès au médium sans fil pour la norme IEEE 802.11 :

Comme mentionné plus haut, la principale fonctionnalité de la couche MAC 802.11 est de définir les mécanismes d'accès au support physique.

Le standard définit deux méthodes d'accès au support: DCF et PCF.

- **Le Distributed Coordination Function (DCF) : [13], [19], [22], [24]**

La technique DCF est basée sur le mécanisme CSMA/CA ou méthode d'accès multiple à détection de porteuse et évitement de collision. Cet algorithme distribué est exécuté localement sur chaque station afin de déterminer les périodes d'accès au médium.

- **Pourquoi CSMA/CA ?**

Les réseaux locaux sans fils adoptent la méthode d'accès CSMA/CA au lieu de la méthode CSMA/CD généralement utilisée dans les réseaux LANs classiques.

La méthode CSMA/CD consiste, pour une station désirant transmettre des données, à écouter le canal. Si le canal est libre alors la station peut transmettre. Sinon, elle attend que le canal redevienne libre. La station doit pouvoir détecter d'éventuelles collisions. Elle avortera dans ce cas la transmission et tentera de réémettre ultérieurement.

L'utilisation de cette méthode s'avère très coûteuse pour des réseaux sans fils. En effet, pour pouvoir implémenter la méthode CSMA/CD on doit disposer d'un circuit full duplex pour la détection de collision.

Ainsi, la méthode CSMA/CA a été retenue pour les WLANs puisque le canal varie au cours du temps. Cette méthode abandonne la détection de collisions, tout en renforçant les mécanismes pour les éviter. Dans un environnement radio-mobile, ce n'est pas possible d'appliquer le CSMA/CD.

- **Description générale du mécanisme DCF :**

Avant chaque émission, la station désirant émettre écoute le support. S'il est libre pendant une certaine durée DIFS (démontrer plus bas), la transmission est possible. Si le support est occupé, une procédure de *Backoff* est enclenchée.

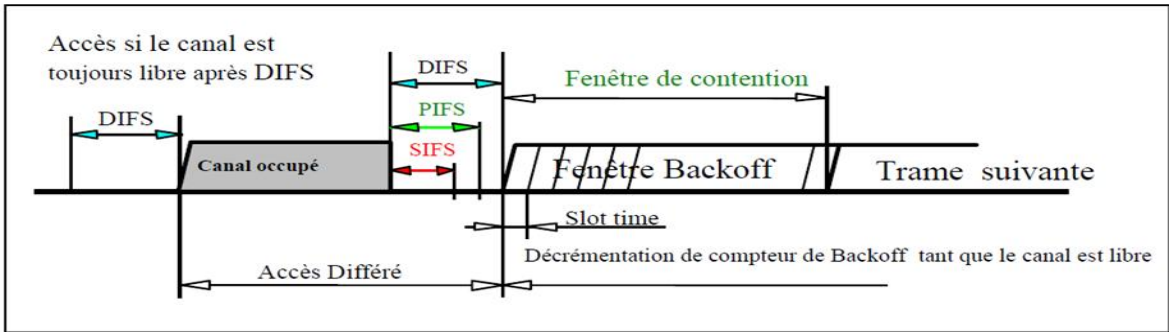

Figure 2.16: Accès au médium en mode DCF

Une station ayant correctement reçu un paquet, renvoie un accusé de réception (ACK) à la station émettrice. L'ACK indique à l'émetteur qu'aucune collision n'a eu lieu. Par contre, si l'émetteur ne reçoit pas d'acquittement au bout d'un certain temps, le fragment est retransmis jusqu'à réception d'un acquittement par le récepteur.

Enfin, si après un nombre défini de retransmissions, aucun accusé de réception n'est reçu, l'émission est abandonnée.

- **IFS (Inter-Frame Space) :**

Un espace inter-trames IFS est la durée pendant laquelle une station doit attendre avant de transmettre sur le canal. Pour définir les différentes sortes d'IFS, la norme a tout d'abord introduit la notion de Time Slot comme étant l'intervalle de temps qui

permet à une station de savoir si une autre station a accédé au canal au début du slot précédent. La valeur d'un Time Slot dépend de la couche physique utilisée. Pour la couche PMD à étalement de spectre à séquence directe, cette valeur est 20 µs.

A partir de la notion de Time Slot, la norme a ensuite introduit 4 types d'espaces inter trames, définis comme suit :

- **Short Inter-Frame Spacing (SIFS):**

Est le plus court des IFS. Il est utilisé pour séparer les différentes trames transmises au sein d'un même dialogue comme par exemple, entre des données et leurs acquittements ou entre différents fragments d'une même trame ou pour toute autre transmission relative à un même dialogue (question-réponse).

- **DCF Inter-Frame Spacing (DIFS) :**

Est le temps que doivent attendre les autres stations avant d'émettre un paquet en mode DCF. La valeur du DIFS est égale à celle d'un SIFS augmentée de deux times slot.

- **PCF Inter-Frame Spacing (PIFS) :**

Est le temps que doit attendre les autres stations avant d'émettre un paquet en mode PCF. La valeur est inférieure au DIFS, pour permettre de favoriser ce mode. Le mode PCF est expliqué dans la partie suivante.

- **Extended Inter-Frame Spacing (EIFS) :**

Est le plus long des IFS. Lorsqu'une station reçoit une trame erronée, elle doit attendre pendant un EIFS l'acquittement de cette trame.

- **L'algorithme de Backoff :**

La procédure de Backoff est un mécanisme simple, basé sur le calcul d'un temporisateur gérant les transmissions et les retransmissions. Il permet de réduire la probabilité de collision sur le canal en essayant de minimiser les chances d'avoir plusieurs stations qui accèdent au support en même temps.

- **Déroulement :**

Une station S désirant envoyer des données attend pendant une période DIFS. Si après cette durée le canal est libre, la station accède directement au canal. Dans le cas contraire, la station déclenche le mécanisme de Backoff qui se déroule en 3 étapes :

✓ La station calcule son temporisateur Backoff_Timer :

Backoff_Timer _Random ( ) x TS

Avec :

Random ( ) : nombre pseudo-aléatoire choisi ente 0 et CW-1 ; où CW est la taille de la fenêtre de contention qui sera détaillée plus loin.

TS : durée d'un time-slot définie comme étant l'intervalle de temps nécessaire pour une station pour savoir si une autre a accédé au canal au début du time-slot précédant.

- ✓ Quand le canal devient libre, et après un DIFS, la station commence à décrémenter son temporisateur time-slot par time-slot.
- ✓ Lorsque la valeur de Backoff_Timer est égale à 0, la station peut alors envoyer. Si par contre au cours de la phase de décrémentation, une autre station S' termine de décrémenter son temporisateur, la station S bloque son temporisateur. Elle pourra continuer de le décrémenter une fois la transmission de la station S' finie.

- **Fenêtre de contention :**

La taille de la fenêtre de contention CW a pour valeur initiale CWmin. Deux cas de figures peuvent se présenter :

- ✓ *Transmission réussie :* dans ce cas, CW est réinitialisée à CW_{min}.
- ✓ *Transmission échouée :* c'est-à-dire que la station émettrice ne reçoit pas d'acquittement au bout d'un certain temps. CW est alors incrémenté de la façon suivante:

$$CW_{new} \_ 2*CW_{old} +1$$

La station suppose dans ce cas qu'il y a eu collision lors de la transmission, et incrémente la taille de sa fenêtre de contention afin de diminuer les chances de collisions lors des prochaines retransmissions. Une valeur limite CWmax est cependant définie.

Si pour CW = CWmax la transmission échoue toujours, la valeur n'est plus incrémentée et est maintenue à CWmax.

La figure 2.18 montre un diagramme de variations de la taille de la fenêtre de contention en fonction du nombre de tentatives de transmissions.

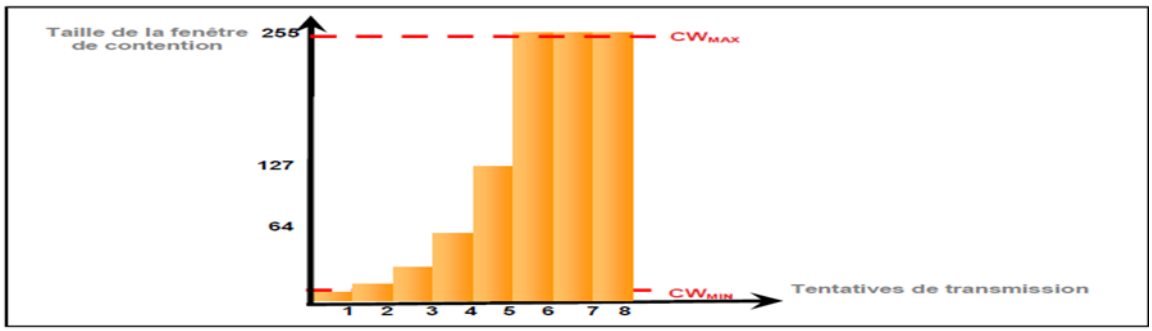

Figure 2.17: Exemple typique de la variation de la taille de la fenêtre de contention

iv. Diagramme de fonctionnement

La figure 2.19 résume le fonctionnement de la procédure CSMA/CA et de l'algorithme de Backoff.

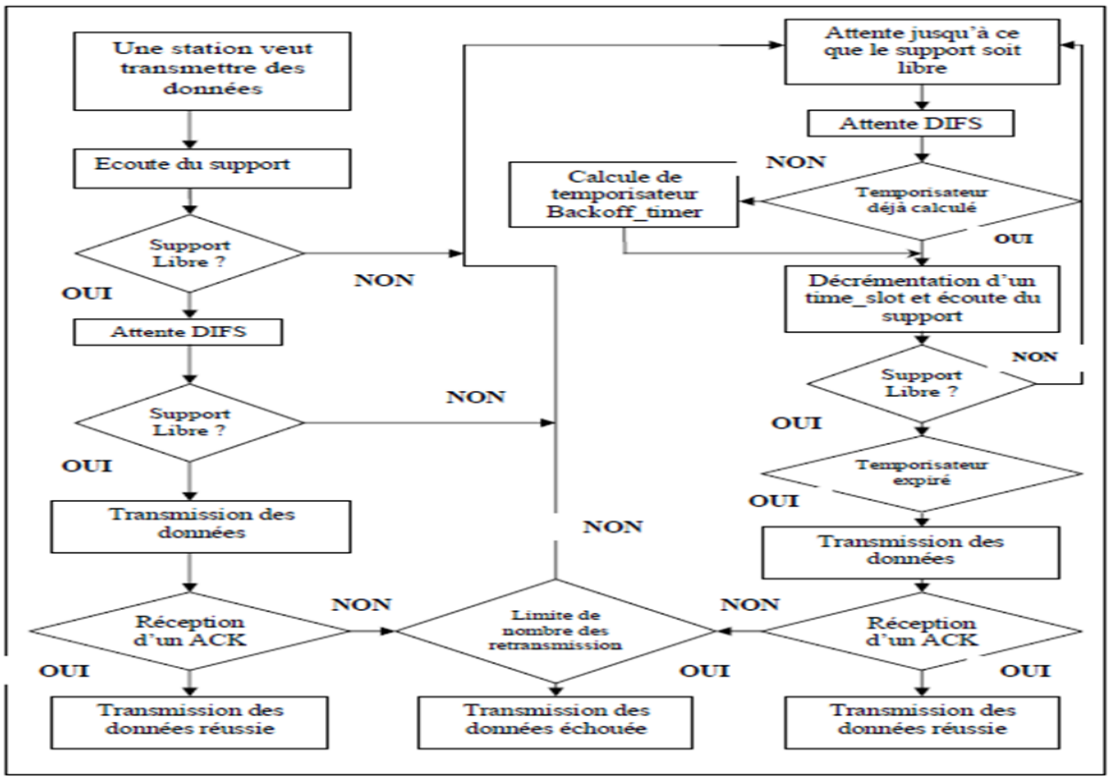

Figure 2.18: Procédure CSMA/CA

v. Le mécanisme de Virtual carrier Sense VCS :

Dans le but de résoudre le problème des stations cachées, le standard 802.11 définit sur sa couche MAC un mécanisme optionnel de type RTS/CTS appelé mécanisme VCS. Lorsque cette fonction est utilisée, une station émettrice transmet un RTS et attend en réponse un CTS. Toutes les stations du réseau recevant soit RTS soit le CTS, déclencheront pour une durée fixée leur indicateur NAV pour retarder toutes transmissions prévues. La station émettrice peut alors transmettre et recevoir son accusé de réception sans aucun risque de collision.

vi. Description générale du mécanisme PCF:

Le **PC** normalement installé sur l'AP, contrôle l'accès au médium par la méthode du Polling. Il faut noter que **PCF** est optionnel, et peut donc être implémenté avec **DCF**.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

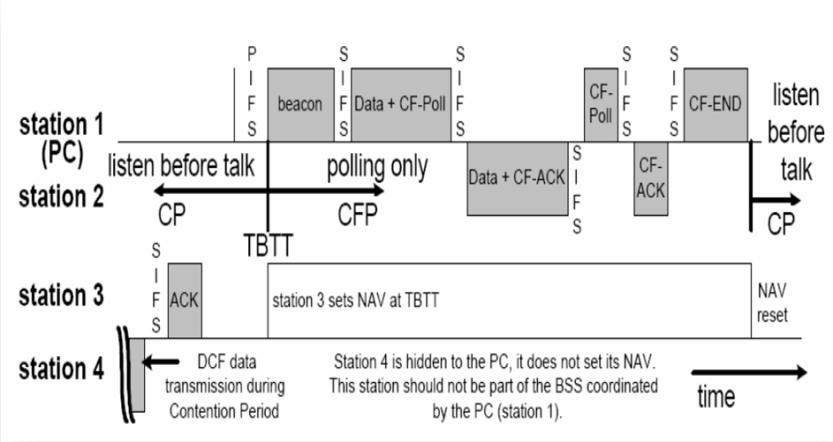

Figure2.19 : principe du fonctionnement du PCF

Le **PC** après un temps **PIFS** pendant lequel le canal est libre envoie la balise " **Beacon** " qui marque le début de la super trame, divisée en deux parties: la CFP et la CP. Initialement la durée maximale de la CFP CFPMaxDuration, ainsi que sa fréquence sont données; mais cette dernière n'est pas respectée la plupart du temps car le beacon peut être retardé, à cause d'une longue transmission d'une trame à la fin de la CP. Ce problème ne permet pas d'avoir une séquence rigoureusement périodique de la balise. Comme PCF a été développé au dessus de DCF toutes les stations doivent activer leur NAV au début de la CFP à la valeur CFPMaxDuration pour bloquer toute transmission parasite (contention) pendant la durée CFP, car aucune station n'a le droit d'émettre que si on le lui demande pendant la CFP.

Pendant la durée du CFP le PC attend une durée SIFS après le beacon avant d'envoyer une trame de données ou le CF-Poll ou faire du piggybacking (une trame de données qui contient aussi un message de polling). Le PC va séquentiellement faire du polling, pendant la durée CFP, pour toutes les stations déjà enregistrées dans sa liste. La station concernée va répondre au PC ou à une autre station dans le réseau par des trames de données ou un ACK séparés par SIFS. Si une station ne répond pas, le PC passe à la suivante après PIFS. Si le PC ou les stations n'ont plus de trames a transmettre, la CFP se termine par l'envoi de la trame CF-end par le PC. Toutes les stations vont alors remettre a zéro leur NAV, et la CP va débuter, et on repasse alors au mode DCF. Il faut noter qu'aucune station n'a le droit d'émettre que si on le lui demande pendant la CFP.

Si le PCF est utilisé pour les applications à contraintes temporelles, le PC doit établir une liste de polling. Chaque station doit être votée au moins une fois par CFP. Les stations peuvent demander une place dans la liste de polling avec des trames de gestion d'associations. Le PC peut avoir un modèle de priorité pour les différentes stations.

III. Conclusion :

Afin de gérer la priorité d'accès au support et garantir la qualité de service pour les trafics multimédia, le groupe IEEE 802.11 a développé de nouveaux mécanismes dans le but de garantir une certaine qualité de service.

Ce standard repose sur deux mécanismes d'accès : EDCF qui fonctionne durant la période CP et HCF qui fonctionne durant les deux périodes.

Le chapitre suivant a pour objectif d étudier ces deux mécanismes ainsi que tous les autres aspects liés a la Qualité de service.

I. Introduction :

Les réseaux locaux basés sur la technologie IEEE 802.11 ont pris une ampleur telle qu'ils sont déployés un peu partout dans notre entourage quotidien (aéroports, hôtels, gares, campus, etc.).Ce déploiement est favorisé par la maturité atteinte par le standard grâce aux travaux des groupes 802.11 chargés de rendre le standard plus compétitif (QoS, sécurité, haut débit).

Le groupe de travail 802.11 e répond aux challenges de garantie de la qualité se service (QoS) aux applications temps réel en définissant de nouveaux mécanismes d'accès au médium. Le draft résultant des travaux du groupe 802.11 e propose deux nouveaux mécanismes : EDCA et HCCA.

Dans ce chapitre nous nous intéresserons plus particulièrement à la gestion de la QoS et ses contraintes dans les réseaux IEEE 802.11. Dans un premier temps, nous passerons en revue les mécanismes de QoS les plus significatifs proposés dans la littérature ; nous nous attarderons sur les deux mécanismes du draft 802.11 e EDCA et HCCA. Dans un deuxième temps, nous introduirons un autre mécanisme de la QoS c'est le protocole du lien direct permettant la communication directe entre les stations dans un mode de fonctionnement avec infrastructure.

II. Généralité sur la qualité de service :

II.1 Définition de la QoS:

Plusieurs définitions ont été proposées pour le terme de la qualité de service dont les plus importantes sont :

- La **Qualité de Service** (QoS) est la capacité à véhiculer dans de bonnes conditions un type de trafic donné, en termes de disponibilité, débit, délais de transmission, taux de perte de paquets…
- La **Qualité de Service** est une notion subjective. Selon le type d'un service envisagé, elle pourra résider dans le débit (Un débit permet de mesurer le flux d'une quantité relative à une unité de temps au travers d'une surface quelconque.), le délai (pour les applications interactives ou la téléphonie), la disponibilité (accès à un service partagé) ou encore le taux de pertes de paquets (pertes sans influence de la voix ou de la vidéo (La vidéo regroupe l'ensemble des techniques, technologie, permettant l'enregistrement ainsi que la restitution d'images animées…). [1]
- La **Qualité de Service** regroupe un ensemble de technologies mises en œuvre pour assurer des débits suffisants et constants sur les réseaux, y compris Internet. [2]

II.2 But de la QoS :

Le but de la QoS est donc d'optimiser les ressources du réseau (Un réseau informatique est un ensemble d'équipements reliés entre eux pour échanger des informations. Par analogie avec un filet (un réseau est un « petit rets », c'est-à-dire un petit filet), on appelle nœud (node) l'extrémité d'une connexion, qui peut être une intersection de plusieurs connexions (un ordinateur, un routeur, un concentrateur, un commutateur) et de garantir de bonnes performances aux applications critiques. La Qualité de Service sur les réseaux permet d'offrir aux utilisateurs des débits et des temps (Le temps est un concept développé pour représenter la variation du monde : l'Univers n'est jamais figé, les éléments qui le composent bougent, se transforment et évoluent pour l'observateur qu'est l'homme. Si on considère l'Univers...) de réponse différenciés par application suivant les protocoles mis en œuvre au niveau de la couche réseau.

Elle permet ainsi aux fournisseurs de services (départements réseaux des entreprises, opérateurs…) de s'engager formellement auprès de leurs clients sur les caractéristiques de transport (Le transport, du latin trans, au-delà, et portare, porter, est le fait de porter quelque chose, ou quelqu'un, d'un lieu à un autre.) des données (Dans les technologies de l'information (TI), une donnée est une description élémentaire, souvent codée, d'une chose, d'une transaction d'affaire, d'un événement, etc.) applicatives sur leurs infrastructures IP. [4]

Selon le type d'un service envisagé, la qualité pourra résider :

- Le débit (téléchargement ou diffusion vidéo).
- Le délai (pour les applications ou la téléphonie).
- La disponibilité (accès à un service partagé).
- Le taux de pertes de paquets. [5]

II.3 Services de la QoS :

La mise en place de la qualité de service nécessite en premier lieu la reconnaissance des différents services:

- La source et la destination du paquet.
- Le protocole utilisé (UDP/TCP/etc.).
- Les ports de source et de destination dans le cas TCP et UDP.
- La congestion des réseaux.

- La validité du routage (gestion des pannes dans un routage en cas de routes multiples par ex.)
- La bande passante consommée.
- Les temps de latence.

II.4 Critères de la QoS :

Les principaux critères permettant d'apprécier la qualité de service sont les suivants :

- **Débit** (en anglais bandwidth): parfois appelé bande passante, il définit le volume maximal d'information (bits) par unité de temps (b/s).
- **Perte de paquet** (en anglais *packet loss*): elle correspond à la non-délivrance d'un paquet de données, la plupart du temps due à un encombrement du réseau.
- **Gigue** (en anglais jitter) : C'est un paramètre important pour les applications communicantes de type voix ou vidéo où la gigue doit être la plus faible possible. La gigue est due principalement aux délais de transferts variables dans les nœuds du réseau (swicthes et routeurs).
- **Latence** (en anglais *delay*) : elle caractérise le retard entre l'émission et la réception d'un paquet. [5]

II.5 Degrés de la QoS :

Les trois principaux degrés de Qualité de Service (trois niveaux de services), du plus fiable au plus lâche, sont les suivants :

II.5.1 Le service garanti ou premium :

Il vise à émuler une liaison spécialisée : malgré un multiplexage des paquets sur le médium, le lien propose les mêmes garanties que s'il était basé sur une ligne indépendante. Des pertes de paquets ou une certaine gigue peuvent néanmoins être acceptées en fonction du contrat négocié. Au niveau technologies, le service garanti se retrouve avec le GS d'IntServ, l'EF de DiffServ et le CBR de l'ATM que nous détaillerons plus loin.

II.5.2 Le service « mieux que Best-Effort ».

II.5.3 Le service Best-Effort :

Le protocole IP de base en est un exemple, ou encore UBR de l'ATM. [6]

III. Qualité de service suivant le standard IEEE 802.11 :

Pour assurer une qualité de service adéquate dans les réseaux sans fil le standards IEEE 802.11 à définit deux méthodes d'accès au canal:

- Distributed Coordination Function (DCF)

- Point Coordination Function (PCF)

Les deux méthodes sont bien illustrer dans le chapitre précédant.

III.1 Problématique de la QoS dans les réseaux IEEE 802.11:

Le développement du réseau Internet et le grand nombre d'utilisateurs connectes a ce réseau imposent le recours a des supports de qualité de service. Dans cette perspective, plusieurs groupes de travail ont vu le jour pour les réseaux filaires. Les nouveaux besoins en termes de mobilité des utilisateurs et la croissance des réseaux permettant le nomadisme des utilisateurs ont fait migrer le problème vers la boucle locale sans fils, entre autres les réseaux IEEE 802.11. Actuellement, le marché des télécommunications des réseaux Hots-pot est relativement faible mais on s'attend à ce qu'il subisse une croissance accrue les prochaines années. Les fournisseurs d'accès a Internet commencent à mettre en place un large nombre de hots-pots 802.11 ou Wifi dans les divers lieux publics. Des applications multimédia telles que la voix sur IP ou la vidéo sur demande en plus des applications classiques seront de plus en plus utilisées dans ce type de réseaux. Ces applications multimédia nécessitent un niveau minimal de qualité de service en termes de bande passante, de délai, de gigue ou de taux de perte. D'autres types d'applications avec des contraintes plus aigues en termes de QoS commencent à émerger. Des applications du standard 802.11 en milieu industriel pour la commande et la supervision des systèmes ou en milieu médical pour la télémédecine imposent des exigences strictes en termes de QoS (délais + taux d'erreurs). La réponse à ces besoins accrus en QoS dans les hots-pots 802.11 est d'autant plus difficile a cause des caractéristiques spécifiques du medium sans fils. En effet, pour la couche physique DSSS permettant un débit au-delà de 11 Mbps, parmi 11 canaux possibles, seulement 3 ne se chevauchent pas. Ce medium présente alors un taux de perte assez élève à cause des interférences. En plus, les caractéristiques du support physique ne sont pas constantes et varient dans le temps et dans l'espace. Quand les utilisateurs bougent, les chemins de bout en bout changent et les utilisateurs se réassocient chaque fois a des nouveaux APs.

Ces utilisateurs doivent avoir la même QoS indépendamment de leurs associations et du chemin de bout en bout du trafic. Plusieurs travaux de recherche ont essaye d'évaluer les performances du standard IEEE802.11 quant a sa capacité de répondre aux besoins en termes de QoS des utilisateurs. Ces travaux ont investigue essentiellement les possibilités offertes par la sous couche MAC du standard pour garantir un niveau

minimal de QoS pour les utilisateurs. Dans le même objectif, d'autres travaux ont adopte des modèles analytiques ou des approches par simulation. Plusieurs solutions ou approches pour l'amélioration du support de QoS par la couche MAC 802.11 ont été proposées. [3]

Toutes ces insuffisances dans les modes de fonctionnement DCF et PCF du standard ont conduit à plusieurs activités de recherche pour améliorer les performances de la sous couche MAC 802.11.

III.2 Limites en termes de QoS du standard IEEE 802.11 : [3]

Le contrôle d'accès au medium, le maintien de la QoS et la sécurité sont les fonctions les plus importantes de la sous couche MAC 802.11. Cependant plusieurs limitations se présentent quant au support de la qualité de service.

III.2.1 Limitations de la méthode d'accès de base DCF :

Le protocole CSMA/CA utilisé avec cette méthode permet un accès Best Effort au canal. Les utilisateurs ne peuvent avoir aucune garantie de qualité de service minimale. Toutes les stations d'un même BSS concourent pour l'accès au canal et aux ressources du réseau avec les mêmes priorités. Aucun mécanisme de différenciation entre plusieurs types de flux n'est mis en place pour garantir la bande passante, le délai de bout en bout ou la gigue pour des trafics a hautes priorités tels que la voix sur IP ou la vidéo/visioconférence. Le taux des erreurs dues a la couche physique 802.11 est a peu prés trois fois plus grand que celui observe dans les réseaux locaux filaires. Le nombre important de collisions et de retransmissions implique des délais de transmission imprévisibles et une dégradation de la qualité de transmission des flux temps réel tels que pour la voix ou la vidéo.

III.2.2 Limitations de la méthode d'accès PCF :

Spécialement conçue pour apporter un support de qualité de service en priorisant les applications temps réel par rapport aux autres, cette procédure d'accès avec scrutation souffre de plusieurs défaillances. Tout d'abord ce mode ne peut être utilisé qu'en alternance avec le mode d'accès DCF et ne peut jamais fonctionner a part entière. PCF présente tous les inconvénients d'une approche centralisée tel que l'effet d'une défaillance du point central. En plus, à faible charge, les stations voulant émettre en mode PCF subiront des délais très élevés.

Elles seront obligées d'attendre d'être scrutées avant d'émettre. De plus, le coordinateur (généralement confondu avec le point d'accès) doit systématiquement

accéder au canal sans fil lors de la période DCF afin de débuter la période PCF suivante. Dans le mode PCF, il sera très difficile de répondre aux besoins d'un nombre important de trafics temps réel sans pénaliser les applications qui se dérouleront par la suite dans la période avec contention. Un autre problème de ce mode est l'impossibilité de prévoir la durée de transmission des stations sollicitées. Une station sollicitée par le point coordinateur peut transmettre un MSDU de taille maximale 2304 octets. Cependant, le standard n'empêche pas sa fragmentation en plusieurs MPDU. Ceci, en plus des débits de transmission dépendant de l'état du canal physique, conduit a une durée de transmission d'un MSDU non contrôlée par le point coordinateur ce qui induira des délais supplémentaires pour le reste des stations en mode PCF. Enfin le mode PCF est géré par un algorithme de scrutation Round-Robin à une seule classe. Il ne lui est donc pas possible de répondre aux besoins de QoS de plusieurs types de flux (voix, vidéo,…).

III.3 Les différentes solutions de QoS dans les réseaux IEEE 802.11 :

Depuis l'écriture du standard IEEE 802.11 a la fin des années 90, plusieurs propositions, issues de travaux de recherches et/ou d'initiatives de la part de constructeurs, ont vu le jour pour l'amélioration du support de qualité de service dans ces réseaux. Un groupe de travail spécifique à été forme au sein de l'IEEE dans l'objectif de normaliser des amendements de la qualité de service pour le protocole 802.11. La norme 802.11e a ainsi été élaborée. Elle reprend entre autres des techniques introduites dans divers travaux de recherche. Dans la suite de ce chapitre nous présentons tout d'abord la norme IEEE 802.11e puis nous présenterons plusieurs approches visant à améliorer la QoS dans les réseaux 802.11.

III.3.1 Le nouveau standard IEEE 802.11 e :

Pour supporter la qualité de service, le groupe de travail "e" du standard 802.11 définit des améliorations de la couche MAC de 802.11 en introduisant une fonction de coordination hybride HCF. Ce dernier définit deux mécanismes d'accès au canal (synonyme d'accès au medium dans 802.11e) : accès avec contention et accès contrôlé. La méthode d'accès avec contention est nommée EDCA. La deuxième méthode, offrant un accès contrôlé, est nommée HCCA. Les stations sans fils opérant sous 802.11e sont appelées stations améliorées. La station améliorée qui joue le rôle de contrôleur central au sein de la même cellule QBSS est appelée le point de coordination hybride (HC). Le point de coordination hybride est typiquement combine au point d'accès. Un QBSS est un BSS qui inclut un HC et des stations améliorées. Les paramètres QoS sont ajustes au

Rapport-Gratuit.com

cours du temps par le coordinateur hybride et sont annonces périodiquement a travers les trames balises. Plusieurs entités de Backoff (Backoff Entity) fonctionnent en parallèle dans une station améliorée. Une entité de Backoff est une file de transmission pour une classe de trafic bien déterminée avec des paramètres d'accès au canal spécifiques. Une station 802.11e ou plus précisément une entité de Backoff ne peut utiliser le canal que pour une durée limitée. L'intervalle de temps durant lequel la station a le droit d'émettre est appelé l'opportunité de transmission TXOP. TXOP est défini par un instant de début et une durée. Un intervalle TXOP obtenu suite à une contention au canal est appelé EDCA-TXOP. Quand cet intervalle est obtenu dans la période contrôlée par le HC, il est appelé HCCA-TXOP. La durée d'une EDCA-TXOP est limitée par la valeur du paramètre QBSS-limit-TXOP régulièrement distribuée par le point d'accès à travers les trames balises (beacon). Ce paramètre permet donc de contrôler la durée maximale d'une transmission en cours ce qui est important pour les délais d'accès et de transmission de l'ensemble des stations. L'utilisation de ce paramètre permet aussi d'assurer a un instant précis et sans retard, le démarrage de chaque période d'accès contrôlée par le HC.

Une autre amélioration est apportée par le nouveau standard : les stations améliorées sont maintenant autorisées à transmettre directement des trames a une autre entité du QBSS sans être obligées de passer par le point d'accès. Ce fait permet d'optimiser l'utilisation de la bande passante partagée entre les utilisateurs. Dans le standard 802.11, toutes les communications passaient obligatoirement par le point d'accès. [3]

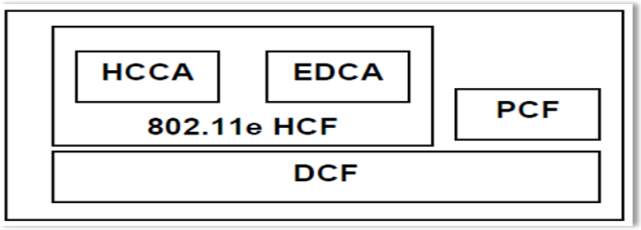

Figure 3.1 : Architecture de la norme 802.11e

i. HCF : une fonction d'accès au médium avec QoS

HCF est utilisée uniquement dans ce que le standard appelle un réseau QoS (c'est le réseau où le point d'accès met en place un HC). La modification IEEE 802.11e a introduit cette nouvelle méthode ainsi que d'autres mécanismes afin d'apporter certaines propriétés QoS au niveau de l'accès. HCF introduit des modifications à DCF et PCF ainsi qu'un certain nombre de mécanismes et de types de trames permettant la mise en place de transferts avec qualité de service pendant la CP et la CFP. HCF

introduit la notion d'opportunité de transmission (TXOP) qu'une QoS-STA peut obtenir en utilisant l'une des méthodes d'accès d'HCF : la méthode d'accès avec contention EDCA ou la méthode d'accès par scrutation (HCF). L'obtention d'un TXOP peut permettre l'envoi d'une ou plusieurs trames. Si TXOP vaut 0, une seule trame donnée peut être envoyée par opportunité de transmission. [9]

- **La méthode d'accès EDCA : [5], [7]**

Il s'agit d'une amélioration du DCF qui ajoute un système de priorité pour la gestion de l'accès au support. Ce dernier se fait alors selon le niveau de priorité de la trame. Selon la définition du dernier draft de la norme 802.11e, la couche MAC au niveau d'une station est formée de quatre files de transmission dont chacune fonctionne comme une entité de Backoff en mode DCF. La structure de cette couche est illustrée par la figure 3.1.

La norme IEEE 802.11e a donc défini, au niveau MAC, quatre catégories d'accès : AC relatives aux applications traitées dans les couches supérieurs. Chaque catégorie de trafic constitue une file d'attente FIFO. Elles sont notées respectivement :

- ***AC_VO*** : pour les applications temps réels tel que la voix
- ***AC_VI*** : pour les applications vidéo
- ***AC_BE*** : pour le trafic « Best Effort »
- ***AC_BK*** : pour le trafic Background

Pour introduire la notion de différentiation entre les différentes AC, Chaque catégorie de trafic possède son propre DIFS, on parle donc de AIFS. Ces catégories de trafic, gèrent huit niveaux de priorités allant de 0 à 7 relatives à la norme 802.1D. Les correspondances entre ces priorités et les catégories d'accès sont récapitulées aussi au niveau de la figure 3.1. En outre, il est important de signaler que les tailles limites de la fenêtre de contention diffèrent selon la classe de trafic. On parle alors de CW_{Min} [AC] et CW_{Max} [AC].

Chaque AC détient son propre compteur de Backoff qui est désormais compris entre 1 et 1 + CW [AC].

Quand deux ACs finissent en même temps leur durée de Backoff, alors c'est le paquet de plus haute priorité qui sera transmis, les autres entités doivent augmenter leurs fenêtres de Backoff.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

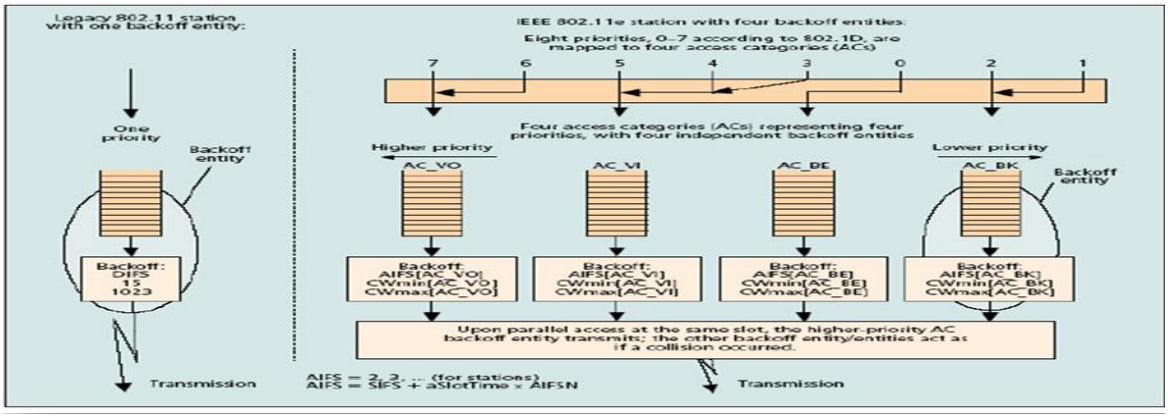

Figure 3.2 : Une station implémentant IEEE802.11 e

Les paramètres décrits ci-dessus sont annoncés par le point d'accès AP à travers des trames balises. Ce dernier peut alors les adapter aux conditions du réseau. La figure 3.2 illustre le mécanisme d'accès au support en mode EDCA.

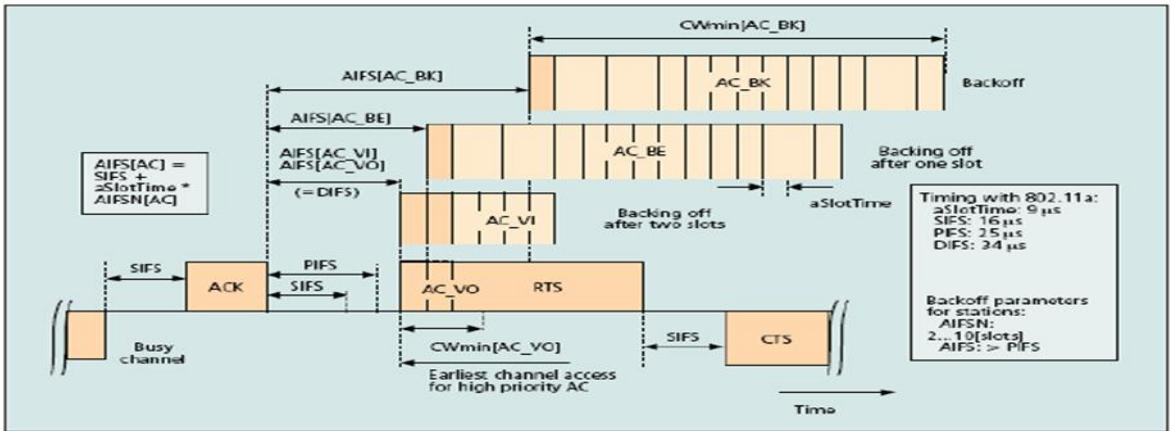

Figure 3.3 : L'accès en mode EDCA

Pour sa version actuelle, la norme 802.11e a aussi introduit le paramètre TXOP. Il s'agit d'un intervalle de temps pendant lequel une station a le droit d'émettre. Au niveau de la trame balise, l'AP annonce aussi à chaque AC la limite de l'intervalle TXOP (TXOPLimit [AC]) tout en définissant aussi la date de début de transmission. Durant un TXOP, la station peut transmettre plusieurs MPDUs pour un seul AC. Ces MPDUs sont espacés d'un SIFS de leurs acquittements. Cette transmission de plusieurs MPDUs est notée CFB. La figure 3.3 présente la structure du CFB :

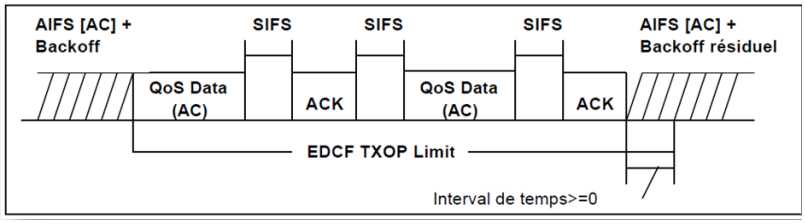

Figure 3. 4 Structure temporelle du CFB

- **La méthode d'accès HCCA:**

Le mécanisme d'accès HCCA combine les avantages des modes DCF et PCF. Il utilise un coordinateur central appelé HC qui utilise des règles différentes de celles du mode PCF. Avec HCCA, le TXOP est alloué par l'AP et peut-être actif à la fois dans la période sans contention (CFP) mais aussi dans la période avec contention (CP). En effet, il est possible de découper l'intervalle de temps CP en une nouvelle période sans contention appelée CAP qui utilise le mécanisme HCCA, et une période avec contention qui utilise EDCA. Les périodes CAP sont utiles pour rendre indépendante la fréquence d'émission des balises (beacons) des contraintes de latence que peuvent avoir les applications multimédias. D'autre part, pour remédier au phénomène de désynchronisation des beacons qui se produit avec le mode PCF avec HCCA, une station n'est autorisée à émettre un paquet que dans la mesure où sa transmission ne gène pas l'émission de la prochaine balise. Afin de garantir un service différencié, le mécanisme HCCA se base sur une négociation de trafic TSPEC entre le point d'accès et les stations. Avant de transmettre un flot qui nécessite une garantie de service, un circuit virtuel appelé TS doit s'établir entre l'AP et les différentes stations pour échanger certains paramètres (comme le débit du flot, la taille des paquets, la latence maximale acceptable, etc.) En fonction des paramètres TSPEC, un ordonnanceur localisé dans l'AP calcule une durée de TXOP pour chacune des stations. [8]

ii. Autres améliorations :

Le standard présente différents mécanismes, complémentaires à HCF, permettant d'offrir une QoS pour l'accès 802.11. L'essentiel de ces mécanismes fut introduit par la modification IEEE 802.11e. Nous en exposons certains dans ce paragraphe, celui qui nous intéresse étant essentiellement le protocole de lien direct.

- **Direct Link Protocol :**

Les spécifications de trafic dans le standard 802.11 original en mode AP ne permettent l'écoulement du trafic entre stations qu'en passant par l'AP uniquement.

Le protocole de liaison directe (DLP) dans la norme 802.11 e donne la possibilité aux stations d'envoyer le trafic directement entre elles sans traverser l'AP. Cette possibilité peut potentiellement augmenter la largeur de bande disponible pour la communication de station à station. Le DLP fonctionnera seulement quand les stations qui veulent communiquer sont dans la portée l'une de l'autre. Le DLP pourrait également augmenter potentiellement le temps d'exécution dans le cas où le lien entre les stations

qui communiquent est meilleur que le lien entre les stations et l'AP. Ceci a pu être le cas quand les stations sont plus près l'une de l'autre que de l'AP. Si après la durée « *DLPIdleTimeout* » il n'y a aucune transmission de trames entre les deux stations, le lien direct est coupé. [10]

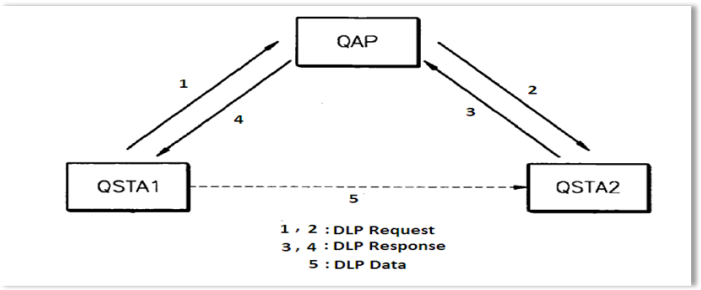

Figure 3.5 : Dialogue DLP

Avec le DLP, l'expéditeur envoie d'abord un message de demande de lien direct (DLP Request) au récepteur par le QAP. Une fois que le récepteur reconnait la demande, le lien direct entre les deux stations est établi.

Plusieurs formats de trames sont définis dans le but de la gestion du DLP citons : [11]

- **DLP Request :**

La trame DLP Request est utilisée pour l'établissement du lien direct entre deux stations dans un même BSS. Le corps de la trame DLP Request contient les informations mentionnées dans le tableau suivant :

Order	Information
1	Category
2	Action
3	Destination MAC Address
4	Source MAC Address
5	Capability Information
6	DLP Timeout Value
7	Supported rates

Tableau 3.1: Corps de la trame DLP Request

- **Category :**

Le tableau suivant représente les codes des catégories ainsi avec la signification :

Code	Signification
1	QOS
2	DLP
3	Block ACK

Tableau 3.2 : Les codes du champ « Category »

Dans la trame DLP Request la valeur du « Category » vaut **2** (représentant le DLP).

- **Action :**

Les différentes valeurs du champ « Action » avec leurs significations sont présentées dans le tableau suivant :

Code	Signification
0	DLP Request
1	DLP Response
2	DLP Teardown

Tableau 3.3 : Les codes du champ « Action »

Dans la trame DLP Request la valeur du champ « Action » vaut **0** (représentant le DLP Request).

- **Destination MAC Adress :**

Représente l'adresse MAC de la station destination.

- **Source MAC Adress :**

Représente l'adresse MAC de la station émettrice.

- **Capability information :**

Informations sur la capacité de la station émettrice de la demande.

- **DLP Timeout Value :**

Champ utilisé pour indiquer la valeur du temps de mort du lien direct, la longueur de cette valeur est 2 octets. Ce champ contient la durée en seconde après laquelle le lien direct est terminé s'il n'y a aucune trame échangée entre les deux QSTAs.

- **Supported rates :**

Contient les informations de taux de charge de la station émettrice.

- **DLP Response :**

La trame DLP Response est envoyée comme réponse à une trame DLP Request. Le corps de la trame DLP Response contient les informations mentionnées dans le tableau suivant :

Order	Information
1	Category
2	Action
3	Status Code
4	Destination MAC Address
5	Source MAC Address
6	Capability Information
7	Supported rates

Tableau 3.4 : Corps de la trame DLP Request

- **Category :**

Dans la trame DLP Response la valeur du « Category » vaut **2** (représentant le DLP).

- **Action :**

Dans la trame DLP Response la valeur du champ « Action » vaut **1** (représentant le DLP Response).

- **Status code :**

Les différentes valeurs du champ « Status » avec leurs significations sont présentées dans le tableau suivant :

Code	Signification
0	Établissement du lien direct avec Succès.
32	Échec non précisée.
33	Association (avec le QBSS) refusée car le QAP n'a pas une bande passante suffisante pour traiter une autre QSTA.
34	Association (avec le QBSS) refusée du a un taux de perte de trame excessif
35	Association (avec le QBSS) refusée car la station demandée ne supporte pas la QOS.
37	La demande a été refusée
38	La demande n'a pas été couronnée de succès, un ou plusieurs paramètres ont des valeurs invalides.
39	Le TS n'a pas été créé car la demande ne peut pas être honorée. Cependant une TSPEC suggéré est prévu pour que la QSTA source puisse tenter d'établir un autre TS avec les modifications proposées à la TSPEC.
40	Le TS n'a pas été créé car la demande ne peut pas être honorée.

	Cependant le HC peut être en mesure de créer un TS en réponse à une demande après le temps indiqué dans l'élément de retard TS.
41	Le lien direct n'est pas autorisé dans ce BSS
42	La station destination n'est pas présente dans le même QBSS
43	La station destination n'est pas une QSTA

Tableau 3.5 : Les codes du champ « Status »

- **Destination MAC Adress and the source Mac Adress:**

Elles sont copiées du champ correspondant dans la trame DLP Request.

- **Capability information :**

Informations sur la capacité de la station destination. Cette information est incluse seulement dans le cas où la valeur du « DLP Status Code » vaut **0** (Succès).

- **Supported rates :**

Contient les informations de taux de charge de la station destination. Cette information est incluse seulement dans le cas où la valeur du « DLP Status Code » vaut **0** (Succès).

- **DLP Teardown :**

La trame DLP Teardown est envoyée pour terminer le lien direct avec. Le corps de la trame DLP Teardown contient les informations mentionnées dans le tableau suivant :

Order	Information
1	Category
2	Action
3	Destination MAC Address
4	Source MAC Address

Tableau 3.6 : Corps de la trame DLP Teardown

- **Category :**

Dans la trame DLP Teardown la valeur du « Category » vaut **2** (représentant le DLP).

- **Action :**

Dans la trame DLP Teardown la valeur du champ « Action » vaut **2** (représentant le DLP Teardown).

- **Destination MAC Adress :**

Représente l'adresse MAC de la station destination.

- **Source MAC Adress :**

Représente l'adresse MAC de la station émettrice.

- **Block ACK (acquittement groupé):**

Cette procédure optionnelle permet d'améliorer l'utilisation du médium. En effet, elle permet à une station d'envoyer plusieurs paquets sans que ceux-là soient acquittés individuellement. Le bloc de paquets pourra être acquitté à la fin de l'envoi du bloc ou dans un TXOP ultérieur. L'utilisation du réseau s'en trouve ainsi améliorée. [9]

- **Le contrôle d'admission**

Un cadre pour le contrôle d'admission a été mis en place par la modification 802.11e. Ce cadre concerne l'accès par HCF avec ou sans contention (par EDCA ou par HCCA). Le contrôle d'admission servira à la gestion et la régulation de la bande passante disponible. Une QSTA souhaitant avoir des garanties de QoS (sur les délais d'accès, sur les débits ou sur le taux de pertes par exemple) devra passer par le contrôle d'admission. Les algorithmes de contrôle d'admission ne sont pas définis par le standard, le choix de l'algorithme utilisé est laissé à l'équipementier. Le standard définit cependant un cadre et un certain nombre de règles que les algorithmes devront respecter. [9]

- **NoAck**

Permet la mise en place de classes de services avec lesquels les messages transmis ne sont pas acquittés. Cette amélioration permet d'éviter la retransmission inutile de données à haute criticité temporelle. [6]

- **Respect des échéances**

D'autres modifications ont été introduites par IEEE 802.11e permettant d'améliorer le respect des échéances en contraignant les durées d'accès des stations :

- Les stations utilisant le médium sont contraintes de respecter le TBTT annoncé par le paquet Beacon. Une station voulant accéder au médium doit vérifier que la transmission entamée (jusqu'à la réception éventuelle du ACK) ne doit pas dépasser le TBTT annoncé. Le CFP ne sera, par conséquent, pas retardé par les stations accédant au médium.
- Chaque accès au médium se fait dans la limite de l'opportunité de transmission (TXOP) accordée. La valeur du TXOP est fixée par le HC. [9]

III.3.2 Les mécanismes de qualité de service niveau IP : IntServ / DiffServ

i. Le protocole à intégration de service IntServ :

Les applications traditionnelles non temps réels comme FTP se sont longtemps satisfaites du service best effort. Mais avec l'arrivée des communications multimédias, de nombreuses applications sont devenues sensibles au délai si bien que le service best effort traditionnel ne suffit plus. Bien que certaines applications soient adaptatives, il est souvent nécessaire de fournir de nouvelles classes de service offrant une meilleure qualité de service (en termes de bande passante, délai ou pertes). Ces nouvelles classes de service s'ajoutent au best effort traditionnel pour créer un Internet à intégration de services.

Un mécanisme explicite est utilisé pour signaler les exigences de qualité de service par flot aux éléments du réseau (hôtes, routeurs ou sous-réseaux). Les éléments du réseau, selon les ressources disponibles, implémentent l'un des services IntServ en fonction du type de qualité de service souhaité pendant la transmission des données. Le modèle distingue plusieurs types de services, en fonction du délai de transit par paquet **(**Service à contrôle de charge (CL)**)**, Service garanti (GS)**).**

L'architecture IntServ repose sur deux principes fondamentaux :

- le réseau doit être contrôlé et soumis à des mécanismes de contrôle d'admission,
- des mécanismes de réservation de ressources sont nécessaires pour fournir des services différenciés.

Le modèle IntServ définit une architecture capable de prendre en charge la qualité de service en définissant des mécanismes de contrôle complémentaires sans toucher au fonctionnement IP. C'est un modèle basé sur un protocole de signalisation RSVP. Dans le modèle présenté par, les routeurs réservent les ressources pour un flot de données spécifiques en mémorisant des informations d'état. Il est important de rafraîchir périodiquement les informations au cas où il y a eu changement de la route emprunté par le flot. En effet, il est inutile de continuer à réserver les ressources sur un routeur qui ne fait plus partie du chemin emprunté. Au niveau technique, la faiblesse principale de l'architecture IntServ est sa non-résistance au facteur d'échelle. Le nombre de flux qui peuvent bénéficier d'une réservation est assez limité, en particulier dans les routeurs du cœur du réseau. Ces équipements doivent traiter des milliers des flux simultanément, et le coût introduit par la gestion d'états et l'ordonnancement par flux peut entraîner une réduction considérable de leur performance. **[**7]

Rapport-Gratuit.com

ii. Le protocole à différentiation de service DiffServ :

L'approche DiffServ est souvent comparée à celle de IntServ pour sa capacité à être déployée sur de « grands réseaux » ; alors que IntServ, de par le traitement par flots, ne peut s'appliquer que sur des réseaux de « petite taille », DiffServ réduit au maximum la taille des tables en considérant des agrégations de flots. Ainsi, les flots ne sont pas trait´es individuellement mais par agrégats, ce qui allège considérablement la charge des routeurs du réseau. De plus, le contrôle d'admission n'est plus assuré individuellement par chaque routeur travers´e, mais par les routeurs de bordures, rendant DiffServ beaucoup plus adapté aux grands réseaux, et notamment aux réseaux d'opérateurs.

Le niveau de QoS d'un flot est indiqué dans un champ de l'en-tête de ses paquets. Lorsqu'un routeur de bordure décide d'admettre un nouveau trafic dans le réseau, il fixe la valeur du champ DSCP dans l'en-tête IP :

• le code 0 signifie que le paquet doit être trait´e en Best-Effort, après tous les autres (niveau le moins prioritaire),

• un code autre que 0 aura une autre signification ; cette correspondance est fixée par l'opérateur lui-même, selon les contrats qu'il propose à ses clients.

Lorsqu'un routeur du cœur du réseau devra traiter ce paquet, il devra inspecter le champ DSCP et traiter le paquet en conséquence. Ainsi, DiffServ présente l'avantage de ne pas nécessiter de signalisation puisque le décodage du champ DSCP se fait via des tables inscrites dans la mémoire du routeur. Le contrôle de congestion se fait directement par les routeurs de bordure, selon le principe de la capacité finie et connue du réseau. Dans le cœur du réseau, il n'y a pas de réservation de ressources, seule la différentiation de traitement des paquets suffit à appliquer une qualité de service pour la traversée du paquet dans le réseau. Cependant, afin de palier les pertes sur le médium, le réseau cœur, par exemple un réseau ATM, nécessite d'être légèrement surdimensionné. [6]

IV. Conclusion :

Dans ce chapitre, nous avons défini la problématique de la qualité de service dans les réseaux sans fil. Nous avons ensuite présenté un ensemble de solutions de qualité de service apportées à ces technologies. Les améliorations apportées à la technologie 802.11 ont été détaillées.

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.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

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.

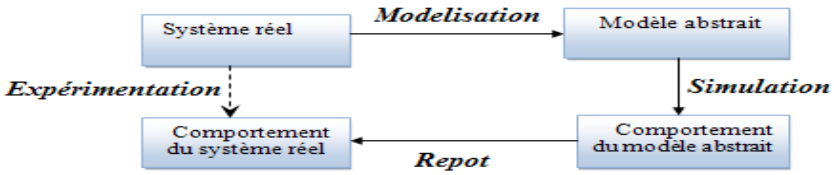

Figure 4.1 : Cycle modélisation-simulation

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.

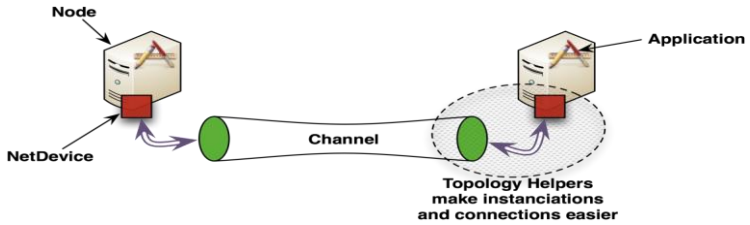

Figure 4.2 : Établissement d'une connexion entre deux nœuds

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.

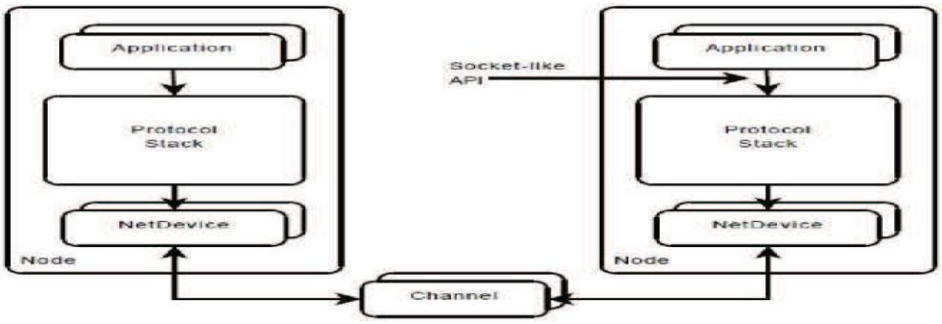

Figure 4.3 : Architecture du nœud NS-3

- ***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.

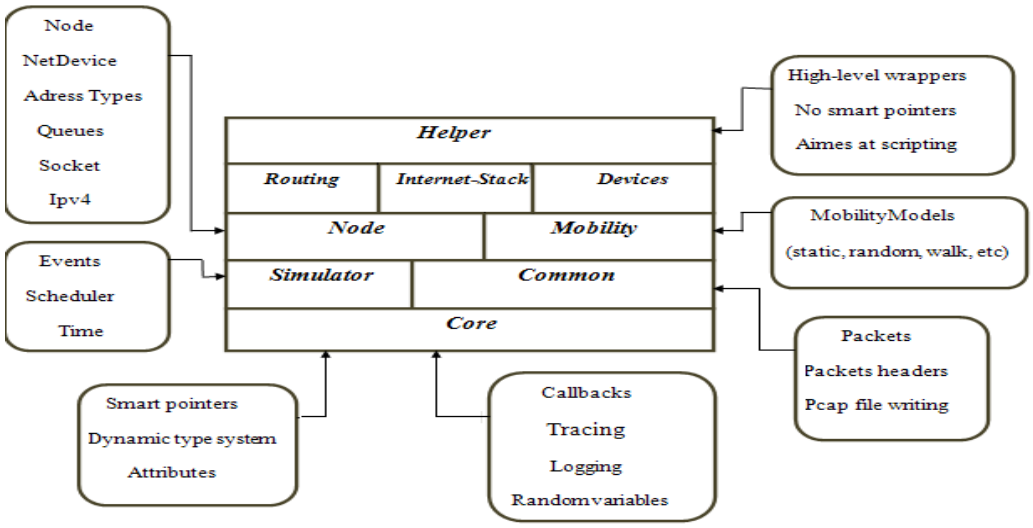

Figure 4.4:Les principaux modules de NS3

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**

Rapport-Gratuit.com

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.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

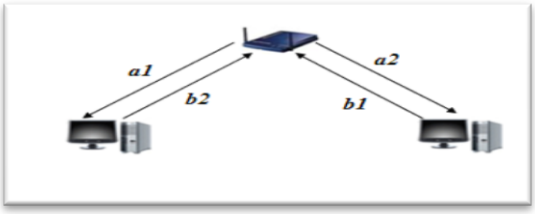

Figure 4.6 : Réseau wifi test

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.

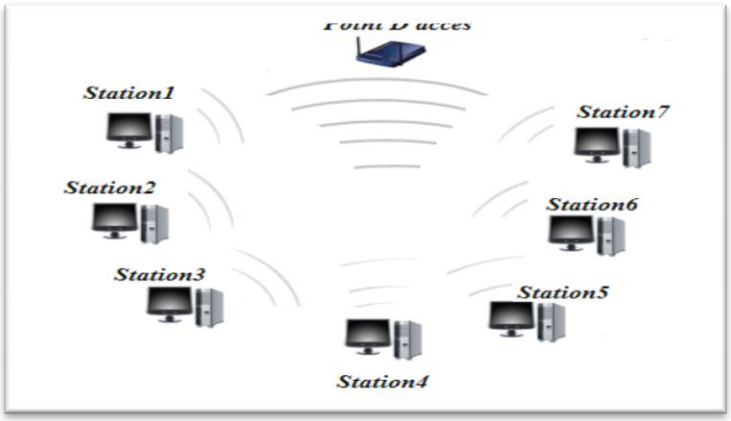

Figure 4.7 : Augmentation du nombre des stations

- ***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).

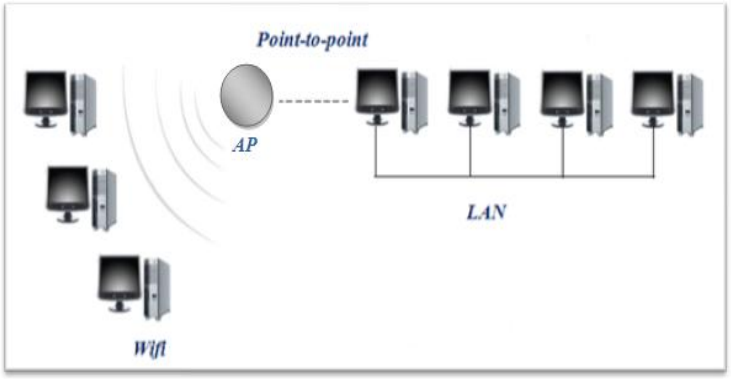

Figure 4.8 : Topologie réseau mixte test

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 :**

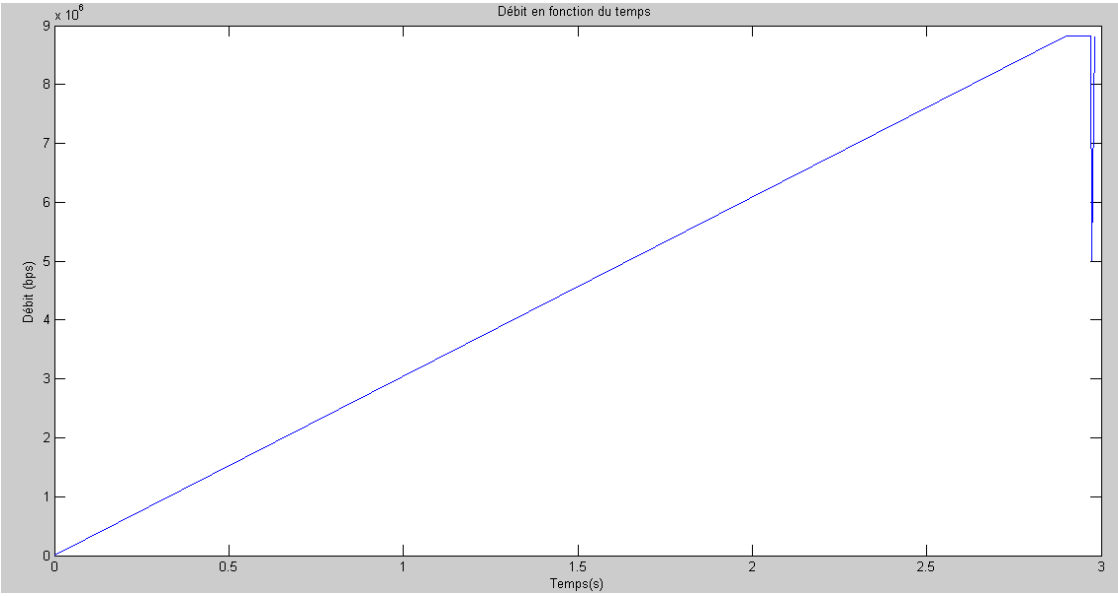

Figure 4.9 : Débit sans DLP

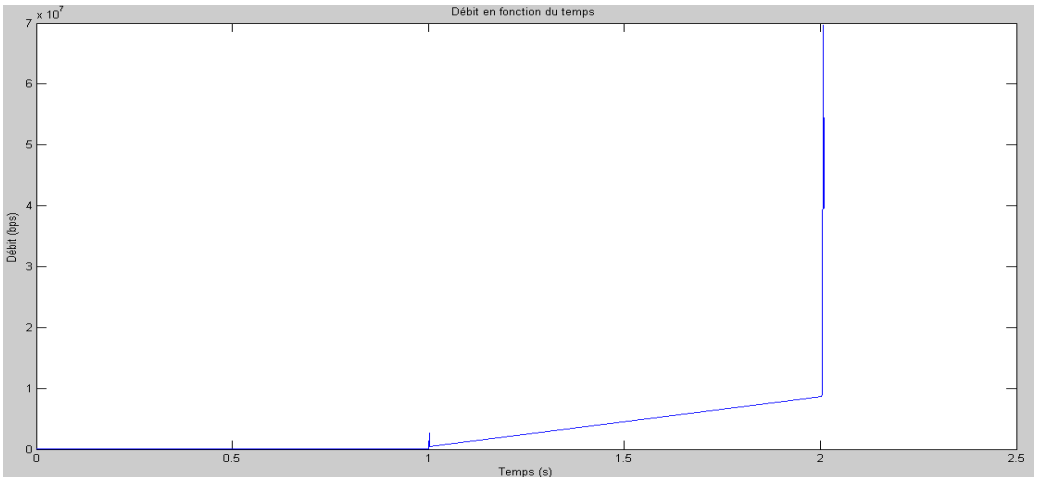

Figure 4.10 : débit avec DLP

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

- **Délai :**

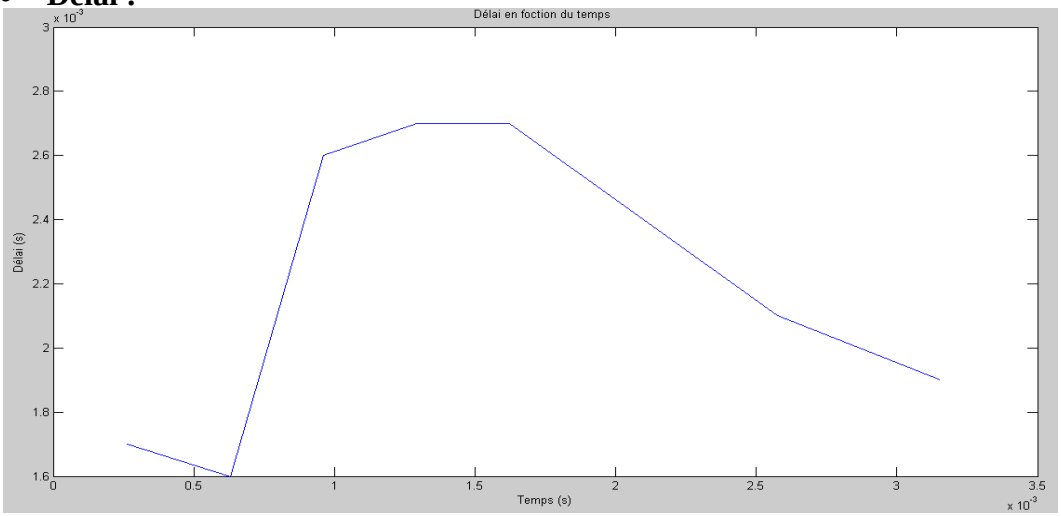

Figure 4.11: Délai sans DLP

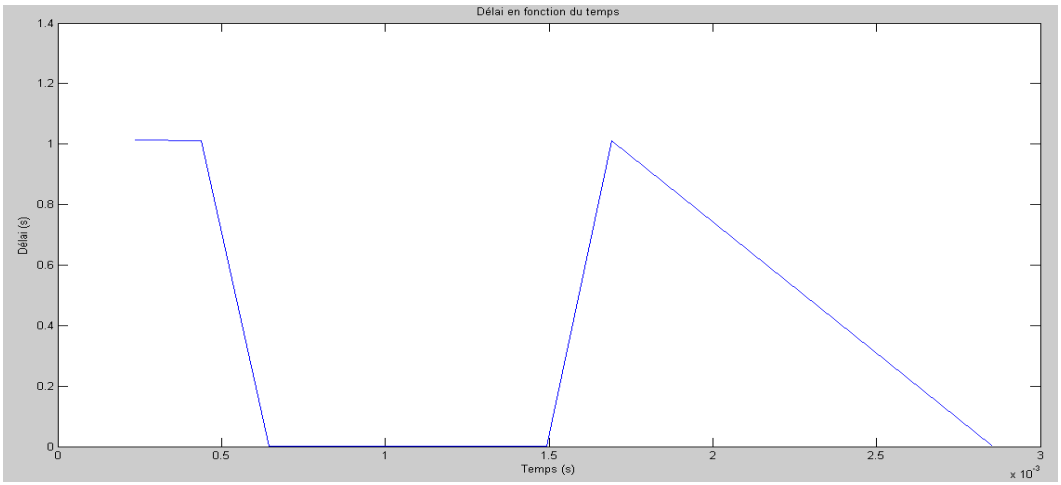

Figure 4.12: Délai avec DLP

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 :**

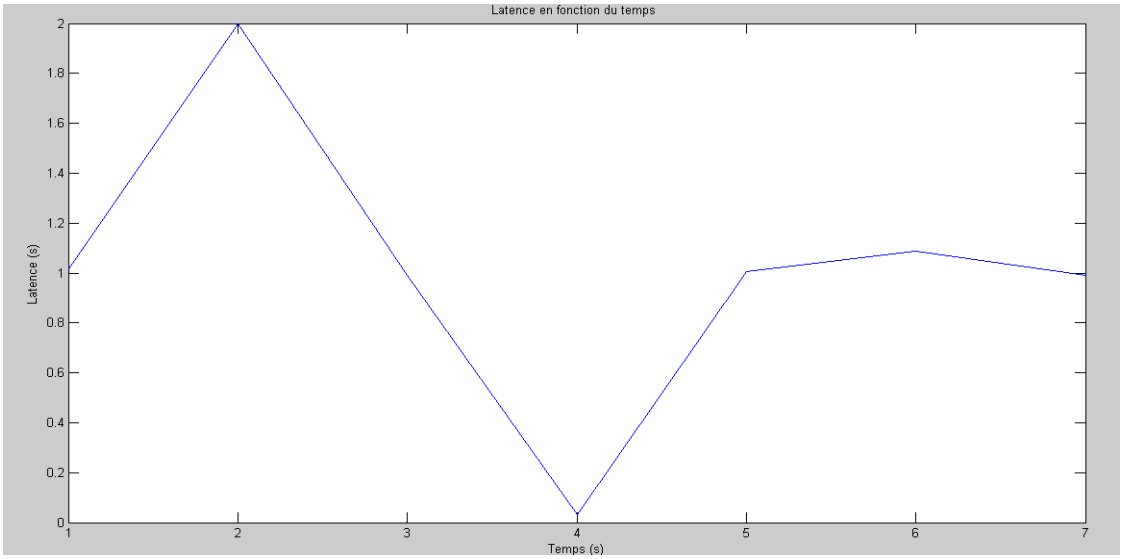

Figure 4.13 : Latence sans DLP

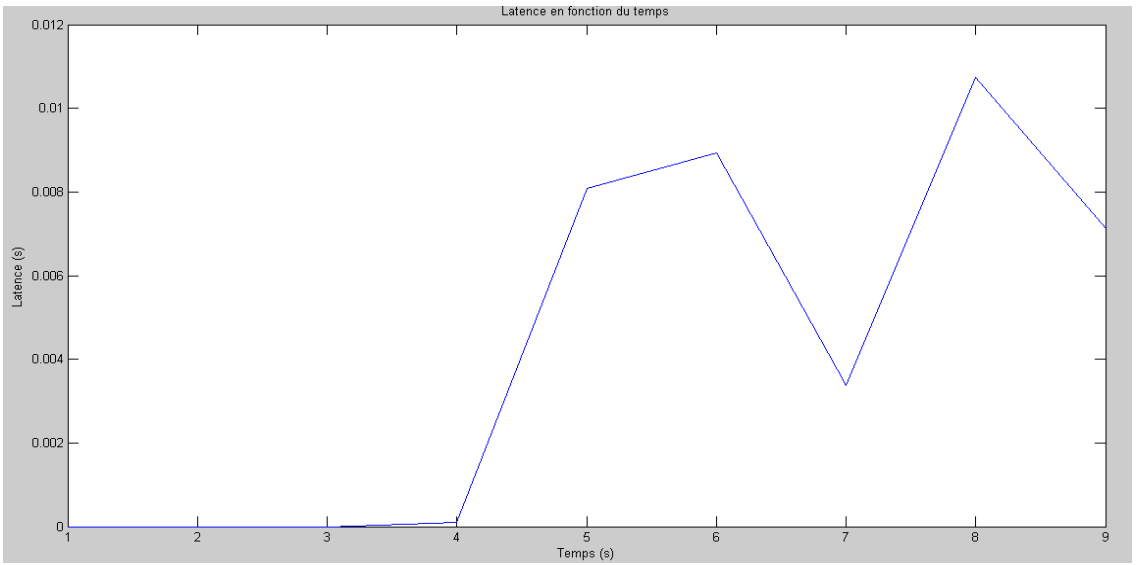

Figure 4.14: Latence avec DLP

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 :**

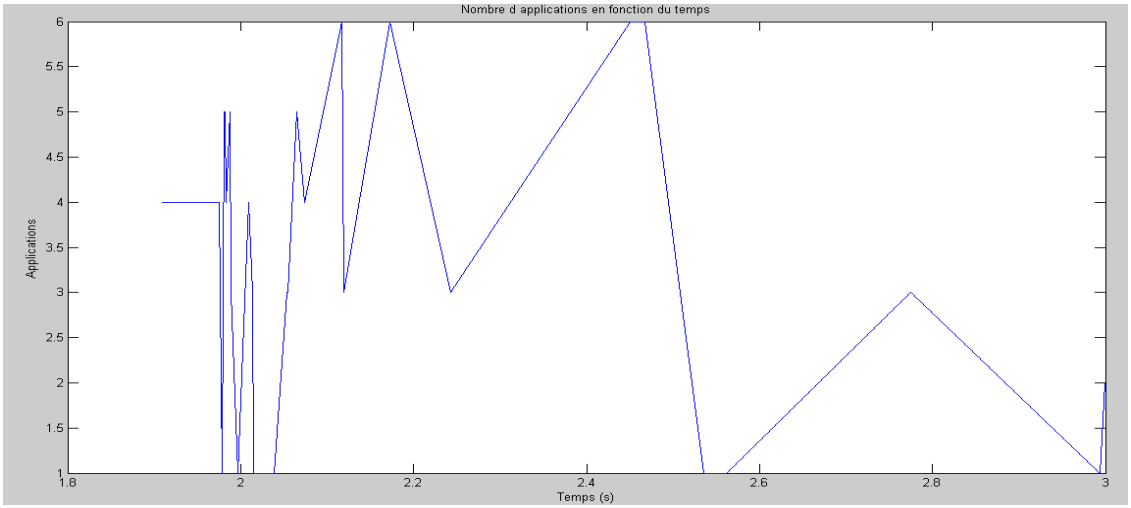

Figure 4.15: Nombres d'applications sans DLP

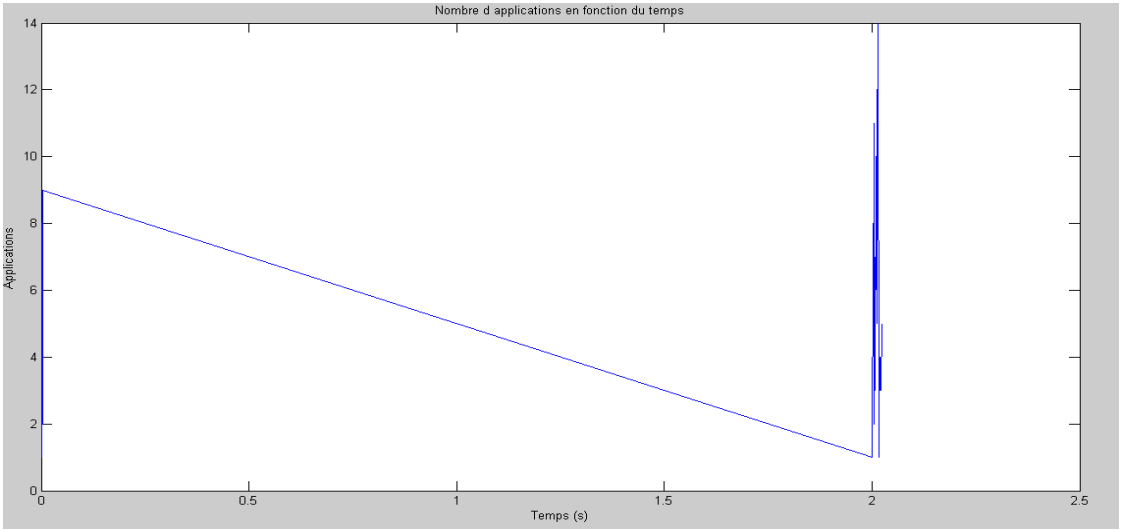

Figure 4.16 : Nombres d'applications avec DLP

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 :**

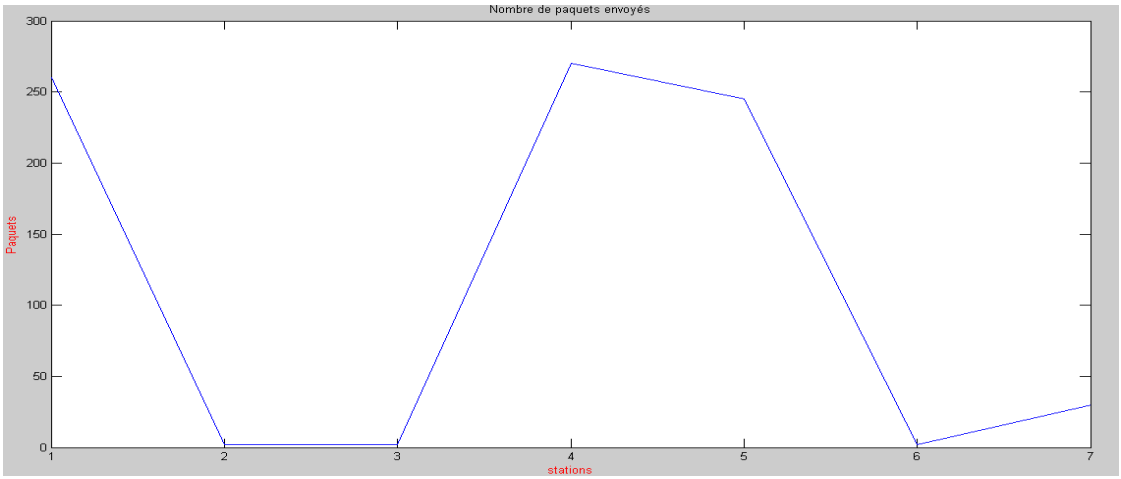

Figure 4.17: Nombre de paquets envoyés sans DLP

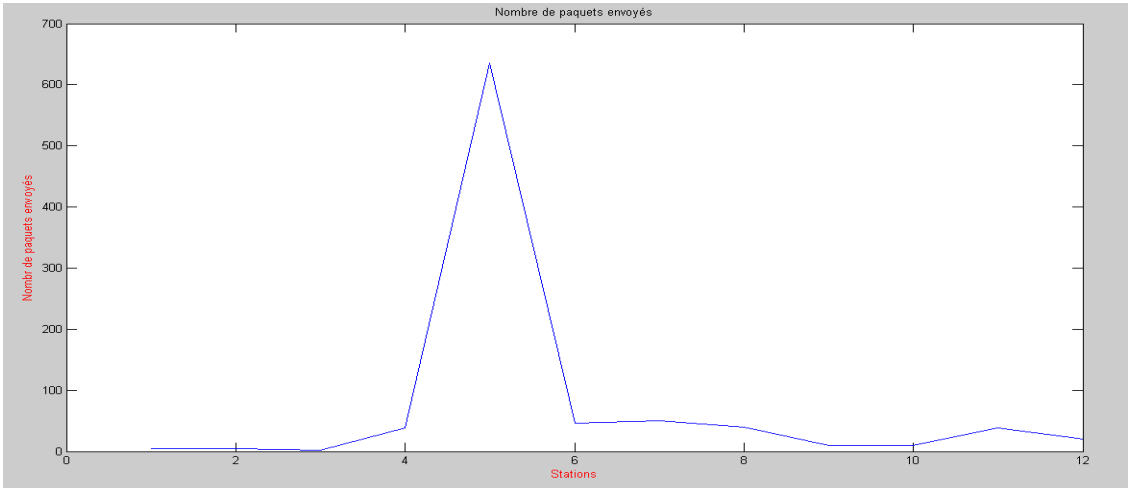

Figure 4.18: Nombre de paquets envoyés avec DLP

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.

Conclusion générale

L'objectif de ce mémoire, était d'étudier les problèmes liés à la gestion de la qualité de service dans des environnements sans fil.

IEEE 802.11 est la norme des réseaux locaux sans fil la plus déployée mais elle ne garantit pas une qualité de service suffisante pour les différents types de service. De ce fait, le groupe de travail 802.11 a mis en œuvre une nouvelle version de la norme 802.11 à savoir la norme 802.11e qui a pour but de garantir la qualité de service pour les utilisateurs et pour les applications multimédias.

Dans ce contexte, l'introduction du protocole DLP au niveau MAC afin d'améliorer la qualité de service s'avère importante. Notre but s'articule autour de cette analyse des problèmes de garantie de qualité de service dans les réseaux 802.11 et de l'apport de la norme 802.11e.

Un tel service est efficace dans la mesure où les utilisateurs choisissent de manière pertinente la classe de services en fonction des caractéristiques de l'application. Une solution est de jumeler le protocole de contrôle d'accès au médium avec un mécanisme de contrôle des flots émis par les différentes stations. Il serait alors possible de s'assurer que les services à forte contrainte de latence sont bien utilisés par des applications multimédias. Ce type de vérification, complète certaines études récentes visant à détecter de mauvais comportements des utilisateurs dans les réseaux wifi ou à estimer les ressources des WLAN consommées par chaque utilisateur. Les protocoles de qualité de services devront donc s'efforcer d'évaluer au mieux l'état du réseau afin de ne pas sur ou sous-évaluer sa capacité mais devrons aussi être adaptatif afin de réagir rapidement et efficacement à la mobilité de ces réseaux. Dans notre travail, on a étudié un nouveau mécanisme de qualité de service le DLP qui consiste à donner la possibilité aux stations d'un réseau infrastructure d'envoyer le trafic directement entre elles sans traverser l'AP. Cette possibilité peut potentiellement augmenter la largeur de bande disponible pour la communication de station à station. Puis on a comparé en utilisant la simulation le comportement d'un réseau infrastructure sans DLP et un réseau avec la présence du protocole DLP et on a remarqué l'efficacité de ce protocole, puisque les performances du réseau s'améliorent nettement. Donc on aurait tendance à conseiller aux constructeurs d'implémenter ce mécanisme dans les AP, car il a une nette influence sur les performances globales du réseau et des paramètres de QoS des trafics temps réel en particulier.

REFERENCES BIBLIOGRAPHIQUES

[1] : http://www.commentcamarche.net/contents/wireless/wlintro.php3

[2] : http://www.urec.cnrs.fr/IMG/pdf/cours.sf.pdf

[3] : http://www.commentcamarche.net/faq/3020-wifi-cours-d-introduction

[4] : http://www.commentcamarche.net/contents/wireless/wlintro.php3

[5] : http://www.commentcamarche.net/contents/wireless/wpan.php3

[6] : http://www.commentcamarche.net/contents/wireless/wlan.php3

[7] : http://www.commentcamarche.net/contents/wifi/wifiintro.php3

[8] : http://www.commentcamarche.net/contents/wireless/wman.php3

[9] : http://www.commentcamarche.net/contents/wireless/wwan.php3

[10] : http://www.commentcamarche.net/contents/telephonie-mobile/gsm.php3

[11] : http://www.commentcamarche.net/contents/telephonie-mobile/gprs.php3

[12] : Master M2 Pro Sir ; Thierry Bohbot : Les réseaux sans fil et les problèmes des sécurités ; Professeur : BERNARD TOURANCHEAU ANNEE : 2009-2010 (PDF).

[13] : Adlen KSENTINI ; 2005- « qualité de service dans les réseaux locaux sans fil basés sur la technologie IEEE 802.11 » ; THESE PRESENTE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITE DE CERGY-PONTOISE.

[14] : Adrien VAN DEN BOSSCHE ; 2007- « Proposition d'une nouvelle méthode d'accès déterministe pour un réseau personnel sans fil à fortes contraintes temporelles » ; MEMOIRE DE THESE, DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE II, Ecole Doctorale Systèmes.

[15] : Michel Terré ; Mars 2007- «Le Standard 802.11, Couche physique et couche MAC» ; en ligne : http://stic.cnam.fr/elau/publi/terre/images/WiFi.pdf .

[16] : Rabih MOAWAD ; Décembre 2004 - « QoS dans les WPAN, WLAN et WMAN» ; MEMOIRE DE DEA RESEAUX ET TELECOMMUNICATIONS.

[17] : Madjid Naziha ; 2008- « étude de la qualité de service dans les réseaux 802.11e » ; MEMOIRE DE FIN D'ETUDE POUR L'OBTENTION DU DIPLOME D'INGENIEUR D'ETAT EN TELECOMMUNICATIONS.

[18] : Diarra Charles Drissa & Goita Oumar ; 2009- « étude de la qualité de service dans les réseaux Wifi » ; MEMOIRE DE FIN D'ETUDE POUR L'OBTENTION DU DIPLOME D'INGENIEUR D'ETAT EN TELECOMMUNICATIONS.

Rapport-Gratuit.com

[19] : NEHDI Mourad ; 2004- « Evaluation de protocole EDCA » ; PROJET DE FIN D'ETUDE.

[20] : Michel Duchateau : 2004- « Analyse et simulation du déploiement d'un réseau sans fil à l'ULB » ; MEMOIRE DE FIN d'ETUDE PRESENTE EN VUE DE L'OBTENTION DU GRADE D'INGENIEUR CIVIL ELECTRICIEN SPECIALITE EN TELECOMMUNICATION.

[21] : CERT ; « Wireless Fidelity » ; république tunisienne, ministère des technologies de la communication, Centre d études et de recherche des télécommunications.

[22] : Najet Tibi-2004 « Etude des mécanismes de différentiation de service niveau IP et Mac » ; RAPPORT DE PROJET DE FIN D'ETUDES.

[23] : http://www.pouf.org/documentation/securite/html/node9.html

[24] : http://www.pouf.org/documentation/securite/html/node17.html

[25] : http://docinsa.insa-lyon.fr/these/2009/ben_hamida/these.pdf

[26] : http://www.simutools.org/2008/ns3.shtml

[27] : http://www1.cse.wustl.edu/~jain/cse567-08/ftp/simtools/index.html

[28] : Network Simulator-3: A Review :Ashish Mohta, Sonali I. Ajankar, M. M. Chandane ; Department of Computer Technology Veermata Jijabai Technological Institute ; Mumbai, India

[29]: Mme Sedjelmaci Amina Nadjet : Extention de la QoS du Wifi vers le Wimax.

[30] :https://juboite.hd.free.fr/doku.php?id=tuto:tools:network:simulation:ns:ns3:models:wifi

[31]: Design and Development of a Simulation Toolbox for ; Content Sharing Applications in Ad Hoc Networks _Shahriar Kaisar : Dept. of Computer Science ; University of Saskatchewan ; Saskatoon,Canada

[32]: http://www.nsnam.org/docs/release/3.10/manual/html/wifi.html

ANNEXE

L'implémentation de la simulation est centrée autour des scripts NS-3 suivants:

X.1 Script 1 :

- Nous avons créé un réseau infrastructure simple (2 stations et un point d'accès avec QOS)
- Nous remarquons que le point d accès s'intervient dans toutes les communications appart une (**la dernière application**) cela indique son **saturation** donc dans ce cas là le point d'accès ne supporte que ***six clients.***

Voici le code source du Script 1 :

```
#include "ns3/core-module.h"
#include "ns3/common-module.h"
#include "ns3/node-module.h"
#include "ns3/helper-module.h"
#include "ns3/mobility-module.h"
#include "ns3/contrib-module.h"
#include "ns3/wifi-module.h"
#include "ns3/athstats-helper.h"
#include <iostream>
using namespace ns3;
int main (int argc, char *argv[])
{
 Packet::EnablePrinting ();
```

- **Activation du rts cts tout le temps**

```
  Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue ("0"));
```

- **Désactivation de la fragmentation**

```
  Config::SetDefault ("ns3::WifiRemoteStationManager::FragmentationThreshold", StringValue ("2200"));
```

- **Configuration par défaut de WifiHelper qui permettra de créer et d'installer les WifiNetDevice sur un ou plusieurs nœuds.**

```
 WifiHelper wifi = WifiHelper::Default ();
```

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

- **Configuration d un modèle de la mobilité:**

MobilityHelper mobility;

- **Instancier le point d'accès et les stations Wifi (dans des *NodeContainer*)**

NodeContainer stas;

NodeContainer ap;

NetDeviceContainer staDevs;

- **Création et configuration d une interface de communication (Un *Socket* dans notre cas pour les applications *OnOffApplication*)**

PacketSocketHelper packetSocket;

- **Création de deux stations et un point d'accès**

stas.Create (2);

ap.Create (1);

- **Installation du *socket* dans les nœuds**

packetSocket.Install (stas);

packetSocket.Install (ap);

- **Configuration du modèle de la couche Mac**

QosWifiMacHelper wifiMac = QosWifiMacHelper::Default ();

- **Configuration du modèle de la couche physique qui sera installé dans les *WifiNetDevice* en utilisant le *YansWifiPhyHelper***

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();

- **Configuration et instanciation du modèle de canal Wifi en utilisant *YansWifiChannelHelper***

YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();

- **Association du canal précédemment créé à la couche physique**

wifiPhy.SetChannel (wifiChannel.Create ());

Ssid ssid = Ssid ("wifi-default");

wifi.SetRemoteStationManager ("ns3::ArfWifiManager");

- **Pour les stations, le modèle *StaWifiMac* doit être utilisé, et l'"*ActiveProbing*" doit être désactivé**

```
wifiMac.SetType ("ns3::StaWifiMac",
                 "Ssid", SsidValue (ssid),
                 "ActiveProbing", BooleanValue (false), “QosSupported”, BooleanValue (true));
  staDevs = wifi.Install (wifiPhy, wifiMac, stas);
```

- **Pour le point d'accès, le modèle *ApWifiMac* doit être utilisé, et la génération de *beacons* (à une fréquence configurable) doit être activée**

```
 wifiMac.SetType ("ns3::ApWifiMac", "Ssid", SsidValue (ssid), "BeaconGeneration", BooleanValue(true), "BeaconInterval",TimeValue (Second (2.5)), “QosSupported”, BooleanValue (true));
  wifi.Install (wifiPhy, wifiMac, ap);
```

- **Installation du modèle de mobilité crée précédemment au niveau des stations et du point d accès**

```
  mobility.Install (stas);
  mobility.Install (ap);
```

- **Installation d’une application de type *OnOffapplication***

```
  PacketSocketAddress socket;
  socket.SetSingleDevice(staDevs.Get (1)->GetIfIndex ());
  socket.SetPhysicalAddress (staDevs.Get (0)->GetAddress ());
  socket.SetProtocol (1);
  OnOffHelper onoff ("ns3::PacketSocketFactory", Address (socket));
  onoff.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1)));
  onoff.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));
 ApplicationContainer apps = onoff.Install (stas.Get (1));
  apps.Start (Seconds (2.99));
  apps.Stop (Seconds (3.0));
  Simulator::Stop (Seconds (3.0));
```

- **Activation du traçage *PCAP***

```
  wifiPhy.EnablePcapAll ("script1");
```

- **Début de la simulation**

```
  Simulator::Run ();
```

- **Fin de la simulation**

```
Simulator::Destroy ();
return 0;
}.
```

X.2 Script 2 :

- Réseau maillé : Se réseau est composé d'un :
  - ✓ Réseau point à point avec deux nœuds.
  - ✓ Réseau LAN dont son premier nœud est le deuxième nœud du réseau point à point.
  - ✓ Réseau wifi avec infrastructure dont son point d'accès est le premier nœud du réseau point à point (définition d un protocole ipv4 et d'adressage afin de garantir que les stations wifi soient dans le même BSS).

Voici le code source du Script 2 :

```
#include "ns3/core-module.h"
#include "ns3/simulator-module.h"
#include "ns3/node-module.h"
#include "ns3/helper-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/ipv4-global-routing-helper.h"
using namespace ns3;
void
Tx (std::string context, Ptr<Packet const> packet)
{
    NS_LOG_UNCOND (context <<
                   "\tSend " <<
                   "\tTime " << Simulator::Now().GetSeconds());
}
void
Rx (std::string context, Ptr<Packet const> packet, Address const& add)
{
    NS_LOG_UNCOND (context <<
```

```
                    "\tReceive " <<
                    "\tTime " << Simulator::Now().GetSeconds());
}
NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");
int
main (int argc, char *argv[])
{
  bool verbose = true;
  uint32_t nCsma = 3;
  uint32_t nWifi = 3;
 double SimTime = 50;
  CommandLine cmd;
  cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
cmd.Parse (argc,argv);
  if (verbose)
    {
      LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
      LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }
  NodeContainer p2pNodes;
  p2pNodes.Create (2);
 PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
  NetDeviceContainer p2pDevices;
  p2pDevices = pointToPoint.Install (p2pNodes);
  NodeContainer csmaNodes;
  csmaNodes.Add (p2pNodes.Get (1));
  csmaNodes.Create (nCsma);
  CsmaHelper csma;
  csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
```

```
  csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
  NetDeviceContainer csmaDevices;
  csmaDevices = csma.Install (csmaNodes);
NodeContainer wifiStaNodes;
  wifiStaNodes.Create (nWifi);
  NodeContainer wifiApNode = p2pNodes.Get (0);
  YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
  YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
  phy.SetChannel (channel.Create ());
  WifiHelper wifi = WifiHelper::Default ();
  wifi.SetRemoteStationManager ("ns3::AarfWifiManager");
  QosWifiMacHelper mac = QosWifiMacHelper::Default ();
  Ssid ssid = Ssid ("ns-3-ssid");
  mac.SetType ("ns3::StaWifiMac",
               "Ssid", SsidValue (ssid),
               "ActiveProbing", BooleanValue (false));
  NetDeviceContainer staDevices;
  staDevices = wifi.Install (phy, mac, wifiStaNodes);
  mac.SetType ("ns3::ApWifiMac",
               "Ssid", SsidValue (ssid));
 NetDeviceContainer apDevices;
  apDevices = wifi.Install (phy, mac, wifiApNode);
 NetDeviceContainer infradevices (apDevices,staDevices);
  MobilityHelper mobility;
  mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                                 "MinX", DoubleValue (0.0),
                                 "MinY", DoubleValue (0.0),
                                 "DeltaX", DoubleValue (5.0),
                                 "DeltaY", DoubleValue (10.0),
                                 "GridWidth", UintegerValue (3),
                                 "LayoutType", StringValue ("RowFirst"));
 mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                            "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
```

```
mobility.Install (wifiStaNodes);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);
InternetStackHelper stack;
stack.Install (csmaNodes);
stack.Install (wifiApNode);
stack.Install (wifiStaNodes);
```

- Distribution des adresses IPv4

```
Ipv4AddressHelper address;
```

- Distribution des adresses IPv4 dans le réseau point à point

```
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);
```

- Distribution des adresses IPv4 dans le réseau LAN

```
address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);
```

- Distribution des adresses IPv4 dans le réseau wifi

```
address.SetBase ("10.1.3.0", "255.255.255.0");
```

- Récupération des adresses des objets IPv4Interfaces

```
Ipv4InterfaceContainer staswifiInterfaces = address.Assign (staDevices);
Ipv4InterfaceContainer apwifiInterfaces = address.Assign (apDevices);
Ipv4InterfaceContainer wifiInterfaces = address.Assign (infradevices);
```

- Création des applications de type Client/Serveur

```
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (3.0));
UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
```

```
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps =
    echoClient.Install (wifiStaNodes.Get (nWifi - 1));
  clientApps.Start (Seconds (1.0));
  clientApps.Stop (Seconds (3.0));
```

- Configuration de routage

```
 Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

- Création des applications de type ***OnOffApplication***

```
 uint16_t port1 = 9;
        OnOffHelper onoff1 ("ns3::UdpSocketFactory", Address());
        ApplicationContainer apps_onoff1;
        PacketSinkHelper sink1 ("ns3::UdpSocketFactory", Address ());
        ApplicationContainer apps_sink1;
        onoff1.SetAttribute ("DataRate", DataRateValue(4000));
        onoff1.SetAttribute ("PacketSize", UintegerValue(500));
        onoff1.SetAttribute ("Remote", AddressValue (InetSocketAddress
(wifiInterfaces.GetAddress (2), port1)));
        onoff1.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1)));
        onoff1.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));
        apps_onoff1.Add (onoff1.Install (wifiStaNodes.Get (0)));
        apps_onoff1.Get(0)->SetStartTime (Seconds (1.0));
        apps_onoff1.Get(0)->SetStopTime (Seconds (3));
    sink1.SetAttribute("Local", AddressValue (InetSocketAddress(Ipv4Address::GetAny
(), port1)));
        apps_sink1.Add(sink1.Install (wifiStaNodes.Get(1)));
        apps_sink1.Get(0)->SetStartTime(Seconds (0.0));
        apps_sink1.Get(0)->SetStopTime(Seconds (SimTime));
std::ostringstream oss;
        oss << "/NodeList/*/ApplicationList/*/$ns3::OnOffApplication/Tx";
        Config::Connect (oss.str(), MakeCallback(&Tx));
        oss.str("");
        oss << "/NodeList/*/ApplicationList/*/$ns3::PacketSink/Rx";
```

```
        Config::Connect (oss.str(), MakeCallback(&Rx));
        oss.str("");
  Simulator::Stop (Seconds (3.0));
  pointToPoint.EnablePcapAll ("monap2");
  phy.EnablePcap ("monap2", apDevices.Get (0));
  csma.EnablePcap ("monap2", csmaDevices.Get (0), true);
  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

X.3 Script 3 :

```
#include "ns3/core-module.h"
    #include "ns3/simulator-module.h"
    #include "ns3/node-module.h"
    #include "ns3/helper-module.h"
    #include "ns3/wifi-module.h"
    #include "ns3/mobility-module.h"
    #include "ns3/ipv4-global-routing-helper.h"
    using namespace ns3;
    void
    Tx (std::string context, Ptr<Packet const> packet)
    {
          NS_LOG_UNCOND (context <<
                "\tSend " <<
                "\tTime " << Simulator::Now().GetSeconds());
    }
    void
    Rx (std::string context, Ptr<Packet const> packet, Address const& add)
    {
          NS_LOG_UNCOND (context <<
                "\tReceive " <<
                "\tTime " << Simulator::Now().GetSeconds());
    }
```

Rapport-Gratuit.com

```
NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");
int
main (int argc, char *argv[])
{
 bool verbose = true;
 uint32_t nCsma = 3;
 uint32_t nWifi = 9;
//double SimTime = 50;
 CommandLine cmd;
 cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
 cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
 cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
 cmd.Parse (argc,argv);
 if (verbose)
  {
   LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
   LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
  }
 NodeContainer p2pNodes;
 p2pNodes.Create (2);
 PointToPointHelper pointToPoint;
 pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
 pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
 NetDeviceContainer p2pDevices;
 p2pDevices = pointToPoint.Install (p2pNodes);
 NodeContainer csmaNodes;
 csmaNodes.Add (p2pNodes.Get (1));
 csmaNodes.Create (nCsma);
 CsmaHelper csma;
 csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
 csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
 NetDeviceContainer csmaDevices;
 csmaDevices = csma.Install (csmaNodes);
```

```
NodeContainer wifiStaNodes;
wifiStaNodes.Create (nWifi);
NodeContainer wifiApNode = p2pNodes.Get (0);
YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());
WifiHelper wifi = WifiHelper::Default ();
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");
QosWifiMacHelper mac = QosWifiMacHelper::Default ();
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac",
             "Ssid", SsidValue (ssid),
             "ActiveProbing", BooleanValue (false));
NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);
mac.SetType ("ns3::ApWifiMac",
             "Ssid", SsidValue (ssid));
NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);
NetDeviceContainer infradevices (apDevices,staDevices);
MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                               "MinX", DoubleValue (0.0),
                               "MinY", DoubleValue (0.0),
                               "DeltaX", DoubleValue (5.0),
                               "DeltaY", DoubleValue (10.0),
                               "GridWidth", UintegerValue (3),
                               "LayoutType", StringValue ("RowFirst"));
mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                           "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility.Install (wifiStaNodes);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);
```

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

```
InternetStackHelper stack;
stack.Install (csmaNodes);
stack.Install (wifiApNode);
stack.Install (wifiStaNodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);
address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);
address.SetBase ("10.1.3.0", "255.255.255.0");
Ipv4InterfaceContainer staswifiInterfaces = address.Assign (staDevices);
Ipv4InterfaceContainer apwifiInterfaces = address.Assign (apDevices);
Ipv4InterfaceContainer wifiInterfaces = address.Assign (infradevices);
UdpEchoServerHelper echoServer (9);
```

- La premiere application

```
ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (3.0));
UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps =
  echoClient.Install (wifiStaNodes.Get (nWifi - 1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (3.0));
```

- La deuxieme application

```
ApplicationContainer serverApps1 = echoServer.Install (csmaNodes.Get (nCsma -
1));
serverApps1.Start (Seconds (1.0));
serverApps1.Stop (Seconds (3.0));
```

```
UdpEchoClientHelper echoClient1 (csmaInterfaces.GetAddress (nCsma - 1), 9);
echoClient1.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient1.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient1.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps1 =
  echoClient1.Install (wifiStaNodes.Get (nWifi - 2));
clientApps1.Start (Seconds (2.0));
clientApps1.Stop (Seconds (3.0));
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

- La troisiéme application

```
ApplicationContainer serverApps2 = echoServer.Install (csmaNodes.Get (nCsma - 2));
serverApps2.Start (Seconds (1.0));
serverApps2.Stop (Seconds (3.0));
UdpEchoClientHelper echoClient2 (csmaInterfaces.GetAddress (nCsma - 2), 9);
echoClient2.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient2.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient2.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps2 =
  echoClient2.Install (wifiStaNodes.Get (nWifi - 3));
clientApps2.Start (Seconds (2.0));
clientApps2.Stop (Seconds (3.0));
```

- La quatriéme application

```
ApplicationContainer serverApps3 = echoServer.Install (csmaNodes.Get (nCsma - 2));
serverApps3.Start (Seconds (1.0));
serverApps3.Stop (Seconds (3.0));
UdpEchoClientHelper echoClient3 (csmaInterfaces.GetAddress (nCsma - 2), 9);
echoClient3.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient3.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient3.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps3 =
  echoClient3.Install (wifiStaNodes.Get (nWifi - 6));
```

```
clientApps3.Start (Seconds (2.0));
clientApps3.Stop (Seconds (3.0));
```

- La sinquiéme application

```
ApplicationContainer serverApps4 = echoServer.Install (csmaNodes.Get (nCsma - 1));
serverApps4.Start (Seconds (1.0));
serverApps4.Stop (Seconds (3.0));
UdpEchoClientHelper echoClient4 (csmaInterfaces.GetAddress (nCsma - 1), 9);
echoClient4.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient4.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient4.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps4 =
  echoClient4.Install (wifiStaNodes.Get (nWifi - 4));
clientApps4.Start (Seconds (2.0));
clientApps4.Stop (Seconds (3.0));
```

- Creation d'une application infrastructure entre les stations wifi

```
uint16_t port1 = 9;
        OnOffHelper onoff1 ("ns3::UdpSocketFactory", Address());
        ApplicationContainer apps_onoff1;
        PacketSinkHelper sink1 ("ns3::UdpSocketFactory", Address ());
        ApplicationContainer apps_sink1;
        onoff1.SetAttribute ("DataRate", DataRateValue(4000));
        onoff1.SetAttribute ("PacketSize", UintegerValue(500));
        onoff1.SetAttribute ("Remote", AddressValue (InetSocketAddress (Ipv4Address ("255.255.255.255"), port1)));
        onoff1.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1)));
        onoff1.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));
        apps_onoff1.Add (onoff1.Install (wifiStaNodes.Get (0)));
        apps_onoff1.Get(0)->SetStartTime (Seconds (0.0));
        apps_onoff1.Get(0)->SetStopTime (Seconds (3));
```

```
   sink1.SetAttribute("Local", AddressValue
(InetSocketAddress(Ipv4Address::GetAny (), port1)));
        apps_sink1.Add(sink1.Install (wifiStaNodes.Get(1)));
        apps_sink1.Get(0)->SetStartTime(Seconds (0.0));
        apps_sink1.Get(0)->SetStopTime(Seconds (3));
        OnOffHelper onoff2 ("ns3::UdpSocketFactory", Address());
        ApplicationContainer apps_onoff2;
        PacketSinkHelper sink2 ("ns3::UdpSocketFactory", Address ());
        ApplicationContainer apps_sink2;
        onoff2.SetAttribute ("DataRate", DataRateValue(4000));
        onoff2.SetAttribute ("PacketSize", UintegerValue(500));
        onoff2.SetAttribute ("Remote", AddressValue (InetSocketAddress
(Ipv4Address ("255.255.255.255"), port1)));
        onoff2.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable
(1)));
        onoff2.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable
(0)));
        apps_onoff2.Add (onoff2.Install (wifiStaNodes.Get (2)));
        apps_onoff2.Get(0)->SetStartTime (Seconds (0.0));
        apps_onoff2.Get(0)->SetStopTime (Seconds (3));
        sink2.SetAttribute("Local", AddressValue
(InetSocketAddress(Ipv4Address::GetAny (), port1)));
        apps_sink2.Add(sink2.Install (wifiStaNodes.Get(3)));
        apps_sink2.Get(0)->SetStartTime(Seconds (1.0));
        apps_sink2.Get(0)->SetStopTime(Seconds (3));
        OnOffHelper onoff3 ("ns3::UdpSocketFactory", Address());
        ApplicationContainer apps_onoff3;
        PacketSinkHelper sink3 ("ns3::UdpSocketFactory", Address ());
        ApplicationContainer apps_sink3;
        onoff3.SetAttribute ("DataRate", DataRateValue(4000));
        onoff3.SetAttribute ("PacketSize", UintegerValue(500));
        onoff3.SetAttribute ("Remote", AddressValue (InetSocketAddress
(Ipv4Address ("255.255.255.255"), port1)));
```

```
        onoff3.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable
(1)));
        onoff3.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable
(0)));
        apps_onoff3.Add (onoff3.Install (wifiStaNodes.Get (2)));
        apps_onoff3.Get(0)->SetStartTime (Seconds (0.0));
        apps_onoff3.Get(0)->SetStopTime (Seconds (3));
        sink3.SetAttribute("Local", AddressValue
(InetSocketAddress(Ipv4Address::GetAny (), port1)));
        apps_sink3.Add(sink3.Install (wifiStaNodes.Get(4)));
        apps_sink3.Get(0)->SetStartTime(Seconds (2.0));
        apps_sink3.Get(0)->SetStopTime(Seconds (3));
    OnOffHelper onoff4 ("ns3::UdpSocketFactory", Address());
        ApplicationContainer apps_onoff4;
        PacketSinkHelper sink4 ("ns3::UdpSocketFactory", Address ());
        ApplicationContainer apps_sink4;
        onoff4.SetAttribute ("DataRate", DataRateValue(4000));
        onoff4.SetAttribute ("PacketSize", UintegerValue(500));
        onoff4.SetAttribute ("Remote", AddressValue (InetSocketAddress
(Ipv4Address ("255.255.255.255"), port1)));
        onoff4.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable
(1)));
        onoff4.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable
(0)));
        apps_onoff4.Add (onoff4.Install (wifiStaNodes.Get (4)));
        apps_onoff4.Get(0)->SetStartTime (Seconds (0.0));
        apps_onoff4.Get(0)->SetStopTime (Seconds (3));
        sink4.SetAttribute("Local", AddressValue
(InetSocketAddress(Ipv4Address::GetAny (), port1)));
        apps_sink4.Add(sink4.Install (wifiStaNodes.Get(5)));
        apps_sink4.Get(0)->SetStartTime(Seconds (2.0));
        apps_sink4.Get(0)->SetStopTime(Seconds (3));
        OnOffHelper onoff5 ("ns3::UdpSocketFactory", Address());
```

```
ApplicationContainer apps_onoff5;
PacketSinkHelper sink5("ns3::UdpSocketFactory", Address ());
ApplicationContainer apps_sink5;
onoff5.SetAttribute ("DataRate", DataRateValue(4000));
onoff5.SetAttribute ("PacketSize", UintegerValue(500));
onoff5.SetAttribute ("Remote", AddressValue (InetSocketAddress (Ipv4Address ("255.255.255.255"), port1)));
onoff5.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1)));
onoff5.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));
apps_onoff5.Add (onoff5.Install (wifiStaNodes.Get (5)));
apps_onoff5.Get(0)->SetStartTime (Seconds (2.0));
apps_onoff5.Get(0)->SetStopTime (Seconds (3));
sink5.SetAttribute("Local", AddressValue (InetSocketAddress(Ipv4Address::GetAny (), port1)));
apps_sink5.Add(sink5.Install (wifiStaNodes.Get(6)));
apps_sink5.Get(0)->SetStartTime(Seconds (2.0));
apps_sink5.Get(0)->SetStopTime(Seconds (3));
OnOffHelper onoff6 ("ns3::UdpSocketFactory", Address());
ApplicationContainer apps_onoff6;
PacketSinkHelper sink6("ns3::UdpSocketFactory", Address ());
ApplicationContainer apps_sink6;
onoff6.SetAttribute ("DataRate", DataRateValue(4000));
onoff6.SetAttribute ("PacketSize", UintegerValue(500));
onoff6.SetAttribute ("Remote", AddressValue (InetSocketAddress (Ipv4Address ("255.255.255.255"), port1)));
onoff6.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1)));
onoff5.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));
apps_onoff6.Add (onoff6.Install (wifiStaNodes.Get (7)));
apps_onoff6.Get(0)->SetStartTime (Seconds (2.0));
```

```
        apps_onoff6.Get(0)->SetStopTime (Seconds (3));
        sink6.SetAttribute("Local",                                    AddressValue
(InetSocketAddress(Ipv4Address::GetAny (), port1)));
        apps_sink6.Add(sink6.Install (wifiStaNodes.Get(8)));
        apps_sink6.Get(0)->SetStartTime(Seconds (2.0));
        apps_sink6.Get(0)->SetStopTime(Seconds (3));
std::ostringstream oss;
        oss << "/NodeList/*/ApplicationList/*/$ns3::OnOffApplication/Tx";
        Config::Connect (oss.str(), MakeCallback(&Tx));
        oss.str("");
        oss << "/NodeList/*/ApplicationList/*/$ns3::PacketSink/Rx";
        Config::Connect (oss.str(), MakeCallback(&Rx));
        oss.str("");
  Simulator::Stop (Seconds (60.0));
  pointToPoint.EnablePcapAll ("script2");
  phy.EnablePcap ("script2", apDevices.Get (0));
  csma.EnablePcap ("script2", csmaDevices.Get (0), true);
  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

X.4 Script 4:

Si nous rajoutons un client nous remarquons que le point d'accés se satirera.

```
#include "ns3/core-module.h"
#include "ns3/simulator-module.h"
#include "ns3/node-module.h"
#include "ns3/helper-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/ipv4-global-routing-helper.h"
using namespace ns3;
void
Tx (std::string context, Ptr<Packet const> packet)
```

```
{
      NS_LOG_UNCOND (context <<
                     "\tSend " <<
                     "\tTime " << Simulator::Now().GetSeconds());
}
void
Rx (std::string context, Ptr<Packet const> packet, Address const& add)
{
      NS_LOG_UNCOND (context <<
                     "\tReceive " <<
                     "\tTime " << Simulator::Now().GetSeconds());
}
NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");
int
main (int argc, char *argv[])
{
  bool verbose = true;
  uint32_t nCsma = 3;
  uint32_t nWifi = 9;
 //double SimTime = 50;
  CommandLine cmd;
  cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
  cmd.Parse (argc,argv);
  if (verbose)
    {
      LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
      LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }
  NodeContainer p2pNodes;
  p2pNodes.Create (2);
  PointToPointHelper pointToPoint;
```

Rapport-Gratuit.com

```
 pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
 NetDeviceContainer p2pDevices;
 p2pDevices = pointToPoint.Install (p2pNodes);
 NodeContainer csmaNodes;
 csmaNodes.Add (p2pNodes.Get (1));
 csmaNodes.Create (nCsma);
 CsmaHelper csma;
 csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
 csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

 NetDeviceContainer csmaDevices;
 csmaDevices = csma.Install (csmaNodes);
 NodeContainer wifiStaNodes;
 wifiStaNodes.Create (nWifi);
 NodeContainer wifiApNode = p2pNodes.Get (0);
 YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
 YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
 phy.SetChannel (channel.Create ());
 WifiHelper wifi = WifiHelper::Default ();
 wifi.SetRemoteStationManager ("ns3::AarfWifiManager");
 QosWifiMacHelper mac = QosWifiMacHelper::Default ();
 Ssid ssid = Ssid ("ns-3-ssid");
 mac.SetType ("ns3::StaWifiMac",
              "Ssid", SsidValue (ssid),
              "ActiveProbing", BooleanValue (false));
 NetDeviceContainer staDevices;
 staDevices = wifi.Install (phy, mac, wifiStaNodes);
 mac.SetType ("ns3::ApWifiMac",
              "Ssid", SsidValue (ssid));
 NetDeviceContainer apDevices;
 apDevices = wifi.Install (phy, mac, wifiApNode);
NetDeviceContainer infradevices (apDevices,staDevices);
```

```
MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                               "MinX", DoubleValue (0.0),
                               "MinY", DoubleValue (0.0),
                               "DeltaX", DoubleValue (5.0),
                               "DeltaY", DoubleValue (10.0),
                               "GridWidth", UintegerValue (3),
                               "LayoutType", StringValue ("RowFirst"));
mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                           "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility.Install (wifiStaNodes);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);
InternetStackHelper stack;
stack.Install (csmaNodes);
stack.Install (wifiApNode);
stack.Install (wifiStaNodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);
address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);
address.SetBase ("10.1.3.0", "255.255.255.0");
Ipv4InterfaceContainer staswifiInterfaces = address.Assign (staDevices);
Ipv4InterfaceContainer apwifiInterfaces = address.Assign (apDevices);
Ipv4InterfaceContainer wifiInterfaces = address.Assign (infradevices);
UdpEchoServerHelper echoServer (9);
```

- La premiere application

```
ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (3.0));
```

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

```
UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps =
  echoClient.Install (wifiStaNodes.Get (nWifi - 1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (3.0));
```

- La deuxiéme application

```
ApplicationContainer serverApps1 = echoServer.Install (csmaNodes.Get (nCsma - 1));
serverApps1.Start (Seconds (1.0));
serverApps1.Stop (Seconds (3.0));
UdpEchoClientHelper echoClient1 (csmaInterfaces.GetAddress (nCsma - 1), 9);
echoClient1.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient1.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient1.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps1 =
  echoClient1.Install (wifiStaNodes.Get (nWifi - 2));
clientApps1.Start (Seconds (2.0));
clientApps1.Stop (Seconds (3.0));
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

- La troisieme application

```
ApplicationContainer serverApps2 = echoServer.Install (csmaNodes.Get (nCsma - 2));
serverApps2.Start (Seconds (1.0));
serverApps2.Stop (Seconds (3.0));
UdpEchoClientHelper echoClient2 (csmaInterfaces.GetAddress (nCsma - 2), 9);
echoClient2.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient2.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient2.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps2 =
  echoClient2.Install (wifiStaNodes.Get (nWifi - 3));
clientApps2.Start (Seconds (2.0));
clientApps2.Stop (Seconds (3.0));
```

- La quatriéme application

```
ApplicationContainer serverApps3 = echoServer.Install (csmaNodes.Get (nCsma - 2));
serverApps3.Start (Seconds (1.0));
serverApps3.Stop (Seconds (3.0));
UdpEchoClientHelper echoClient3 (csmaInterfaces.GetAddress (nCsma - 2), 9);
echoClient3.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient3.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient3.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps3 =
  echoClient3.Install (wifiStaNodes.Get (nWifi - 6));
clientApps3.Start (Seconds (2.0));
clientApps3.Stop (Seconds (3.0));
```

- La sinquiéme application

```
ApplicationContainer serverApps4 = echoServer.Install (csmaNodes.Get (nCsma - 1));
serverApps4.Start (Seconds (1.0));
serverApps4.Stop (Seconds (3.0));
UdpEchoClientHelper echoClient4 (csmaInterfaces.GetAddress (nCsma - 1), 9);
echoClient4.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient4.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient4.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps4 =
  echoClient4.Install (wifiStaNodes.Get (nWifi - 4));
clientApps4.Start (Seconds (2.0));
clientApps4.Stop (Seconds (3.0));
```

- **La sixieme application**

```
ApplicationContainer serverApps5 = echoServer.Install (csmaNodes.Get (nCsma - 3));
serverApps5.Start (Seconds (1.0));
serverApps5.Stop (Seconds (3.0));
UdpEchoClientHelper echoClient5 (csmaInterfaces.GetAddress (nCsma - 3), 9);
echoClient5.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient5.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient5.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps5 =
```

```
  echoClient5.Install (wifiStaNodes.Get (nWifi - 3));
 clientApps5.Start (Seconds (2.0));
 clientApps5.Stop (Seconds (3.0));
```

- Creation d'une application infrastructure entre les stations wifi

```
uint16_t port1 = 9;
      OnOffHelper onoff1 ("ns3::UdpSocketFactory", Address());
      ApplicationContainer apps_onoff1;
      PacketSinkHelper sink1 ("ns3::UdpSocketFactory", Address ());
      ApplicationContainer apps_sink1;
      onoff1.SetAttribute ("DataRate", DataRateValue(4000));
      onoff1.SetAttribute ("PacketSize", UintegerValue(500));
      onoff1.SetAttribute ("Remote", AddressValue (InetSocketAddress (Ipv4Address ("255.255.255.255"), port1)));
      onoff1.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1)));
      onoff1.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));
      apps_onoff1.Add (onoff1.Install (wifiStaNodes.Get (0)));
      apps_onoff1.Get(0)->SetStartTime (Seconds (0.0));
      apps_onoff1.Get(0)->SetStopTime (Seconds (3));
   sink1.SetAttribute("Local", AddressValue (InetSocketAddress(Ipv4Address::GetAny (), port1)));
      apps_sink1.Add(sink1.Install (wifiStaNodes.Get(1)));
      apps_sink1.Get(0)->SetStartTime(Seconds (0.0));
      apps_sink1.Get(0)->SetStopTime(Seconds (3));
      OnOffHelper onoff2 ("ns3::UdpSocketFactory", Address());
      ApplicationContainer apps_onoff2;
      PacketSinkHelper sink2 ("ns3::UdpSocketFactory", Address ());
      ApplicationContainer apps_sink2;
      onoff2.SetAttribute ("DataRate", DataRateValue(4000));
      onoff2.SetAttribute ("PacketSize", UintegerValue(500));
      onoff2.SetAttribute ("Remote", AddressValue (InetSocketAddress (Ipv4Address ("255.255.255.255"), port1)));
      onoff2.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1)));
      onoff2.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));
```

```
apps_onoff2.Add (onoff2.Install (wifiStaNodes.Get (2)));
apps_onoff2.Get(0)->SetStartTime (Seconds (1.0));
apps_onoff2.Get(0)->SetStopTime (Seconds (2));
sink2.SetAttribute("Local", AddressValue (InetSocketAddress(Ipv4Address::GetAny (), port1)));
apps_sink2.Add(sink2.Install (wifiStaNodes.Get(3)));
apps_sink2.Get(0)->SetStartTime(Seconds (1.0));
apps_sink2.Get(0)->SetStopTime(Seconds (2));
OnOffHelper onoff3 ("ns3::UdpSocketFactory", Address());
ApplicationContainer apps_onoff3;
PacketSinkHelper sink3 ("ns3::UdpSocketFactory", Address ());
ApplicationContainer apps_sink3;
onoff3.SetAttribute ("DataRate", DataRateValue(4000));
onoff3.SetAttribute ("PacketSize", UintegerValue(500));
onoff3.SetAttribute ("Remote", AddressValue (InetSocketAddress (Ipv4Address ("255.255.255.255"), port1)));
onoff3.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1)));
onoff3.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));
apps_onoff3.Add (onoff3.Install (wifiStaNodes.Get (2)));
apps_onoff3.Get(0)->SetStartTime (Seconds (2.0));
apps_onoff3.Get(0)->SetStopTime (Seconds (3));
sink3.SetAttribute("Local", AddressValue (InetSocketAddress(Ipv4Address::GetAny (), port1)));
apps_sink3.Add(sink3.Install (wifiStaNodes.Get(4)));
apps_sink3.Get(0)->SetStartTime(Seconds (2.0));
apps_sink3.Get(0)->SetStopTime(Seconds (3));
OnOffHelper onoff4 ("ns3::UdpSocketFactory", Address());
ApplicationContainer apps_onoff4;
PacketSinkHelper sink4 ("ns3::UdpSocketFactory", Address ());
ApplicationContainer apps_sink4;
onoff4.SetAttribute ("DataRate", DataRateValue(4000));
onoff4.SetAttribute ("PacketSize", UintegerValue(500));
```

```
onoff4.SetAttribute ("Remote", AddressValue (InetSocketAddress (Ipv4Address ("255.255.255.255"), port1)));
onoff4.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1)));
onoff4.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));
apps_onoff4.Add (onoff4.Install (wifiStaNodes.Get (4)));
apps_onoff4.Get(0)->SetStartTime (Seconds (2.0));
apps_onoff4.Get(0)->SetStopTime (Seconds (3));
sink4.SetAttribute("Local", AddressValue (InetSocketAddress(Ipv4Address::GetAny (), port1)));
apps_sink4.Add(sink4.Install (wifiStaNodes.Get(5)));
apps_sink4.Get(0)->SetStartTime(Seconds (2.0));
apps_sink4.Get(0)->SetStopTime(Seconds (3));
OnOffHelper onoff5 ("ns3::UdpSocketFactory", Address());
ApplicationContainer apps_onoff5;
PacketSinkHelper sink5("ns3::UdpSocketFactory", Address ());
ApplicationContainer apps_sink5;
onoff5.SetAttribute ("DataRate", DataRateValue(4000));
onoff5.SetAttribute ("PacketSize", UintegerValue(500));
onoff5.SetAttribute ("Remote", AddressValue (InetSocketAddress (Ipv4Address ("255.255.255.255"), port1)));
onoff5.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1)));
onoff5.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));
apps_onoff5.Add (onoff5.Install (wifiStaNodes.Get (5)));
apps_onoff5.Get(0)->SetStartTime (Seconds (2.0));
apps_onoff5.Get(0)->SetStopTime (Seconds (3));
sink5.SetAttribute("Local", AddressValue (InetSocketAddress(Ipv4Address::GetAny (), port1)));
apps_sink5.Add(sink5.Install (wifiStaNodes.Get(6)));
apps_sink5.Get(0)->SetStartTime(Seconds (2.0));
apps_sink5.Get(0)->SetStopTime(Seconds (3));
OnOffHelper onoff6 ("ns3::UdpSocketFactory", Address());
ApplicationContainer apps_onoff6;
PacketSinkHelper sink6("ns3::UdpSocketFactory", Address ());
```

```
        ApplicationContainer apps_sink6;
        onoff6.SetAttribute ("DataRate", DataRateValue(4000));
        onoff6.SetAttribute ("PacketSize", UintegerValue(500));
        onoff6.SetAttribute ("Remote", AddressValue (InetSocketAddress (Ipv4Address
("255.255.255.255"), port1)));
        onoff6.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1)));
        onoff5.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));
        apps_onoff6.Add (onoff6.Install (wifiStaNodes.Get (7)));
        apps_onoff6.Get(0)->SetStartTime (Seconds (2.0));
        apps_onoff6.Get(0)->SetStopTime (Seconds (3));
        sink6.SetAttribute("Local",                                    AddressValue
(InetSocketAddress(Ipv4Address::GetAny (), port1)));
        apps_sink6.Add(sink6.Install (wifiStaNodes.Get(8)));
        apps_sink6.Get(0)->SetStartTime(Seconds (2.0));
        apps_sink6.Get(0)->SetStopTime(Seconds (3));
std::ostringstream oss;
        oss << "/NodeList/*/ApplicationList/*/$ns3::OnOffApplication/Tx";
        Config::Connect (oss.str(), MakeCallback(&Tx));
        oss.str("");
        oss << "/NodeList/*/ApplicationList/*/$ns3::PacketSink/Rx";
        Config::Connect (oss.str(), MakeCallback(&Rx));
        oss.str("");
  Simulator::Stop (Seconds (60.0));
  pointToPoint.EnablePcapAll ("script2");
  phy.EnablePcap ("script2", apDevices.Get (0));
  csma.EnablePcap ("script2", csmaDevices.Get (0), true);
  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```