Réf: 2007/ II / 3ex **Soutenu à la session de Juillet 2007**

Université de la Manouba
ECOLE NATIONALE DES SCIENCES DE L'INFORMATIQUE

RAPPORT
de Mémoire de Fin d'Etudes

Présenté en vue de l'obtention du titre
d'INGENIEUR EN INFORMATIQUE

par
Ahmed Ayadi

SUJET :

Extensions du simulateur Omnet++ pour la validation de mécanismes de transmission multimédia dans les réseaux sans fils IEEE 802.11

Organisme : Projet Planète INRIA – Sophia Antipolis

Nom du responsable : Dr. Walid Dabbous

Encadré par : Dr. Thierry Turletti

Supervisé par : Pr. Abdelfetteh Belghith

Adresse : 2004 route des lucioles BP 93 06902 Sophia Antipolis France

Tél : +33 4 92 38 77 77 Fax: +33 4 92 38 77 65

Dédicace

Je dédie ce travail

A ma mère

A mon père

A mes frères

A ma sœur

A toute ma famille

A mes chers amis.

Ahmed

Remerciement

Ces travaux de mémoire de fin d'étude se sont déroulés au sein du projet Planète à l'INRIA Sophia Antipolis.

Je remercie Monsieur Walid Dabbous le chef du projet, de m'y avoir accueilli et donné les moyens de mener à bien mes travaux. Je remercie vivement mon encadrant Monsieur Thierry Turletti pour sa disponibilité et ses précieux conseils qui m'a permis d'enrichir mon travail, je le remercie également pour son soutien tout au long du déroulement de mon projet.

Je suis reconnaissant à Monsieur Abdelfettah Belghith Professeur à l'école nationale des sciences de l'informatique d'avoir accepté d'être superviseur de mon travail.

Je tiens aussi à adresser ma gratitude à toutes les personnes qui m'ont aidé pendant la préparation de ce travail, je pense ici à Monsieur Diego Dujovne doctorant à l'équipe Planète.

Je tiens à remercier profondément l'ensemble des stagiaires de l'équipe Amir Krifa, Mehdi Msekni, Mohamed Karim Sbai, Jayoung Choi, Mouna Abdelmomen, Naveed Ben Rais avec lesquels j'ai eu des échanges scientifiques et culturels pendant toute la durée du projet.

Je ne saurais terminer ces remerciements sans penser aux membres du jury pour l'honneur qu'ils m'ont fait d'avoir voulu examiner et évaluer cette modeste contribution et à toute personne qui a contribué, directement ou indirectement, à l'achèvement de ce travail.

Résumé

Le présent rapport a été élaboré dans le cadre du projet de fin d'études pour l'obtention du diplôme d'ingénieur en informatique. Ce travail consiste à implémenter de nouveaux modules dans un simulateur OMNET++ afin de simuler de nouvelles approches permettant d'améliorer la qualité de transmission vidéo dans le réseau IEEE 802.11. Après une description des protocoles de couche liaisons de donnés et physiques su standard 802.11, nous présentons les algorithmes proposés sélection du débit de transmission physiques. Nous présentons ensuite la transmission vidéo dans les réseaux sans fils et la nouvelle approche basé sur le Leader. Enfin, nous présentons les résultats de nos simulations réalisés au cours de ce projet.

Mot clés : Réseaux Locaux sans fils IEEE 802.11, Sélection du débit de transmission physique, Protocoles temps réel RTP/RTCP, Transmission multimédia dans les réseaux locaux sans fils.

Abstract

This report was elaborate within the framework of the graduation project for obtaining the diploma of computer engineer. This work consists in implementing new module on OMNET++ simulator to simulate new approach for improving quality of video transmission over IEEE 802.11 networks. After a short description of physical and link layer, we present algorithms proposed of selecting physical rate. We then present video transmission over IEEE 802.11 wireless networks and the new approach based on leaser. Finality, we show results of our simulations.

Key words: IEEE 802.11, Wireless LAN, 802.11 MAC/PHY Layer, Physical Rate Selection Mechanisms, Real Time protocols, Multimedia Transmission in WLANs.

Tables des matières

Liste des figures

Liste des tableaux

Introduction générale

Aujourd'hui, nous assistons à une évolution de l'Internet en nombre d'utilisateurs. Parmi les facteurs de cette évolution se trouve le succès des réseaux sans fil 802.11. Les réseaux IEEE 802.11 deviennent de plus en plus populaires car ils permettent aux utilisateurs de se connecter à l'Internet à un prix abordable avec une bande passante relativement importante et aussi la possibilité de se déplacer sans être déconnecté. De plus, de nos jours les cartes réseau sans fil IEEE 802.11 sont déployées dans la majorité des technologies comme les PDAs et les laptops.

En parallèle, les techniques de communication multimédia ont aussi évolué avec les nouveaux algorithmes de compression et de codage. Ainsi, de nombreuses applications multimédia deviennent accessibles à partir des réseaux sans fil. Mais ils présentent encore des obstacles au déploiement. Les problèmes majeurs de ces réseaux sont le taux de perte et la variation de délai sachant que les applications multimédia sont très exigeantes. Une solution évidente pour optimiser l'utilisation de la bande passante et améliorer la qualité de la vidéo consiste à transmettre la vidéo en multipoint à un ensemble d'utilisateurs.

Mais l'utilisation du multipoint standard présente trois problèmes principaux. Le premier est l'impossibilité d'adapter la fenêtre de collision suivant l'état du réseau. Le second est l'impossibilité d'adapter le débit physique suivant l'état du support de transmission, donc les paquets sont transmis à un débit physique fixe. Le troisième est l'impossibilité de retransmettre au niveau de la couche MAC les paquets perdus.

Une nouvelle approche a récemment été proposée pour remédier à ses problèmes. Elle consiste en l'élection d'un récepteur appelé leader pour assurer l'acquittement des paquets reçus. Ainsi, l'émetteur peut adapter le débit physique et retransmettre les paquets perdus.

Nous allons dans ce projet montrer l'apport de cette nouvelle approche par rapport à la méthode de transmission multipoint standard à l'aide de la simulation. Nos simulations seront effectuées à l'aide du simulateur OMNET++.

Sujet traité :

Ce projet consiste à intégrer de nouvelles fonctionnalites dans le simulateur OMNET++ (implementer de nouveaux protocoles, ajouter de nouveaux modules, etc..) afin de permettre des simulations plus réalistes. De plus, il s'agit d'implémenter de nouveaux algorithmes d'élection et d'adaptation de débit physique pour la transmission vidéo en ajoutant de nouvelles trames de gestion du réseau. Ces trames serviront à estimer l'état actuelle du réseau (débit, SNR, Perte de paquet, etc..). Enfin, Il s'agit de simuler quelques exemples de scénarios de transmission multipoint.

Dans ce rapport, nous proposons un cadre théorique pour la transmission vidéo dans les réseaux IEEE 802.11. En premier lieu, nous allons introduire le cadre général de notre projet, ensuite nous allons traiter la transmission vidéo dans les réseaux IEEE 802.11. En effet, dans un premier chapitre, nous étudierons l'existant. Nous y présenterons d'abord les modèles physiques de propagation, la norme IEEE 802.11, les protocoles de temps réel ainsi que les différentes approches proposées pour la diffusion multipoint. Dans un deuxième chapitre, nous spécifierons les besoins fonctionnels et les contraintes de notre application. Dans un quatrième chapitre, nous entamerons la conception avec une description des nouveaux

modules ajoutés. Enfin, un cinquième chapitre s'intéressera à la réalisation dans lequel nous montrerons les résultats de nos simulations.

Présentation de l'organisme d'accueil

Dans cette première partie, il s'agit de mettre le stage dans son cadre général. Nous nous proposons alors de présenter l'environnement du stage à travers une présentation de l'équipe PLANETE INRIA- Sophia Antipolis qui a proposé ce sujet.

1. INRIA

L'INRIA, Institut National de Recherche en Informatique et Automatique (FRANCE) placé sous la double tutelle des ministères français de la recherche et de l'industrie, a pour vocation d'entreprendre des recherches fondamentales et appliquées dans les domaines des sciences et technologies de l'information et de la communication (STIC). L'institut assure également un fort transfert technologique en accordant une grande attention à la formation par la recherche, à la diffusion de l'information scientifique et technique, à la valorisation, à l'expertise et à la participation à des programmes internationaux. L'INRIA accueille dans ses 6 unités de recherche situées à Rocquencourt, Rennes, Sophia Antipolis, Grenoble, Nancy et Bordeaux, Lille, Saclay et sur d'autres sites à Paris, Marseille, Lyon et Metz, 3600 personnes dont 2800 scientifiques qui travaillent dans plus de 138 projets de recherches communs.

2. L'unité de Recherche INRIA Sophia Antipolis

Créée au coeur de la technopole Sophia Antipolis en 1983, l'unité de recherche regroupe, sur ses sites de Sophia Antipolis, Marseille et Montpellier, 500 personnes dont 380 scientifiques réparties au sein d'une trentaine d'équipes en partenariat avec le CNRS, plusieurs universités et grandes écoles. Leurs travaux portent sur la conception et la programmation de systèmes informatiques performants, la représentation et la manipulation d'informations complexes, la création, la modélisation et la simulation d'expériences complexes. Ils permettent l'avancée des connaissances dans quatre grands domaines :

- réseaux et systèmes
- génie logiciel et calcul symbolique
- interaction homme machine
- images, données, connaissances, simulation et optimisation des systèmes complexes.

3. Le projet Planète

Notre stage est effectué au sein de l'équipe PLANETE. Les activités de cette équipe, localisés à l'INRIA Sophia Antipolis et à l'INRIA Rhône-alpes, sont centrées sur la conception, la mise en oeuvre et l'évaluation des protocoles et des applications Internet. Leur travaux s'articulent autour plusieurs axes dont l'étude de l'impact des nouveaux supports de transmission sur les protocoles; le passage à l'échelle du routage multipoint; le support de la qualité de service dans l'Internet; et les applications interactives multi-utilisateurs.

Chapitre 1 : Etat de l'art

Introduction

Notre travail consiste a mettre en évidence l'avantage des nouvelles approches proposées pour améliorer la qualité de transmission multimédia dans les réseaux IEEE 802.11 Pour ce faire, nous avons recourt à la simulation qui sera réalisée grâce au simulateur OMNET++ [Omnet, 07]. Ce dernier pose de diverses contraintes comme le besoin de modèles physique de propagation réalistes, la manque de la couche physique IEEE 802.11a et l'absence des protocoles temps réels RTP et RTCP. Dans ce premier chapitre nous allons essentiellement introduire la norme IEEE 802.11 en détaillant les deux couches physique et liaisons de donnée. Nous présentons dans cette première partie quelques modèles de transmission physique ainsi que les algorithmes d'adaptation du débit physique au niveau liaison de donnée. Nous introduisons ensuite le standard RTP avec ses deux protocoles RTP et RTCP. Puis, nous présentons un état de l'art sur la transmission multipoint dans les réseaux IEEE 802.11 avec une présentation des nouvelles approches adoptées pour améliorer la qualité de la transmission vidéo. Enfin, nous présentons une vue d'ensemble du simulateur OMNET++.

1. Introduction à la norme IEEE 802.11

La norme IEEE 802.11 (ISO/IEC 8802-11) est un standard international décrivant les caractéristiques d'un réseau local sans fil *Wireless Local Area Network* (WLAN). Grâce au IEEE 802.11, il est possible de créer des réseaux locaux sans fils à haut débit à condition que l'ordinateur à connecter ne soit pas trop distante par rapport au point d'accès. Dans la pratique, ce réseau permet de relier des ordinateurs portables, des ordinateurs de bureau, des assistants personnels (PDA) ou tout type de périphérique à une liaison haut débit sur un rayon de plusieurs dizaines de mètres en intérieur, et à plusieurs centaines de mètres en environnement ouvert. Ainsi, des opérateurs commencent à irriguer des zones à fortes concentration d'utilisateurs (gares, aéroports, hotels, trains, ...) avec des réseaux sans fils. Ces zones d'accès sont appelées « hot spots ». Nous présentons dans ce qui suit les deux couches de la norme IEEE 802.11 qui sont la couche physique et la couche liaison de donnée.

1.1. La couche PHY IEEE 802.11

Le rôle de la couche physique *Physical Layer* (PHY) est de transporter correctement les données que l'émetteur souhaite envoyer au récepteur. Elle est divisée en deux sous-couches, PLCP *Physical Layer Convergence Protocol* et PMD *Physical Medium Dependent*. Cette dernière s'occupe de l'encodage des données, alors que la sous couche PLCP prend en charge l'écoute du support, en fournissant à cette occasion un signal à la couche MAC pour lui dire si le support est libre ou non.

La couche PHY définit aussi la modulation des ondes radioélectriques et les caractéristiques de la signalisation pour la transmission de données. La norme 802.11 propose en réalité trois couches physiques, définissant des modes de transmission alternatifs comme le montre le tableau 1.1

La couche IEEE 802.11b utilise la bande 2.4 Ghz et permet d'atteindre un débit théorique de 11 Mbps en utilisant les technologies d'étalement de spectre avec sauts de fréquence *frequency hopping spread spectrum* (FHSS), de séquence directe *direct sequence spread spectrum* (DSSS), et l'infra rouge (IR). Alors que IEEE 802.11a transmet dans la bande 5 GHz et peut atteindre un débit de 54Mbps en utilisant *orthogonal frequency division multiplexing* (OFDM). Le standard IEEE 802.11g est une extension du IEEE 802.11b qui peut atteindre un débit théorique maximale de 54 Mbps.

Chaque mode de transmission est caractérisé par ses méthodes de modulation. La performance d'une modulation par rapport à une autre réside dans sa résistance contre les erreurs de propagation, les interférences et le *fading*.

Pour chaque mode de transmission, il y a toujours un mode de transmission de base utilisé généralement pour la transmission des ACK, RTS, CTS et les entêtes PLCP. Ce mode de transmission utilise BPSK ou DBPSK comme modulation pour avoir le minimum d'erreur. Les modes de transmission de base pour les couches physiques sont décrits dans le tableau 1.1. Par exemple, le débit de base pour IEEE 802.11b est 1 Mbps.

Caractéristiques	802.11a	802.11b	802.11g
Fréquence	5 Ghz	2.4 Ghz	2.4 Ghz
Débit (Mbps)	6, 9, 12, 18, 24, 36, 48, 54	1, 2, 5.5, 11	1, 2, 5.5, 6,9, 11, 12, 18, 22, 24, 33, 36, 48, 54
Modulation	BPSK, QPSK, 16 QAM, 64 QAM (OFDM)	DBPSK, DQPSK, CCK (DSSS, IR, et FH)	BPSK, DBPSK, QPSK, DQPSK, CCK, 16 QAM, 64 QAM (OFDM et DSSS)
FEC	1/2, 2/3, 3/4		1/2, 2/3, 3/4
Débit de base (Mbps)	6	1 ou 2	1, 2 ou 6

Tableau 1.1 Caractéristiques des différentes couches physiques IEEE 802.11 [Manshaei, 05]

Dans la norme IEEE 802.11, chaque paquet peut être envoyé avec deux débits différents. L'entête PLCP est envoyée avec le débit de base, alors que la deuxième partie du paquet est envoyée avec un débit généralement plus élevé. Le débit de la deuxième partie est indiqué dans un champs de l'entête PLCP. Par la suite, le récepteur décode l'entête PLCP afin de connaître le débit de la deuxième partie du paquet.

1.2. Les différents modèles de propagation

Les ondes électromagnétiques sont actuellement le support de la plupart des communications sans fil et l'étude de leur propagation devient de plus en plus importante afin de pouvoir prédire l'onde reçue par une station réceptrice, connaissant l'onde émise. Elles sont utilisées pour diverses applications, à l'intérieur ou à l'extérieur, mais l'influence des effets qu'elles doivent subir est bien souvent différent selon le contexte. En effet, des phénomènes tels que la diffraction, la dispersion, la réflexion, l'absorption ou encore la transmission ont un impact direct sur la propagation du signal.

La modélisation de la propagation dans un réseau IEEE 802.11 comprend les modèles d'affaiblissement du signal à grande échelle *large-scale path loss* et les modèles à petite échelle *small-scale path loss* ainsi que les différentes fonctions de calcul de la puissance reçue et les méthodes de calcul du taux d'erreur d'un bit *Bit Error Rate* (BER) et du taux d'erreur d'un paquet *Packet Error Rate* (PER).

1.2.1. Les modèles à grande échelle

1.2.1.1. Le modèle Free-Space

Free-Space est le modèle le plus utilisé dans la majorité des simulateurs afin de calculer la puissance reçue. Ce modèle exige l'existence d'un chemin direct entre l'émetteur et le récepteur. De plus, Il exige que les deux noeuds se trouvent dans un environnement sans bruit.

La formule (1.1) de Friis est utilisée pour le calcul de la puissance reçue [Rappaport, 02].

$$Pr = \frac{PrGtGr\lambda^2}{(4\pi d)^2 L}. \qquad (1.1)$$

Avec
Pr : la puissance reçue par le récepteur
Pt : la puissance transmise par l'émetteur
Gt : le gain de l'émetteur
Gr : le gain du récepteur
λ : la longueur l'onde
d : la distance entre le récepteur et l'émetteur
L : la perte due au système

1.2.1.2. Le modèle Two-Ray

Ce modèle est plus réaliste que le premier puisqu'il tient en compte des réflexions subites par le signal lors de sa propagation de l'émetteur jusqu'à son arrivée au récepteur. L'émetteur et le récepteur se trouvent dans un environnement sans bruit avec un chemin direct entre eux. Par suite, Le signal émis par l'émetteur subira des réflexions afin d'arriver au récepteur. Ce modèle sera le bon choix surtout lorsque la distance entre l'émetteur et le récepteur est assez grande et que l'émetteur se trouve à une grande hauteur.

A une grande distance de l'émetteur, la distance *d* est suffisamment grande devant $(ht*hr)^2$ et donc, la puissance reçue est calculée grâce la formule (1.2) [Rappaport, 02] :

$$Pr = \frac{PrGtGr(hrht)^2}{d^4 L}. \qquad (1.2)$$

Avec
Pr : la puissance reçue par le récepteur
Pt : la puissance transmise par l'émetteur
Gt : le gain de l'émetteur
Gr : le gain du récepteur
ht : la hauteur de l'émetteur
hr : la hauteur du récepteur
d : la distance entre le récepteur et l'émetteur
L : la perte due au système

1.2.1.3. Le modèle Shadowing

Ce modèle n'exige pas l'existence d'un chemin direct entre l'émetteur et le récepteur. Il modélise les déviations subites par le signal lors de sa propagation. En adoptant ce modèle, nous tenons en compte des phénomènes imprévisibles que peut subir le signal. La puissance reçue dans ce modèle varie en fonction du logarithme de la distance. La perte moyenne pour une distance donnée est exprimée par *PathLossExponent*. Nous ajoutons ensuite le phénomène de *Shadowing* qui est une variation statistique du signal autour de la valeur calculé à l'aide de Free-Space théorique. Cette variation est de moyenne nulle et sa variance σ est bien évidemment non nulle.

Pour calculer la puissance reçue, nous calculons tout d'abord une puissance référence en supposant que le récepteur est à une distance, dans notre cas 1 mètre, de la source à l'aide de la formule de Friis. Ensuite, nous ajoutons la perte due à la distance et l'effet de Shadowing.

$$Pr(dBW) = PuissanceRéférence(dBW) - 10 PathLossExponent \log(distance) + Shadowing \quad (1.3)$$

Les valeurs de PathLossExponent et *Shadowing* dépendent de l'environnement. Le tableau 1.2 présente les valeurs des environnements typiques selon [Khosroshahy, 06] et [Rappaport, 02].

Environnement	Path loss Exponent	Shadowing Variance
Outdoor- Free Space	2	4 -12
Outdoor – Shadowed/Urban	2.7 - 5	4 - 12
Indoor – Line of sight	1.6 – 1.8	3 – 6
Indoor – Obstructed	4 – 6	6 - 8

Tableau 1.2 Les valeurs typiques de Path loss Exponent et Shadowing Variance [Khosroshahy, 06]

1.2.2. Les modèles à petite échelle « fading »

Small-Scale fading, ou simplement fading, est utilisé selon T. Rappaport [Rappaport, 02] pour décrire une fluctuation rapide de l'amplitude ou de la phase du signal pendant une très courte période. La cause de ce phénomène est l'interférence entre les différentes versions du signal émis quand elles arrivent au récepteur dans des instants distincts.

Ce modèle peut être ajouté avec l'un des modèles à grande échelle déjà cité. Il s'intéresse au coté dynamique de l'environnement, comme les mouvements de l'émetteur ou du récepteur au cours des transmission des données.

1.2.3. Calcul de la probabilité d'erreur de bit BER *Bit Error Rate*

Pour obtenir des simulations de réseaux sans fils IEEE 802.11 plus réalistes, il faut pouvoir estimer avec précision les valeurs du BER et PER. Dans notre projet, nous sommes basés sur le travail de Masood Khosroshahy [Khosroshahy, 06] pour le calcul du BER et du PER. Il a déjà intégré son travail dans le simulateur YANS. Les formules pour le calcul du BER pour les différents types de modulations sont bien détaillées dans [Khosroshahy, 06].

1.2.4. Calcul de la probabilité d'erreur de paquet PER *Packet Error Rate*

Dans cette partie, nous introduisons deux méthodes de calcul de la probabilité qu'un paquet soit perdu. La première, qui est la plus simple, est appelée la distribution uniforme de l'erreur *Uniform Error Distribution*. La deuxième est une nouvelle méthode proposée par Ramin Khalili et Kavé Salamatian [Khalili et al., 06].

1.2.4.1. Distribution uniforme de l'erreur

Chaque paquet est composé de deux morceaux, la partie entête (HEADER) et une partie donnée (PAYLOAD). Les deux morceaux sont envoyés dans la plupart des cas avec des débits de transmission différents. La probabilité de succès d'une transmission d'un

morceau *Chunk Success Rate* (CSR) est calculée en fonction du BER de chaque bit de ce morceau à l'aide de l'équation (1.4).

Ensuite, La probabilité de succès d'une transmission d'un paquet PSR sera le produit des deux probabilités de chaque morceau [Manshaei et al. 05]. Enfin, la probabilité d'erreur d'un paquet PER sera 1-PSR.

$$CSR = (1 - BER))^{nbits} \quad (1.4)$$

$$PER = 1 - CSR_{PLCP}.CSR_{PLCP} \quad (1.5)$$

$$PER = 1-(1-BER_{PLCP})^{HEADER\_LENGTH}.(1-BER_{PAYLOAD})^{HEADER\_LENGTH} \quad (1.6)$$

Cette méthode suppose que la valeur de BER est uniforme pour tous les bits d'un paquet.

1.2.4.2. Distribution d'erreur non uniforme

Dans ce paragraphe, nous présentons une deuxième approche pour le calcul du BER et du PER en se basant sur [Khalili et al., 06]. Selon Khalili et Salamatian [Khalili et al., 06] l'hypothèse que la distribution d'erreur est uniforme mène à une surestimation de la valeur PER. Ils ont ajouté de nouvelles notions afin de proposer d'autres formules pour le calcul du PER comme EER et λ. Le taux d'erreur *Error Event Rate* (EER) est une probabilité indiquant la fréquence d'occurrence d'une erreur dans une partie d'un paquet. λ est le paramètre d'une loi géométrique qui décrit la longueur d'une période d'erreur. Ce taux dépend de la valeur de SNR et aussi du FEC. En bref, pour un morceau de paquet, il s'agit de tirer un nombre aléatoire qui serait l'événement d'un début d'une période d'erreur. Ensuite, tirer un autre nombre aléatoire qui représentera la durée en bits de cette erreur. Enfin, pour chaque bit dans cette période nous choisissons un nombre aléatoire et le comparons avec BER/ λ.

1.3. La couche MAC 802.11

La couche de liaison de données de 802.11 se compose de deux sous-couches : le contrôle de la liaison logique *Logical Link Control* (LLC) et le contrôle d'accès au support *Medium Access Control* (MAC). Le standard 802.11 utilise la norme LLC 802.2 et un adressage sur 48 bits, tout comme les réseaux IEEE 802, simplifiant ainsi le pontage entre les réseaux sans fil et filaires. Le contrôle d'accès au support est en revanche propre aux WLAN. La couche MAC 802.11 est très proche de 802.3 dans sa conception : il est conçu pour supporter de multiples utilisateurs sur un support partagé en faisant détecter le support par l'expéditeur avant d'y accéder.

Pour les LAN Ethernet IEEE 802.3, le protocole CSMA/CD *Carrier Sense Multiple Access with Collision Detection* régule l'accès des stations Ethernet au câble. Dans un WLAN 802.11, la détection des collisions est impossible du fait de ce qu'on appelle le problème "near/far". Pour détecter une collision, une station doit être capable de transmettre et d'écouter en même temps. Or, dans les systèmes radio, il ne peut y avoir transmission et écoute simultanées. En plus, la collision se fait au voisinage du récepteur et non pas de l'émetteur. Donc, l'émetteur peut ne pas détecter la collision.

Pour prendre en compte cette différence, le standard 802.11 fait appel à un protocole légèrement modifié, baptisé CSMA/CA *Carrier Sense Multiple Access with Collision Avoidance*, ou à la fonction DCF *Distributed Coordination Function*. Le protocole CSMA/CA tente d'éviter les collisions en imposant un accusé de réception systématique des

paquets (ACK), ce qui signifie que pour chaque paquet de données arrivé intact, un paquet ACK est émis par la station de réception.

Ce protocole CSMA/CA fonctionne de la manière suivante : une station qui souhaite émettre commence par explorer les ondes et, si aucune activité n'est détectée, elle attend un temps aléatoire avant de transmettre si le support est toujours libre. Si le paquet est intact à la réception, la station réceptrice émet une trame ACK qui, une fois reçue par l'émetteur, met un terme au processus. Si la trame ACK n'est pas détectée par la station émettrice (parce que le paquet original ou le paquet ACK n'a pas été reçu intact), une collision est supposée et le paquet de données est retransmis après attente d'un nouveau temps aléatoire.

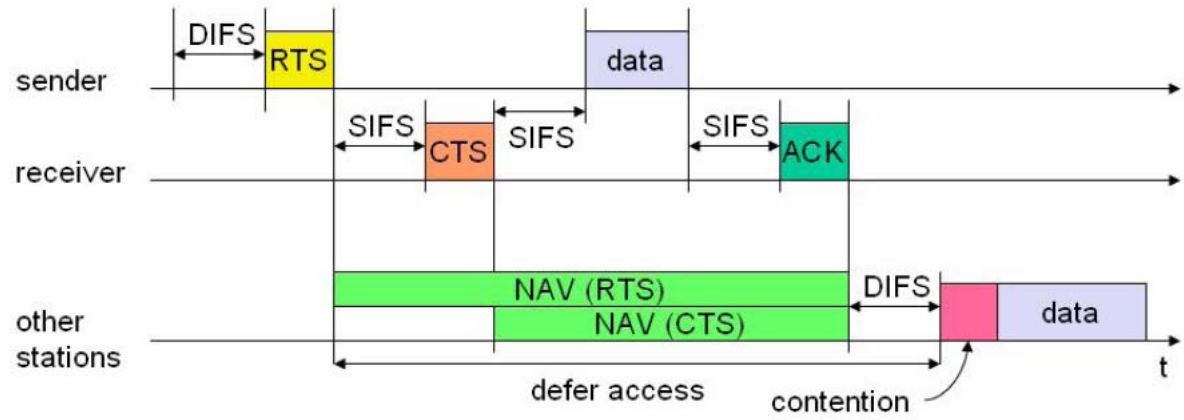


Figure 1.1 Le fonctionnement de CSMA/CA avec RTS/CTS [Van, 06]

CSMA/CA permet donc de partager l'accès aux ondes. Ce mécanisme d'accusé de réception explicite gère aussi très efficacement les interférences et autres problèmes radio.

Autre problème de la couche MAC, spécifique au sans fil, celui du "nœud caché", où deux stations situées de chaque côté d'un point d'accès peuvent entendre toutes les deux une activité du point d'accès, mais pas de l'autre station, problème généralement lié aux distances ou à la présence d'un obstacle. Pour résoudre ce problème, le standard 802.11 définit sur la couche MAC un protocole optionnel de type RTS/CTS *Request to Send*/*Clear to Send*. Lorsque cette fonction est utilisée, une station émettrice transmet un RTS et attend que le point d'accès réponde par un CTS. Toutes les stations du réseau peuvent entendre le point d'accès, aussi le CTS leur permet de retarder toute transmission prévue, la station émettrice pouvant alors transmettre et recevoir son accusé de réception sans aucun risque de collision. Du fait que le protocole RTS/CTS ajoute à la charge du réseau comme le montre la figure 1.1 en réservant temporairement le support, il est généralement réservé aux plus gros paquets, dont la retransmission s'avérerait lourde du point de vue de la bande passante.

1.3.1. Les modes opératoires de IEEE 802.11

Le standard 802.11 concerne deux types d'équipements, une station sans fil, en général un PC équipé d'une carte réseau sans fil, et un point d'accès (AP), qui joue le rôle de pont entre le réseau filaire et le réseau sans fil. Ce point d'accès se compose habituellement d'un émetteur/récepteur radio, d'une carte réseau filaire comme par exemple IEEE 802.3 et d'un logiciel de pontage conforme au standard 802.1d. Le point d'accès se comporte comme la station de base du réseau sans fil, agrégeant l'accès de multiples stations sans fil sur le réseau filaire.

Le standard 802.11 définit deux modes : un mode infrastructure et un mode ad hoc.

1.3.1.1. Le mode infrastructure

Le réseau sans fil consiste au minimum en un point d'accès connecté à l'infrastructure du réseau filaire et un ensemble de postes réseaux sans fil. Cette configuration est baptisée *Basic Service Set* (BSS). Un *Extended Service Set* (ESS) est un ensemble d'au moins deux BSS formant un seul sous-réseau. En entreprise, la plupart des WLAN devront pouvoir accéder aux services pris en charge par le LAN filaire (serveurs de fichiers, imprimantes, accès Internet).

1.3.1.2. Le mode ad hoc

Le mode ad hoc *Independent Basic Service Set* (IBSS) représente simplement un ensemble de stations sans fil 802.11 qui communiquent directement entre elles sans point d'accès. Ce mode permet de créer rapidement et simplement un réseau sans fil là où il n'existe pas d'infrastructure filaire.

1.3.2. Les algorithmes d'adaptation du débit physique

La transmission des ondes est susceptible à beaucoup de phénomènes physiques que nous avons listé dans la section 1.2. Cependant, les variations de ses conditions de transmission peuvent être classé en deux catégories selon leur durée : l'une qui soit rapide comme la fermeture d'un porte, ou le déplacement d'un grand objet, et l'autre qui dure dans le temps comme se déplacer d'une chambre vers une autre. Ces perturbations auront toujours un effet sur l'énergie du signal radio, et dans la majorité des cas elles augmentent le BER

Dans cette partie, nous présentons quelques algorithmes d'adaptation du débit de transmission physique en fonction des conditions du canal.

1.3.2.1. ARF

ARF [Kamerman et al., 97] *Auto Rate Feedback*, est l'un des premiers algorithmes publiés. Chaque station essaie d'augmenter son débit de transmission physique après un certain nombre de transmissions avec succès et de diminuer ce débit en cas d'un ou deux échecs successifs. Plus spécifiquement, ARF augmente le débit de transmission après dix transmissions successives avec succès et il le diminue lors de deux échecs successifs ou bien lors du premier échec juste après une augmentation du débit physique.

1.3.2.2. RBAR

RBAR [Holland et al., 01] *Receiver Based Auto Rate* a pour but d'optimiser le débit au niveau application. Cet algorithme exige l'échange des trames RTS/CTS entre la station émettrice et la station réceptrice. Cette dernière calcule le débit de transmission des données à l'aide du débit avec lequel a été envoyée la trame RTS et aussi à l'aide des valeurs de SNR des trames de données déjà reçus. Le débit de transmission PHY de la prochaine trame de donnée est envoyé avec la trame CTS. La figure 1.2 montre la performance de RBAR en le comparant avec ARF.

L'inconvénient de RBAR est qu'il exige que les toutes les stations écoutent les trames RTS et CTS afin de mettre à jour leur *Network Allocation Vector* (NAV) et aussi des modifications dans l'entête MAC IEEE 802.11.

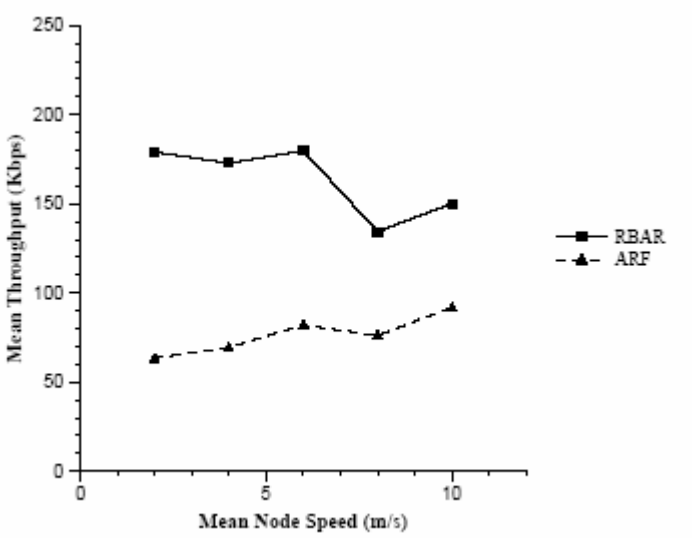


Figure 1.2 Le débit de transmission physique RBAR et ARF [Holland, 01]

1.3.2.3. AARF

AARF [Lacage et al., 04] *Adaptative Auto Rate Feedback* est une amélioration de l'algorithme ARF. Ce dernier est la bonne solution pour un environnement où il y'a beaucoup de mouvement. Mais, pour un environnement stable, le protocole ARF essaye périodiquement d'augmenter le débit alors que le débit de transmission actuel est le meilleur. AARF introduit de nouvelles variables pour minimiser le nombre de d'échecs suite à une augmentation du débit physique. AARF propose de doubler le nombre de transmissions avec succès nécessaire pour augmenter le débit de transmission physique lors d'un échec suite à une augmentation du débit. La figure 1.3 compare l'évolution du débit de transmission physique de deux stations utilisant les deux algorithmes ARF et AARF. Nous constatons que AARF minimise le nombre de pertes liées à une augmentation du débit.

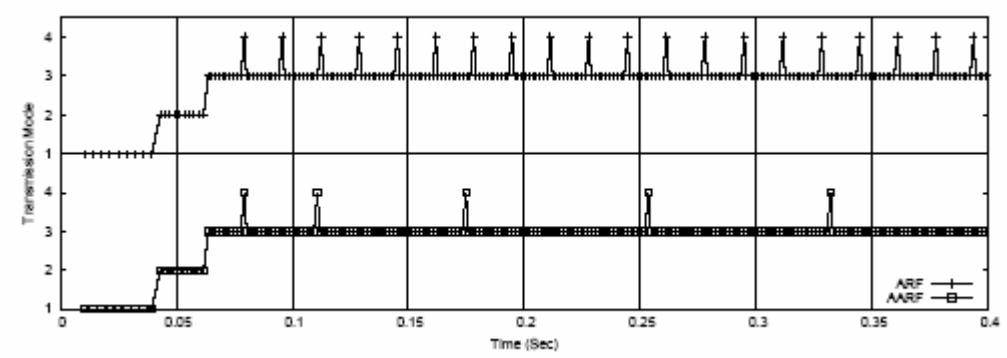


Figure 1.3 Comparaison entre les deux approches ARF et AARF [Manshaei, 05]

La figure 1.4 représente la courbe débit en fonction de la distance, résultat des simulations réalisées au sein de l'équipe PLANETE à l'aide du simulateur NS-2. Elle montre l'apport de l'algorithme AARF par rapport aux autres algorithmes déjà cités.

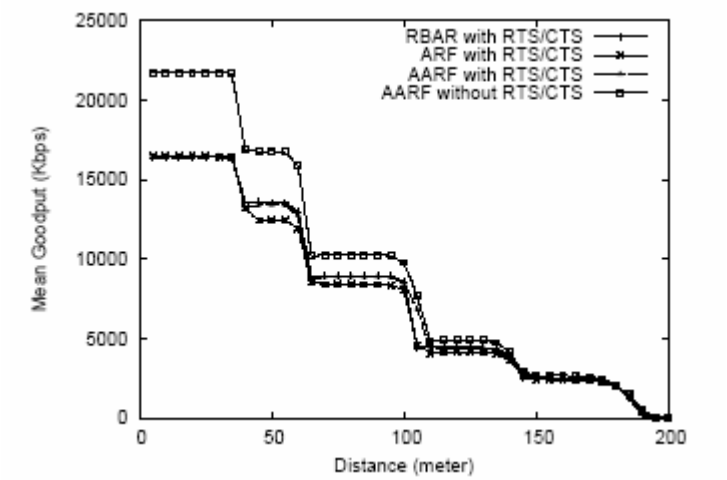


Figure 1.4 Le débit moyen avec les trois différents algorithmes [Manshaei, 05]

2. La transmission video dans les réseaux IEEE 802.11

La deuxième partie de ce chapitre est consacrée à l'étude de la transmission vidéo dans les réseaux IEEE 802.11. Nous commençons par la présentation du protocole RTP puis nous décrivons le protocole RTCP. Ensuite, nous présenterons l'état de l'art de la transmission multipoint dans les réseaux IEEE 802.11.

2.1. La couche RTP

Les applications temps réel sont soumises à des contraintes de temps strictes qui rendent impossible l'utilisation de *Transport Control Protocol* (TCP). Le protocole *Real-time Transport Protocol* [Schulzrinne, et al., 03] (RTP) essaye de palier aux inconvénients des protocoles TCP et *User Datagram Protocol* (UDP). En effet, les mécanismes du protocole TCP, tels que la fiabilité par acquittement et retransmission des paquets manquants, le contrôle de flux par la fenêtre de congestion (slow start), sont incompatibles avec les applications temps réel. En plus les paquets UDP se trouvent dépourvus de certaines informations comme le numéro de séquence.

2.1.1. Le protocole Real-time Transport Protocole (RTP)

Le protocole RTP [Schulzrinne, et al., 03] assure une numérotation et un horodatage des paquets en encapsulant les données dans une nouvelle entête.

Il repose lui-même sur le protocole UDP, qui ne renvoie pas les trames perdues en cours de route, afin de privilégier l'enchaînement du son et des images, plutôt que l'intégrité des données. La qualité peut en souffrir si la liaison est mauvaise, mais c'est un mal nécessaire à la diffusion en direct. Dans le cadre même de notre projet, deux possibilités se présentent. La première consiste à envoyer simultanément un flux de données vers chacun des clients. L'autre solution est le multipoint qui se caractérise par la diffusion d'un flux unique vers un groupe multipoint, dont les clients lisent simultanément les mêmes informations. Dans l'objectif d'avoir une optimisation de la consommation de la bande passante, il est préférable d'utiliser une diffusion en multipoint. Le protocole RTP va ainsi pouvoir assurer les fonctionnalités suivantes :

- reconstituer la base de temps des flux temps réel,
- détecter les pertes de paquets et en informer la source.

Mais le protocole RTP ne permet pas de réserver les ressources dans le réseau, d'assurer une fiabilité et de garantir le délai de livraison.

2.1.2. Le protocole Real-time Transport Controle Protocole (RTCP)

Le protocole RTP est complété par le protocole RTCP qui transmet périodiquement des paquets de contrôle à tous les participants d'une session. Il adopte le même mécanisme de transmission que le protocole RTP.

2.1.2.1. Les fonctions de RTCP

Selon [Mél, 01], le protocole RTCP remplit trois fonctions principales plus une optionnelle. Il retourne les statistiques *feedback* sur la qualité de réception des données transmises dans les paquets RTP. Ces informations temps réel peuvent être exploitées par la

source pour adapter le type de codage au niveau des ressources disponibles. RTCP transporte un identificateur unique de la source RTP, appelé Nom Permanent *Canonical NAME* (CNAME) car l'identifiant SSRC peut changer et ne permet pas, à lui tout seul, d'identifier une source. Les destinataires peuvent avoir besoin du nom permanent, CNAME, pour associer plusieurs flux de données d'un même participant dans un ensemble de sessions RTP, par exemple pour une synchronisation audio-vidéo.

De plus, la réception des *feedbacks* et la connaissance du CNAME supposent que tous les participants envoient des paquets RTCP régulièrement. Ces informations servent à ajuster le débit de sortie et à l'adapter de manière à accommoder toutes les personnes désirant se joindre à l'évènement.

Enfin, la quatrième fonction qui est optionnelle, consiste à transmettre des informations permettant une régulation minimale des membres. RTCP assure l'identification des participants à une conférence en jouant le rôle d'un canal de liaison entre des participants fluctuants, et a priori non identifiés.

2.1.2.2. Les paquets RTCP

Plusieurs types de paquets sont définis, de manière à transporter une grande variété d'informations de contrôle :

- **Rapport d'émetteur (SR)** : ce paquet regroupe un ensemble de statistiques de transmission et de réception en provenance des participants qui sont des émetteurs actifs.

- **Rapport de récepteur (RR)** : c'est un paquet regroupant un ensemble de statistiques en provenance de participants qui sont des récepteurs.

- **Description de source (SDES)** : les paquets de description de source comprennent plusieurs éléments, dont le CNAME. Ils établissent une véritable carte de visite de la source.

- **BYE** : c'est un paquet qui indique le départ d'un participant de la session.

- **APP**: c'est un paquet qui permet d'ajouter des fonctions spécifiques à l'application.

Chaque paquet RTCP commence par une partie fixe, semblable à celle du paquet de données RTP, suivie par des éléments structurés, de taille variable, selon le type du paquet, mais qui se concluent toujours par une terminaison de 32 bits. Plusieurs paquets RTCP peuvent être concaténés, sans adjonction de séparateurs, pour former un paquet RTCP composite.

Ces paquets RTCP doivent respecter et tenir compte d'un certain nombre de paramètres essentiels pour assurer le bon déroulement de la session RTP. Parmi ces paramètres :

- La fréquence de transmission des paquets RTCP : lors d'une conférence de plusieurs centaines de membres, les flux RTP sont par nature autolimitatifs car une ou deux personnes seulement parlent en même temps. En revanche, le flot de contrôle ne s'autolimite pas, bien au contraire. Si les rapports de réception de chaque participant sont toujours envoyés selon la même périodicité, le trafic de contrôle va croître de façon linéaire avec le nombre de participants. Il convient donc de limiter cette fréquence en réduisant le trafic de contrôle à une

fraction minime et connue de la bande passante. A cet effet, il est suggéré de n'allouer à RTCP que 5% au plus de la bande passante globale de la session.

- La mise à jour du nombre de participants : Le calcul de l'intervalle entre les rapports RTCP dépend d'une estimation du nombre de sites participant à la session. Les nouveaux membres sont ajoutés dès leur détection, dans une base indexée par l'identificateur SSRC ou CSRC de leurs en-têtes RTP. Un membre peut être supprimé de la liste quand le paquet RTCP BYE, avec l'identifiant correspondant, est reçu.

- Allocation de bande passante pour la description de source SDES : Plusieurs types de description de sources (SDES) sont prévus, en plus des mentions obligatoires contenues dans le nom permanent CNAME, tels que NAME (nom de la personne) et EMAIL (adresse email). Les descriptions de sources SDES permettent également de définir des types de paquets RTCP spécifiques à une application. Les applications doivent prendre des précautions, en allouant de la bande passante de contrôle à ces informations additionnelles. Elles peuvent ralentir la vitesse d'émission des rapports de réception et des CNAME, diminuant ainsi les performances du protocole. Il est recommandé de limiter à 20% la part de la bande passante de contrôle consacrée au transport des informations complémentaires SDES.

2.2. La transmission multipoint sur IEEE 802.11

La transmission multipoint est une technique qui nous permet de gagner de façon remarquable dans les transmissions multimédia surtout que les besoins des nouvelles applications augmentent avec l'apparition du WiMAX.

Pourtant, le standard IEEE 802.11 ne supporte pas de mécanismes spécifiques pour la transmission multipoint. Cette transmission se fait de la même manière que dans la diffusion. L'utilisation du standard pose trois problèmes principaux qui sont:

Le contrôle de congestion : Sans avoir un mécanisme de *feedback*, le contrôle de congestion est impossible pour les flux multipoint surtout que ce flux sera en concurrence avec d'autres flux point à point. En plus, en cas de collision dans les réseaux IEEE 802.11, on double la taille de la fenêtre de contention. Donc, sans un mécanisme de contrôle de congestion, on ne peut pas adapter le débit de transmission.

La fiabilité de transmission : La plupart des applications multimédia tolèrent la perte des paquets dans le réseau, mais en cas de grande perte de paquets la qualité de transmission se détériorera rapidement.

L'adaptation du débit physique de transmission : Pour optimiser la qualité de transmission d'un flux multimédia, les réseaux IEEE 802.11 doivent adapter dynamiquement leur débit physique de transmission. Un grand nombre de mécanismes ont été proposés comme RBAR, ARF et AARF. Mais ces mécanismes ne sont pas adoptés pour la transmission multipoint. En général, les points d'accès utilisent des débits fixes pour la transmission multipoint.

2.2.1. Le proyocol LEP (Leader Election Protocol)

La solution la plus adéquate pour la résolution des trois problèmes identifiés dans la transmission multicast avec la norme 802.11 est de la faire de la même manière que la transmission unicast avec l'utilisation d'une approche basée sur le Leader, c'est-à-dire qu'une station parmi les récepteurs prend la responsabilité d'acquitter les trames multipoint. Les trames de contrôle serviront comme déclencher d'une retransmission possible, d'une

adaptation de la fenêtre de contention ou bien pour d'une sélection du débit de transmission physique.

Le protocole *Leader Election Protocol* (LEP) [Seok et al., 06], celui le plus récent, sélectionne dynamiquement la station qui a la plus mauvaise condition comme Leader. Le mécanisme LEP est basé sur les paquets IGMP et consiste en quatre phases :

Phase de Collection : Pour pouvoir sélectionner le *Leader*, LEP doit avoir une vue globale sur les conditions de chaque entité réceptrice du réseau multipoint. Pour se faire, chaque récepteur envoie périodiquement IGMP *Membership Report* (MR) incluant l'indicateur SINR dans le champ MRT. Quand le point d'accès reçoit un IGMP MR avec une valeur non nulle dans le champ MRT, il s'assure que toutes les stations supportent le mécanisme LEP. Dans ce cas, le point d'accès enregistre les adresses du groupe multicast, les adresses MAC et les SINR des stations.

Phase d'élection : Une fois que le point d'accès reçoit les paquets IGMP MR, il choisit la station qui possède la valeur SINR minimale. Si le *Leader* ne correspond pas à la plus mauvaise station, alors le point d'accès envoie un IGMP *Group Specific Query* (GSQ) incluant le SINR ce la plus mauvaise station. Dans le cas où deux stations possèdent la même valeur minimale de SINR, le point d'accès met le bit duplicated à 1.Quand une station reçoit le paquet IGMP GSQ, elle vérifie l'adresse IP de l'émetteur. Si elle ne correspond pas à l'adresse IP du point d'accès alors elle considère la trame comme IGMP GSQ venant d'un autre routeur multicast. Ensuite, chaque station envoie un paquet IGMP MR. Enfin, chaque station passe à la phase de confirmation.

Phase de confirmation : A travers le champ MRT du IGMP GSQ, chaque récepteur multicast détermine l'ancien SINR de la nouvelle station élue. Puis, il compare sa propre valeur de SINR pendant la phase de collection avec celle du nouveau Leader élu. Donc, la stations qui aura le même SINR suivent deux alternatives suivant le duplicated bit. Si le duplicated bit vaut 0, elle envoie un IGMP MR additionnel avec le même SINR pour confirmer l'élection, avec le duplicated bit à 0. Sinon, c'est-à-dire que le duplicated bit vaut 1, elle envoie un IGMP MR additionnel avec un nombre aléatoire au lieu de SINR pour déclencher la réélection avec le duplicated bit du IGMP MR à 1. Si le point d'accès reçoit un IGMP MR avec la même valeur de SINR, il termine la procédure d'élection, sinon si le point d'accès ne reçoit pas IGMP MR, il renvoie un nouveau IGMP GSQ avec une confirmation, sinon il lance la phase de réélection.

Phase de réélection : Si le point d'accès reçoit un IGMP MR dont le duplicated bit est à 1, il ne change pas le SINR de cette station car le paquet IGMP MR ne servira que pour le déclenchement de la procédure de réélection du Leader. Ensuite, le point d'accès envoie un paquet IGMP GSQ additionnel. Cependant, le champ MRT de la trame IGMP GSQ est égal à une des valeurs précédentes des paquets IGMP MR avec le duplicated de GSQ est fixé à 0.

2.2.2. Le protocole LBMS (Leader Based Multicast Services)

Le protocole LBMS [Seok , 07] est une nouvelle approche basée aussi sur le mécanisme Leader. Il ajoute à LB-ARF de nouvelles trames de contrôle de congestion dans le réseau qui sont : LBMS request, LBMS report. Le Point d'accès sélectionne un Leader pour chaque flux multicast. Cette élection se base sur les rapports envoyés périodiquement par les stations qui peuvent contenir des informations sur la qualité de réception comme PER, SNR etc.

La trame LMBS request est envoyée par une station vers AP auquel elle est associée pour s'abonner à un ou plusieurs groupes multipoint.
La trame LBMS report est une nouvelle trame envoyée par une station vers le AP de façon périodique. Elle contiendra des statistiques sur sa performance tels que *Packet Loss Rate, SNIR*.
Les trames LBMS sont envoyées par le AP vers une station et vice versa.

2.2.3. Multicast PHY Rate Adaptation Mechanism

Nous décrivons un nouveau mécanisme pour adapter le débit de transmission physique des flux multipoint, appelé LB-ARF. Il est le modèle le plus simple, basé sur le mécanisme de Leader. Donc, le Leader, élu grâce à LEP, acquitte les trames reçues du AP. Ce dernier contrôle le débit physique de la même manière que ARF.

Quand le temporisateur expire ou bien la réception de dix acquittements consécutifs alors le débit de transmission passe au prochain niveau. Dans le cas contraire, on cas de perte de 2 trames successives, le débit de transmission physique diminue, c'est-à-dire le débit descend vers le niveau le plus bas, et le temporisateur se initialisera à 0.

Cependant, LB-ARF n'est pas apprécié dans les environnements mobiles tels que les réseaux Ad-hoc car les positions des stations varient de façon rapide.

3. Le simulateur OMNET ++

Dans ce projet, nous allons réaliser nos expérimentations à l'aide de OMNET++ [Omnet, 07] qui est un simulateur à évènements discrets orienté objet, basé sur C++. Il a été conçu pour simuler les systèmes réseaux de communication, les systèmes multi processeurs, et d'autres systèmes distribués. OMNET++ [Omnet, 07] est un projet open source dont le développement a commencé en 1992 par Andras Vargas [Vargas, 07] à l'université de Budapest. Actuellement, Ce simulateur est utilisé par des dizaines d'université pour la validation de nouveaux matériels et logiciels, ainsi que pour l'analyse de performance et l'évaluation de protocoles de communication.
L'avantage de OMNET ++ est sa facilité d'apprentissage, d'intégration de nouveaux modules et la modification de ceux déjà implémentés. Nous introduisons dans la suite l'architecture du simulateur OMNET++ et les librairies Mobility Framework [MF, 07] et INET [INET, 07].

3.1. Architecture de OMNET++

L'architecture de OMNET++ est hiérarchique composé de modules. Un module peut être soit module simple ou bien un module composé. Les feuilles de cette architecture sont les simples modules qui représentent les classes C++. Pour chaque module simple correspond un fichier .cc et un fichier .h. Un module composé est composé de simples modules ou d'autres modules composés connectés entre eux. Les paramètres, les sous modules et les ports de chaque module sont spécifiés dans un fichier .ned.

La communication entre les différents modules se fait à travers les échanges de messages. Les messages peuvent représenter des paquets, des trames d'un réseau informatique, des clients dans une file d'attente ou bien d'autres types d'entités en attente d'un service. Les messages sont envoyés et reçus à travers des ports qui représentent les interfaces d'entrer et de sortie pour chaque module.

La conception d'un réseau se fait dans un fichier .ned et les différents paramètres de chaque modules sont spécifies dans un fichier .ini. OMNET++ génère à la fin de chaque simulation deux nouveaux fichiers omnet.vec et omnet.sca qui permettent de tracer les courbes et calculer des statistiques.

3.2. La librairie MF

La librairie MF est une extension du simulateur OMNET++. Elle a été développée par une équipe de chercheurs à l'université de Berlin. Ca dernière version a été proposée par Marc Loebbers en Octobre 2003. Il est un bon support pour la simulation des réseaux sans infrastructure et mobile. Il peut être utilisé pour la simulation de :

- réseaux sans fils,
- réseaux mobiles,
- réseaux Ad-hoc.

3.3. La librairie INET

Dans ce paragraphe, nous allons présenter une étude de l'existant de la librairie INET. Nous allons décrire plus précisément l'implémentation des couches PHY, MAC, IP et RTP dans INET.

3.3.1. La couche PHY

Andras Varga s'est basée par l'implémentation de la couche PHY faite dans MF par Marc Löebbers [Löbbers et al., 07]. Dans la librairie INET version Octobre 2006 la couche physique IEEE 802.11b implémenté. Le calcul au niveau PHY de la puissance reçue se fait à l'aide de la formule de Friis, c'est-à-dire que seul le modèle Free-Space est implémenté comme modèle de propagation.

3.3.2. La couche MAC

La couche MAC IEEE 802.11b est implémentée dans la librairie INET, elle supporte le mécanisme RTS/CTS. Il n'y a que l'algorithme DCF implémenté, le PCF n'est pas implémenté. Le débit de transmission physique est constant pendant la simulation. Aucun des algorithmes d'adaptation du débit de transmission physique n'est implémenté. Les paquets multipoints au niveau MAC sont envoyés de la même manière que les paquets broadcast. Le filtrage de ces paquets se fait au niveau IP.

3.3.3. La couche IP

Le routage multicast au niveau IP est déjà implémenté mais de manière statique. C'est à dire que l'utilisateur doit configurer les adresses IP et les adresses de groupes multicast dans les tables de routages avant la simulation. C'est au niveau IP que se passe le filtrage des paquets multicast.

3.3.4. La couche RTP

La couche RTP n'est pas encore intégrée dans la librairie INET. Dans la version 20061020, il y a une implémentation de la couche RTP réalisé par Matthias Opptiz [Oppitz, 02] et ajouté par Andras Vargas mais elle n'est pas intégrée. Le problème ce cette implémentation est qu'elle a été faite dans une ancienne version de l'année 2001 et que l'architecture globale de la librairie INET a changé.

3.3.5. La couche Application

De nombreuses applications sont implémentés dans la librairie INET qui utilise soit le protocole UDP ou le protocole FTP.

Conclusion

Tout au long de ce chapitre, nous avons vu le standard IEEE 802.11 avec ses deux couches PHY et MAC. Nous avons aussi présenté les algorithmes d'adaptation du débit de transmission physique avec certaine comparaison entre ses différents algorithmes. Ensuite, nous avons expliqué les besoins des applications temps réels ainsi que les protocoles RTP et RTCP. Nous avons présenté deux approches permettant d'améliorer la qualité de transmission vidéo dans les réseaux IEEE 802.11. A la fin de ce chapitre nous avons présenté le niveau d'implémentation des différentes couches protocolaire du simulateur OMNET++.

Le chapitre suivant présentera une spécification de notre projet avec une analyse des besoins à l'aide de diagrammes de cas d'utilisation.

Chapitre 2 : Analyse & Spécification

Introduction

Après avoir présenté les notions de base nécessaires pour comprendre les fonctionnalités de notre projet, nous commençons ce chapitre avec une présentation de la problématique de notre projet. Ensuite, nous allons énoncer les spécifications de celui-ci en exposant les besoins fonctionnels et non fonctionnels du travail à réaliser. Enfin, nous enchaînons par la description des diagrammes de cas d'utilisation.

1. La problématique

Le but de ce projet est de simuler de nouvelles approches pour améliorer la qualité de la transmission vidéo dans les réseaux IEEE 802.11. Les simulations seront réalisées à l'aide d'une des plateformes du simulateur OMNET++. Comme nous avons déjà indiqué dans la partie état de l'art, les deux librairies MF et INET ne supporte pas la transmission multipoint. L'autre problème dans ce sujet est que les protocoles RTP et RTCP se sont pas implémentés dans la librairie INET.

2. Les besoins fonctionnels

2.1. Environnement physique

Ce travaille s'intègre dans le cadre du projet RNRT Divine. Ce projet propose l'étude, la conception et le développement d'un système réaliste et simple permettant de diffuser des flux image et vidéo vers des terminaux mobiles hétérogènes au travers de liens hétérogènes sans fils IP. Le projet RNRT Divine vise à étudier des solutions innovantes permettant, au sein d'un système hétérogène la diffusion de vidéo et images basées sur la détection, la gestion et le traitement adaptatif de cette hétérogénéité. Cela permettra d'assurer le transfert multimédia en qualité maximale vers l'utilisateur.

2.2. Les fonctionnalités

Les fonctionnalités définissent une description abstraite des services que le système est sensé fournir. Cette description ne spécifie que le comportement extérieur et n'a rien à voir avec les caractéristiques de conception. Cependant, elle doit être à la fois complète et consistante. La complétude signifie que tous les services requis par l'utilisateur sont spécifiés et la consistance signifie qu'il n'y a pas de besoins contradictoires. Nous allons dans ce qui suit présenter les besoins de notre projet.

2.2.1 Le besoins de simulation

Le simulateur doit permettre de connecter un réseau Ethernet IEEE 802.3 avec un ou plusieurs réseaux IEEE 802.11. Il doit aussi permettre le routage des paquets multipoint au niveau IP. Les nœuds dans un réseau IEEE 802.11 auront la possibilité de se déplacer pendant

la simulation. Le simulateur doit permettre de créer un ou plusieurs flux multipoint avec d'autre flux concurrents point à point.

2.2.2. Au niveau couche Physique

La couche physique du simulateur doit comprendre les deux standards IEEE 802.11b et IEEE 802.11a. L'utilisateur aura le choix entre les différents modèles de propagation Free-Space, Two-Ray, Shadowing et un modèle de fading. Le simulateur doit permettre la génération d'un fichier contenant les masques d'erreurs de transmission des paquets à la fin de la simulation.

2.2.3. Au niveau couche MAC

Le simulateur doit permettre aux utilisateurs de choisir entre la couche MAC IEEE 802.11a ou bien IEEE 802.11b. La couche MAC doit permettre au utilisateur le choix entre les différents algorithmes d'adaptation du débit physique ARF [Kamerman et al., 07] et AARF[Lacage et al. 04]. Le simulateur doit permettre le filtrage des paquets multipoint au niveau MAC. Le simulateur doit ajouter les nouvelles trames de gestion de réseau IEEE 802.11v[Seok et al., 07]. Le point d'accès permettra de lier un réseau IEEE 802.3 et un le réseau IEEE 802.11.

2.2.4. Au niveau RTP

Le simulateur doit implémenter les protocoles temps réels RTP et RTCP afin de permettre la simulation du vidéo streaming. L'implémentation des protocoles RTP/RTCP doit être conforme au RFC 3550 [].

2.2.5. Au niveau Application

Une application d'émission et de réception vidéo doit être implémentée dans le simulateur. Cette application émettrice prend comme paramètre un fichier contenant des trames vidéo à transmettre. Les trames vidéo peuvent être de type I, B et P. L'application réceptrice pourra reconstruire les trames envoyées par l'application émettrice.

3. Les besoins non fonctionnels

Le développement devra se conformer aux contraintes suivantes :

- Technologies utilisées : concernant l'implémentation, l'application sera développée en utilisant le langage C++ dans un environnement Linux.
- L'implémentation du fading au niveau couche physique sera réalisée à l'aide de la librairie IT++ [IT, 07].
- Méthode de conception : la conception devra utiliser la méthode UML. Nous allons décomposer la modélisation de l'application sous deux aspects : l'aspect du modèle statique et l'aspect du modèle dynamique. UML modélise la dynamique sous forme de trois types de diagrammes :
  - Diagramme de cas d'utilisation.
  - Diagramme de séquences.
  - Diagramme de classes.

4. Diagramme des cas d'utilisation

Les diagrammes de cas d'utilisation sont des vues externes du système. La fonctionnalité d'un cas d'utilisation peut se décomposer en un flot d'évènements. Les scénarios décrivent le comportement du système du point de vue utilisateur en termes d'actions et de réactions.

4.1. Définition du système

Notre système est le simulateur OMNET++ et plus précisément la librairie INET et la librairie MF. Ce système peut être aussi décomposé en plusieurs sous-systèmes qui seront les différentes couches protocolaires : la couche PHY, la couche MAC, la couche RTP. L'utilisateur de notre projet est l'utilisateur du simulateur qui peut être un étudiant, un chercheur, un ingénieur etc. Dans ce qui suit, nous allons décrire uniquement les nouvelles fonctionnalités que nous lui devons ajouté.

4.1.1 Le simulateur OMNET++

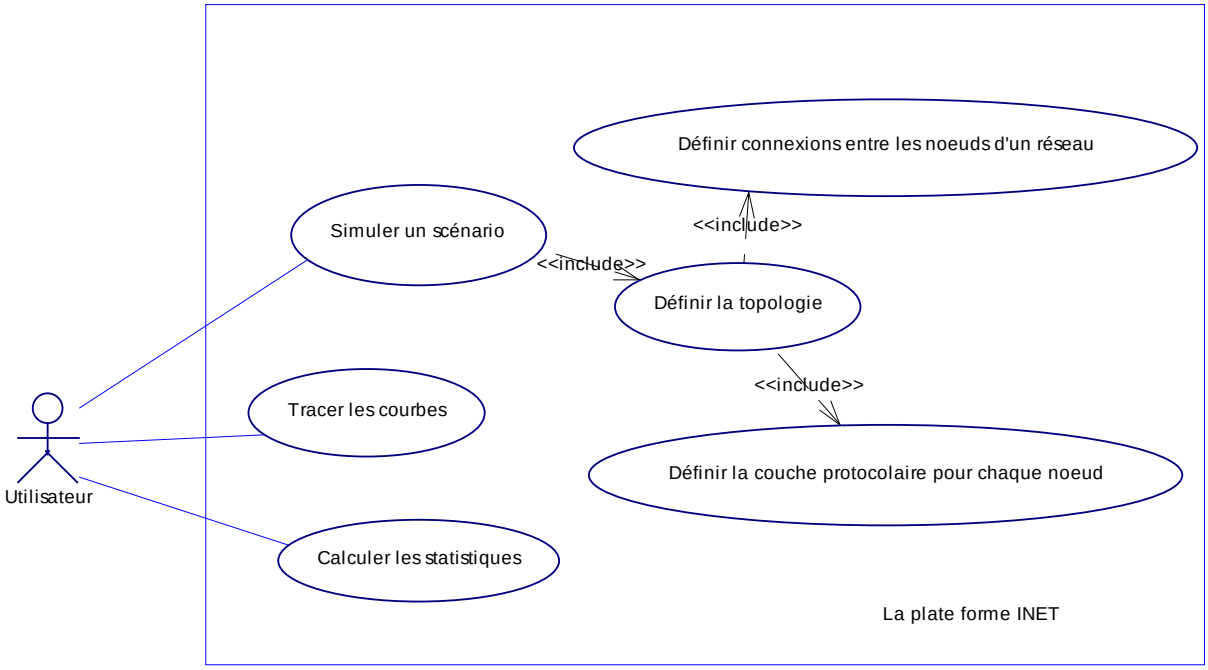


Figure 2.1 Les cas d'utilisation du simulateur OMNET++

Les scénarios :

- Définir la topologie
  - Créer un réseau avec des nœuds déjà développé dans le simulateur OMNET++.
  - Construire de nouveaux modules et les connecter.
- Définir la couche protocolaire
  - Utiliser les noeuds déjà existantes dans le simulateur OMNET++ en spécifiant de nouveaux paramètres dans le fichier omnetpp.ini ou bien dans le fichier .ned.
- Définir les connexions entre les différents nœuds d'un réseau
  - Spécifier les connexions avant la simulation dans le fichier .ned.
  - Connecter les modules du réseau de façon dynamiques lors de la simulation.

- Tracer la courbe
  - Tracer une courbe sur l'évolution d'un paramètre d'un module.
  - Tracer une courbe de statistiques sur différentes simulations
  - Superposer deux courbes dans un seul graphe pour faire les comparaisons
- Calculer les statistiques
  - Enregistrer les paramètres de performance de chaque module du réseau.
  - Faire des statistiques sur un ensemble de simulation.

4.1.2. La couche PHY

Nous allons décrire dans cette partie les nouvelles fonctionnalités que nous avons ajoutées au modèle physique du simulateur OMNET++.

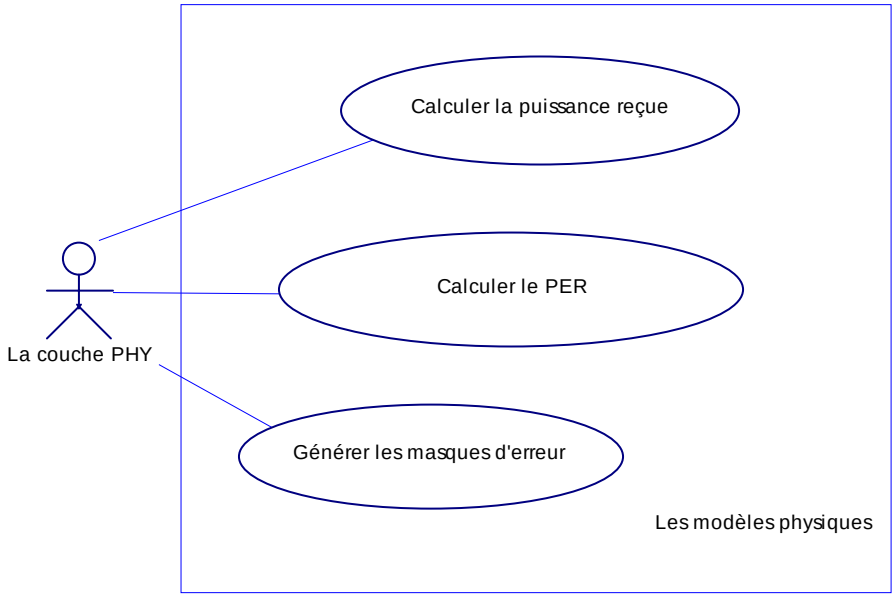


Figure 2.2 Les cas d'utilisation des modèles physiques

Les scénarios :

- Calculer la puissance reçue
  - Calcul à l'aide des modèles Free Space et fading.
  - Calcul à l'aide du modèle Two Ray.
  - Calcul à l'aide des modèles Shadowing Model et fading.
- Calculer le BER
  - Calculer le PER en fonction du modèle physique utilisé en supposant que le BER est uniforme.
  - Estimer le PER en ajoutant en utilisant [KSa06].
- Générer les masques d'erreur
  - Créer un fichier contenant les bits erronés.

4.1.3. La couche MAC

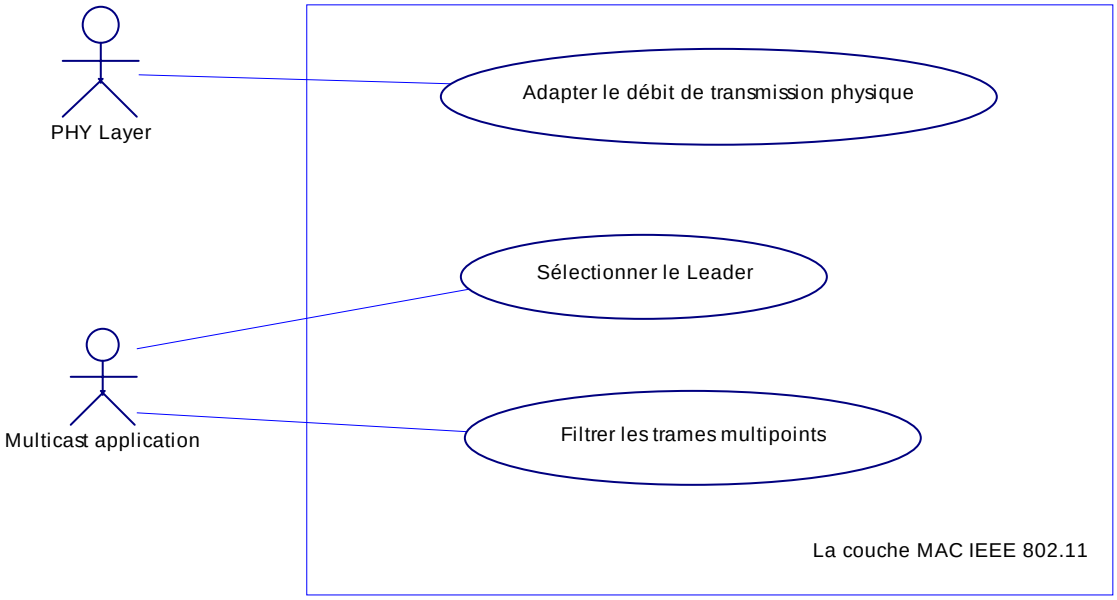


Figure 2.3 Les cas d'utilisation de la MAC IEEE 802.11

Les scénarios :

- Adapter le débit
  - Garder un débit physique de transmission constant.
  - Sélectionner le débit de transmission physique en utilisant l'algorithme ARF.
  - Sélectionner le débit de transmission physique en utilisant l'algorithme AARF.
- Sélectionner le Leader
  - Choisir une station non-AP connectée à un flux multicast comme Leader.
  - Informer une Station quelle n'est plus Leader.
  - Utiliser les trames LMBS pour choisir le Leader.
- Filtrer les trames multipoints
  - Ignorer ses trames et les traiter comme les trames broadcast et le filtrage se fait au niveau IP.
  - Supprimer les trames multipoints dont la station n'appartient pas à leur groupe.

4.1.4. La couche RTP

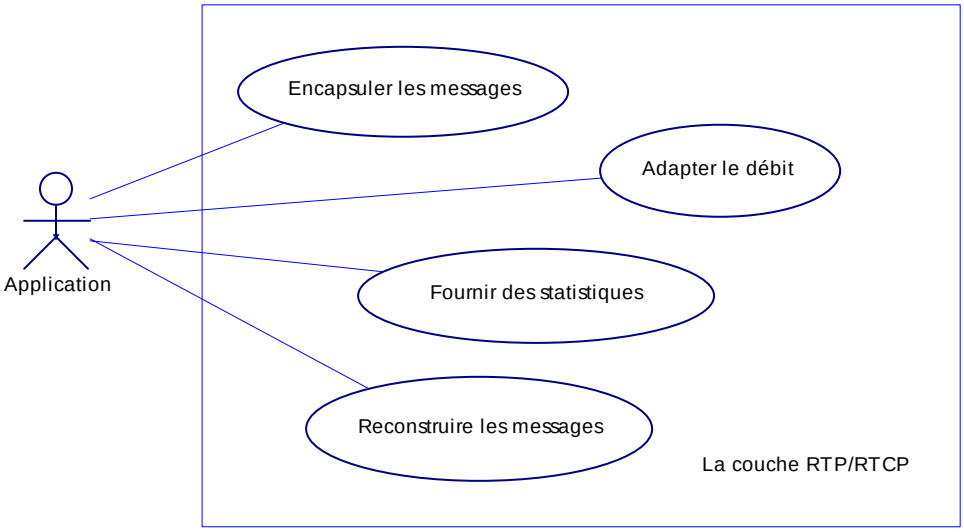


Figure 2.4 Les cas d'utilisation de la couche RTP

Les scénarios :

- Encapsuler les messages
  - Ajouter l'entête RTP au message de la couche application
  - Fragmenter les messages de la couche application en deux ou plusieurs messages RTP en ajoutant le timestamp dans les entêtes pour permettre au module RTP récepteur de reconstruire les messages.
- Fournir des statistiques
  - Calculer les taux de perte des paquets.
  - Calculer le débit et la gigue.
- Reconstruire les messages
  - Décapsuler les messages reçus de la couche UDP et les envoyer à la couche application.
  - Reconstruire les trames vidéo à partir des fragments des trames vidéo envoyées.

Conclusion

Après avoir listé les besoins fonctionnels et non fonctionnels de notre projet, nous allons détailler dans le chapitre suivant la conception de notre solution à travers les diagrammes de classes ainsi que les diagrammes de séquences.

Chapitre 3 Conception

Introduction

Ayant achevé la phase d'analyse, nous allons dégager la hiérarchie détaillée et complète des classes conçues tout en expliquant le rôle de chacune d'elles, puis nous expliquons les interactions entre elles avec les diagrammes de séquence.

1. L'architecture globale

Dans cette partie, nous allons présenter l'architecture globale des modules à développer dans ce projet. Il s'agit seulement de l'interface réseau IEEE 802.11 et de la couche RTP/RTCP.

1.1. L'interface réseau IEEE 802.11

Nous avons ajouté l'interface IEEE 802.11a aux deux librairies de OMNET++ MF et INET. Ce qui suit sera une description de l'architecture de cette interface.

1.1.1. L'interface réseau IEEE 802.11 de la librairie MF

L'interface réseau IEEE 802.11a que nous avons ajouté dans la librairie MF est un nouveau module appelé Nic80211a. Il est composé de 3 modules simples qui sont Decider80211a, SnrEcal80211a et Mac80211a. Comme le montre la figure 3.1, ce module possède deux interfaces de sortie avec la couche IP et avec le canal.

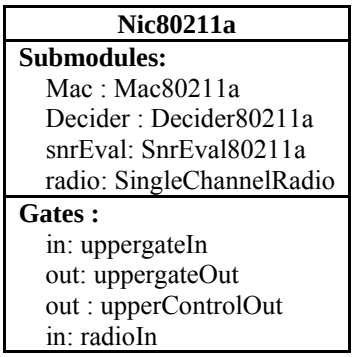


Figure 3.1 Le module Nic80211a

1.1.2. L'interface réseau IEEE 802.11 de la librairie INET

L'interface réseau IEEE 802.11a est un module composé de 3 modules simples qui sont Ieee80211aRadio, Ieee80211aMac et Ieee80211Mgmt. Dans la librairie INET, nous distinguons entre deux types de cartes réseaux sans fils : celle d'une station et celle d'un point d'accès. Cette différence est au niveau module Ieee80211Mgmt. Ce dernier gère les paquets

de gestion (authentification, association, etc.) qui sont échangés entre le point d'accès et les autres stations.

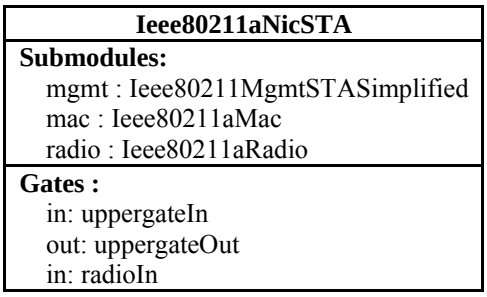


Figure 3.2 Les composants du module Ieee80211aNicSTA

1.2 La couche RTP/RTCP

Le module RTPLayer est composé de plusieurs modules qui sont : rtpModule, rtcpModule, RTPProfile, RTPPayloadReceiver et RTPPayloadSender. Ce module possède deux interfaces avec la couche Application et quatre interfaces avec la couche UDP l'une connecté avec le module simple rtpModule et l'autre avec rtcpModule.

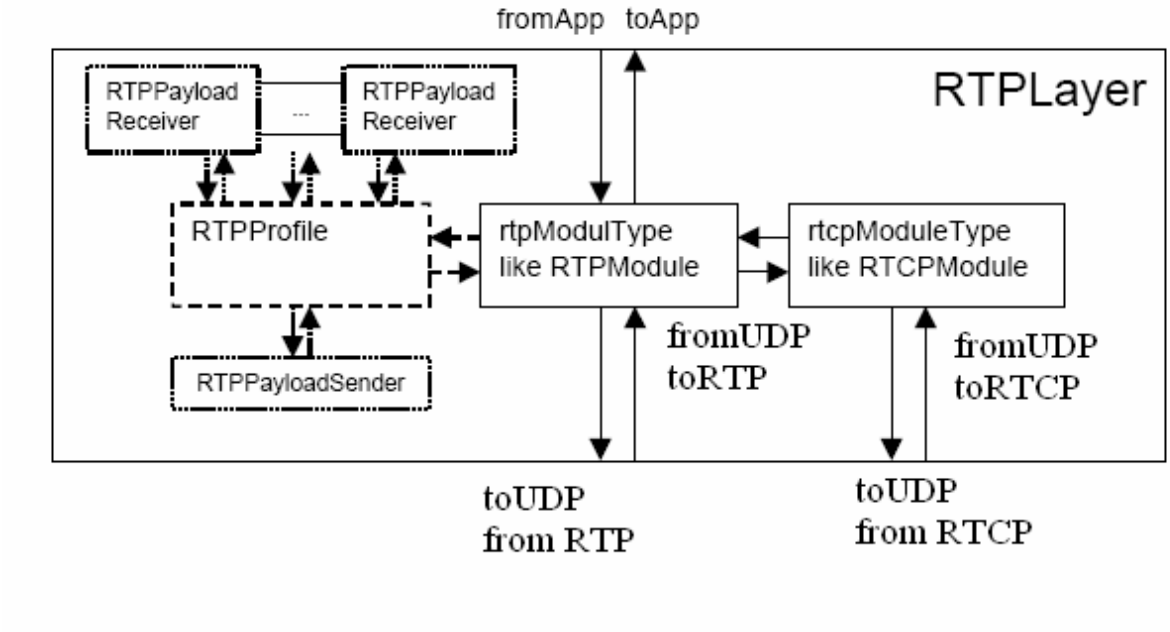


Figure 3.3 Les composants du module RTPLayer

2. Diagramme des classes

Apres avoir décrit l'architecture globale de notre projet, nous allons définir les diagrammes des classes suivant une modélisation UML.

2.1. L'interface réseau IEEE 802.11a de la librairie MF

La figure 3.4 représente le diagramme de classes de la couche physique de la librairie Mobility framework (MF).

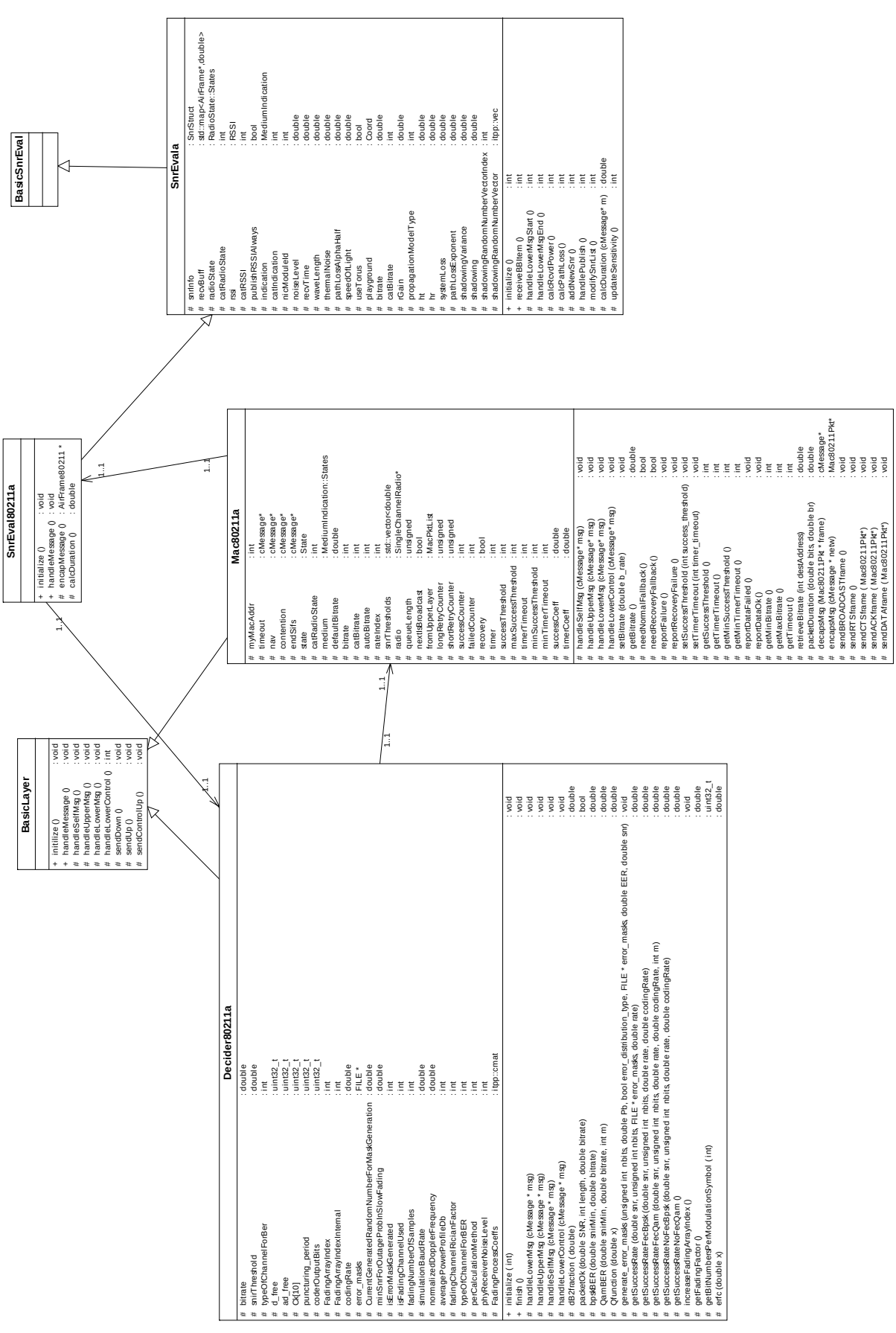


Figure 3.4 Diagramme de classes de la couche PHY de la librairie MF

2.1.1. Classe Mac80211a

La classe Mac80211a est une implémentation de la norme IEEE 802.11a. Nous nous sommes basés sur [Std00] pour extraire les différents paramètres comme le calcul de la durée de transmission packetDuration(), la taille la fenêtre de contention et les durées des périodes ST, SIFS, DIFS, EIFS.

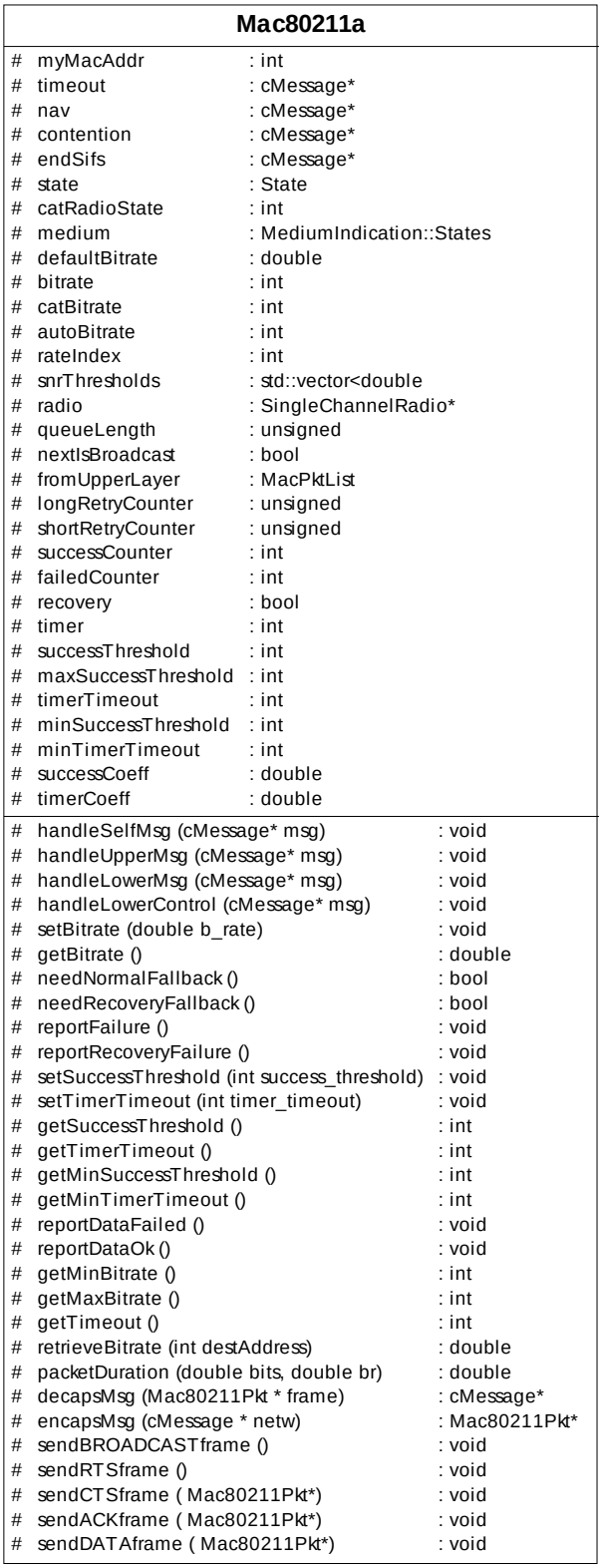


Figure 3.5 La classe Mac80211a

La classe Mac80211a est une sous classe de la classe BasicLayer de la librairie MF. Elle correspond au module simple MAC80211a. Parmi les paramètres du Module Mac80211a nous pouvons cité queueLength qui est la capacité de sa file d'attente, bitrate qui est le débit physique utilisé et autoBitrate qui peut avoir une des trois valeurs 0,1ou2. La valeur 0 du paramètre autoBitrate correspond à une transmission avec un débit physique constant, alors que la valeur 1 correspond à ARF et 2 à AARF. Mac80211a contient aussi les différents paramètres pour chacun de ces derniers algorithmes comme timerTimeout et successThreshold pour ARF, successCoeff timerCoeff et maxSuccessThreshold pour AARF.

2.1.2. Classe SnrEval80211a

La classe SnrEval80211a est une sous classe de SnrEvala. Elle permet de calculer la durée de transmission d'un paquet à l'aide de la méthode calcDuration() et aussi la puissance reçue calcRcvdPower(). Le calcul de la puissance reçue dépend du model de propagation spécifiée dans le module SnrEval80211a.

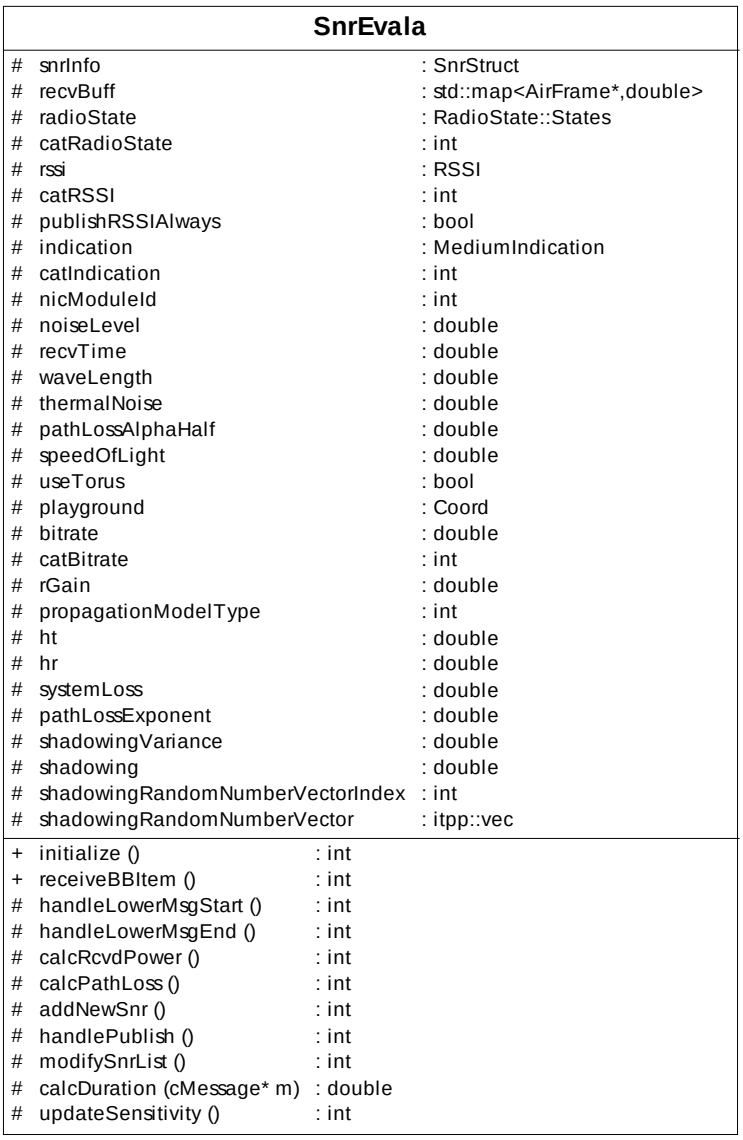


Figure 3.6 La classe SnrEvala

Le module SnrEval80211 correspond à la classe SnrEval80211a. Il a comme paramètre la taille de l'entête headerLength et le bruit thermique thermalNoise, etc. Nous

avons ajouté à ce module de nouveaux paramètres comme propagationModelType qui peut avoir trois valeurs possibles 0, 1, 2. La valeur 0 correspond au modèle Free-Space. La valeur 1 correspond au modèle Two-Ray. La valeur 2 correspond au modèle Shadowing. Nous avons aussi ajouté les paramètres de chaque modèle de propagation comme pathLossExponent, shadowingVaraince et shadowingNumberofSamples pour le modèle Shadowing.

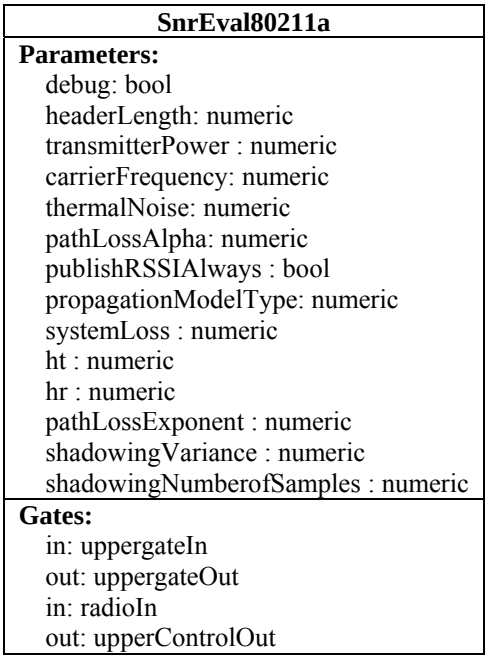


Figure 3.7 Le module SnrEval80211a

2.1.3. Decider80211a

Cette classe décide si le paquet est correctement reçu à l'aide de la méthode isReceivedCorrectly(). Cette dernière fait appel à la méthode packetOK() qui retourne true si le paquet est correctement reçu et false dans le cas contraire. La méthode packetOK() prend comme paramètre le débit avec lequel a été envoyé le paquet afin de connaître la modulation utilisée pour envoyée le *payload*. Selon la modulation, elle calcule la probabilité de succès des deux morceaux des paquets et ensuite le PER. Enfin, elle génère un nombre aléatoire et le compare au PER pour décider s'il y a eu des erreurs de transmission. Cette classe génère les masques d'erreur à l'aide de la méthode generate_error_masks(). L'appel à cette méthode se fait à chaque réception d'un nouveau paquet.

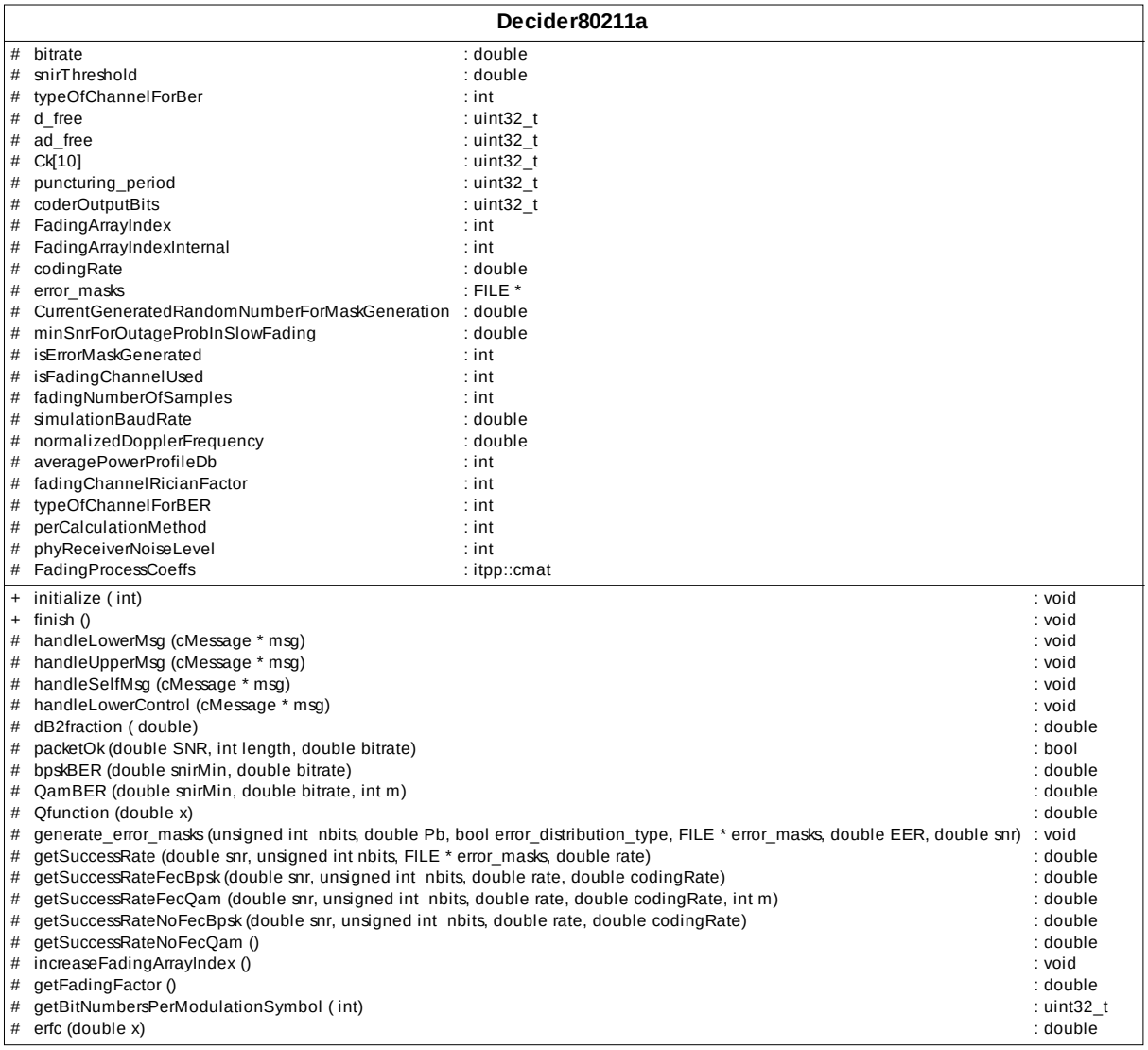


Figure 3.8 La classe Decider80211a

Le module Decider80211a reçoit les paquets du module SnrEval80211a et les transmet au module Mac80211a. Les paramètres principaux de ce modules sont : isErrorMaskGenerated qui vaut 1 s'il y a génération d'un fichier de masques d'erreur, isFadingChannelUsed qui vaut 1 si le modèle fading est utilisé. Ce module contient aussi d'autres paramètres permettant le calcul du BER et du PER.

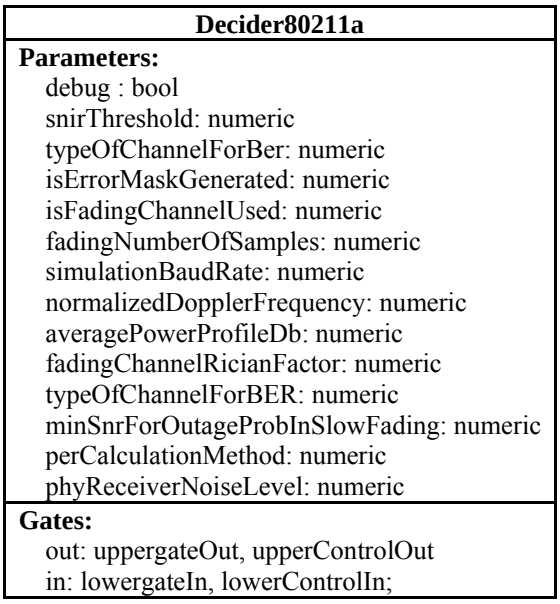


Figure 3.9 Le module Decider80211a

2.2. L'interface réseau IEEE 802.11a de la librairie INET

Après avoir bien expliqué les classes de la couche MAC et PHY de la librairie MF, nous allons décrire les classes de ces deux couches que nous avons ajoutées à la librairie INET. Nous commençons par la couche PHY, ses différentes classes et ensuite la couche MAC 802.11.

2.2.1 Le diagramme des classes de la couche PHY

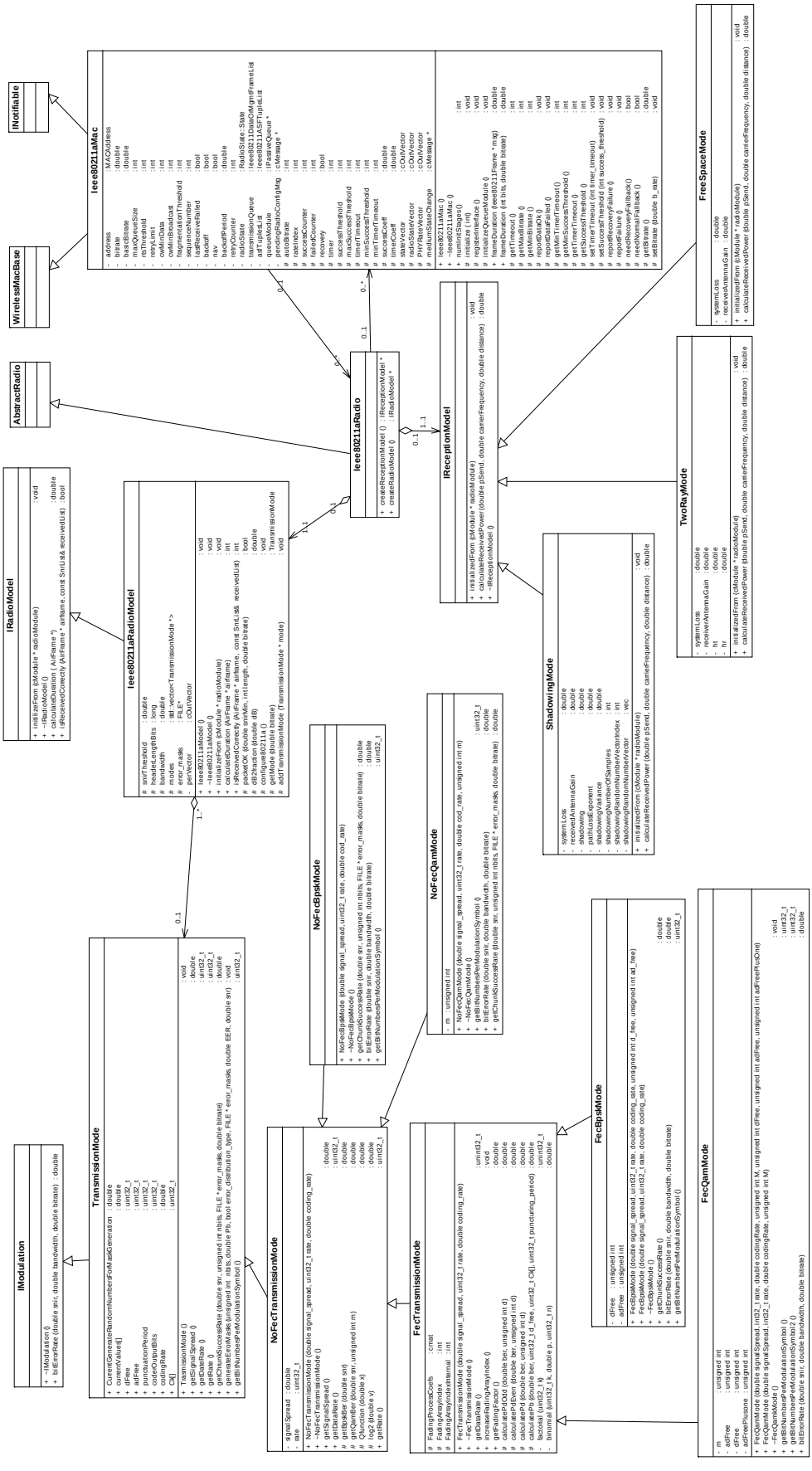


Figure 3.10 Diagramme de classes de la couche PHY de la librairie INET

2.2.1.1. Classe Ieee80211aRadio

Cette classe représente la couche physique IEEE 802.11. Elle est une sous classe de AbstractRadio. Elle calcule la puissance reçue à l'aide de la classe IReceptionModel et le PER à l'aide la classe Ieee80211aRadioModel.

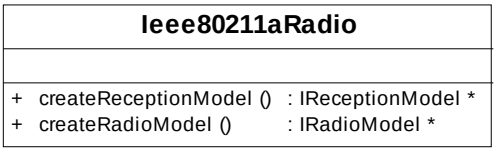


Figure 3.11 La classe Ieee80211aRadio

Le module Ieee80211aRadio possède comme paramètre bitrate le débit de transmission physique, transmitterPower la puissance émise et d'autre paramètre utilisé par les sous classes de la classe IReceptionModel pour le calcul de la puissance reçue.

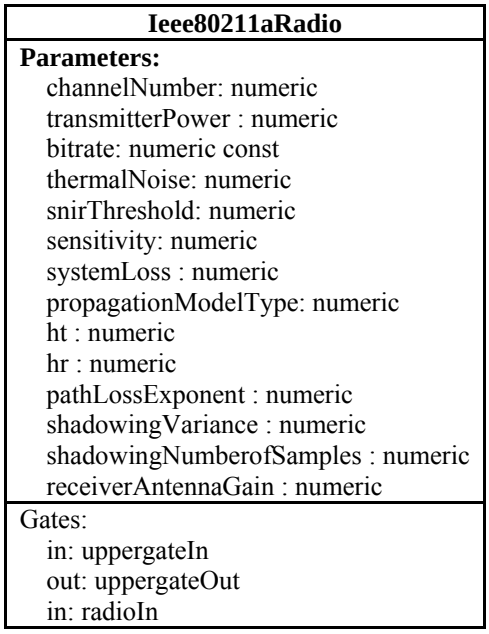


Figure 3.12 Le module Ieee80211aRadio

2.2.1.2. Classe Ieee80211aRadioModel

Cette classe permet de calculer la durée de transmission d'un paquet à l'aide de la méthode calculateDuration(). Elle prend aussi la décision si le paquet reçu est erroné ou pas à l'aide de la méthode isReceivedCorrectly(). Cette dernière fait appel a la méthode packetOK() qui retourne true si le paquet est correctement reçu et false dans le cas contraire. La méthode packetOK() prend comme paramètre le débit avec lequel a été envoyé le paquet a fin de connaître la modulation utilisé pour envoyé le *payload*. Suivant la modulation, elle calcule le PER. Enfin, elle génère un nombre aléatoire ente 0 et 1 et le compare avec PER pour décider s'il y a eu des erreurs de transmission.

Ieee80211aRadioModel		
# snirThreshold	: double	
# headerLengthBits	: long	
# bandwidth	: double	
# modes	: std::vector<TransmissionMode *>	
# error_masks	: FILE*	
- perVector	: cOutVector	
+ Ieee80211aModel ()		: void
+ ~Ieee80211aModel ()		: void
+ initializeFrom (cModule * radioModule)		: void
+ calculateDuration (AirFrame * airframe)		: int
+ isReceivedCorrectly (AirFrame * airframe, const SnrList& receivedList)		: int
# packetOK (double snirMin, int length, double bitrate)		: bool
# dB2fraction (double dB)		: double
# configure80211a ()		: void
# getMode (double bitrate)		: TransmissionMode
# addTransmissionMode (TransmissionMode * mode)		: void

Figure 3.13 La classe Ieee80211aRadioModel

2.2.1.3. IReceptionModel

Cette classe est une classe abstraite utilisée par le module Ieee80211aRadio pour le calcul de la puissance reçue. La méthode calculateReceivedPower() prend comme paramètre la puissance émise et la distance entre l'émetteur et le récepteur. Les nouveaux modèles de propagation tel que Free-Space, Two-Ray et Shadowing ont été construits à partir de cette classe.

IReceptionModel		
+ initializedFrom (cModule * radioModule)		: void
+ calculateReceivedPower (double pSend, double carrierFrequency, double distance)		: double
+ ~IReceptionModel ()		

Figure 3.14 La classe IReceptionModel

2.2.1.4. TransmissionMode

TransmissionMode est une classe abstraite permettant de générer un fichier masque d'erreurs et de calculer le BER. A partir de cette classe, nous avons créer de nouvelle classes contenant chacune leur propre implémentation des fonctions de calcul du BER.

TransmissionMode		
+ CurrentGenerateRandomNumbersForMaskGeneration	: double	
+ currentValues[]	: double	
+ dFree	: uint32_t	
+ adFree	: uint32_t	
+ punctuationPeriod	: uint32_t	
+ coderOutputBits	: uint32_t	
+ codingRate	: double	
+ Ck[]	: uint32_t	
+ TrasmissionMode ()		: void
+ getSignalSpread ()		: double
+ getDateRate ()		: uint32_t
+ getRate ()		: uint32_t
+ getChrunkSuccessRate (double snr, unsigned int nbits, FILE * error_masks, double bitrate)		: double
+ generateErrorMasks (unsigned int nbits, double Pb, bool error_distribution_type, FILE * error_masks, double EER, double snr)		: void
+ getBitNumbersPerModulationSymbol ()		: uint32_t

Figure 3.15 La classe TransmissionMode

2.2.2 Le diagramme des classes de la couche MAC

La figure 3.17 représente le diagramme de classe de la couche MAC de librairie INET. Nous avons ajouté trois nouvelles classes afin d'implémenter la norme IEEE 802.11a et l'algorithme d'élection du Leader pour la transmission multipoint améliorée.

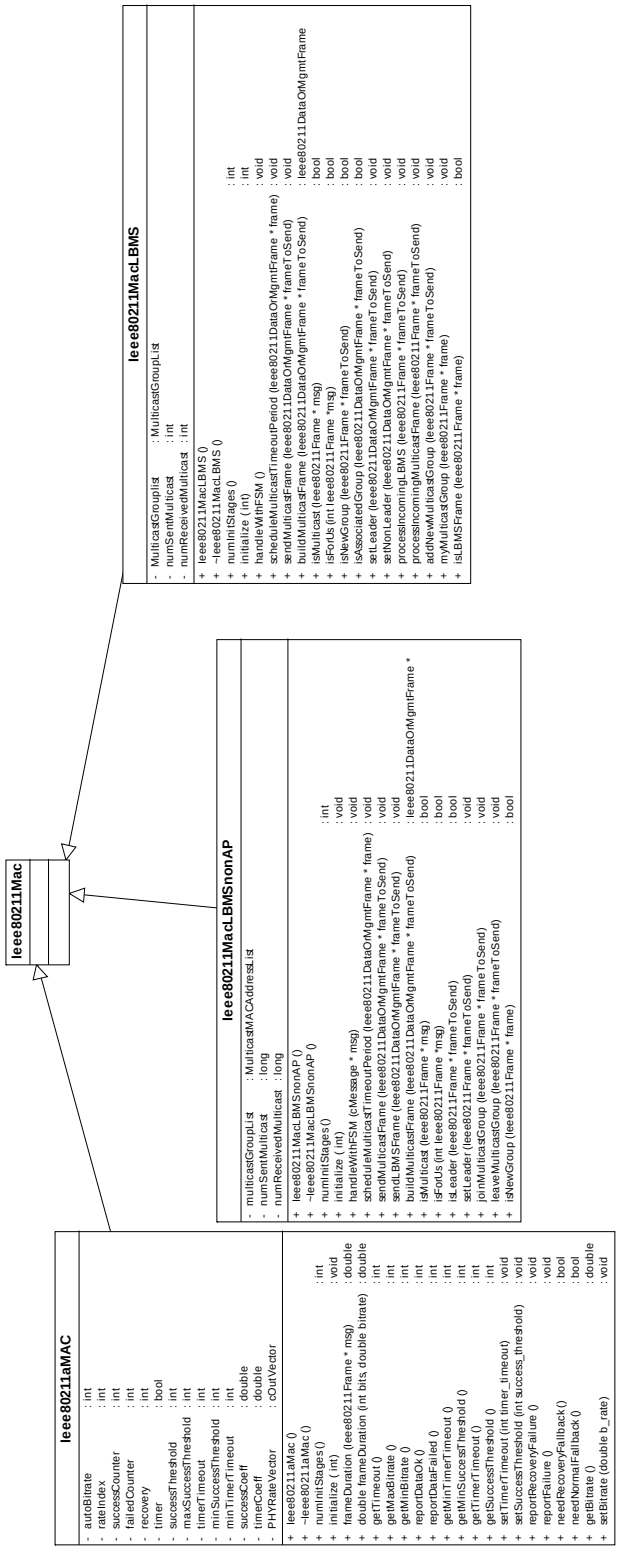


Figure 3.16 Le diagramme de classe de la couche MAC de INET

2.2.2.1. Classe Ieee80211aMAC

Cette classe implémente la norme IEEE 802.11a en modifiant quelques méthodes de la classe Ieee80211MAC comme le calcul de la durée de transmission. Nous avons ajouté aussi des méthodes et des structures qui permetant d'adapter le débit de transmission physique en fonction du nombre de transmission avec succès et du nombre de retransmission.

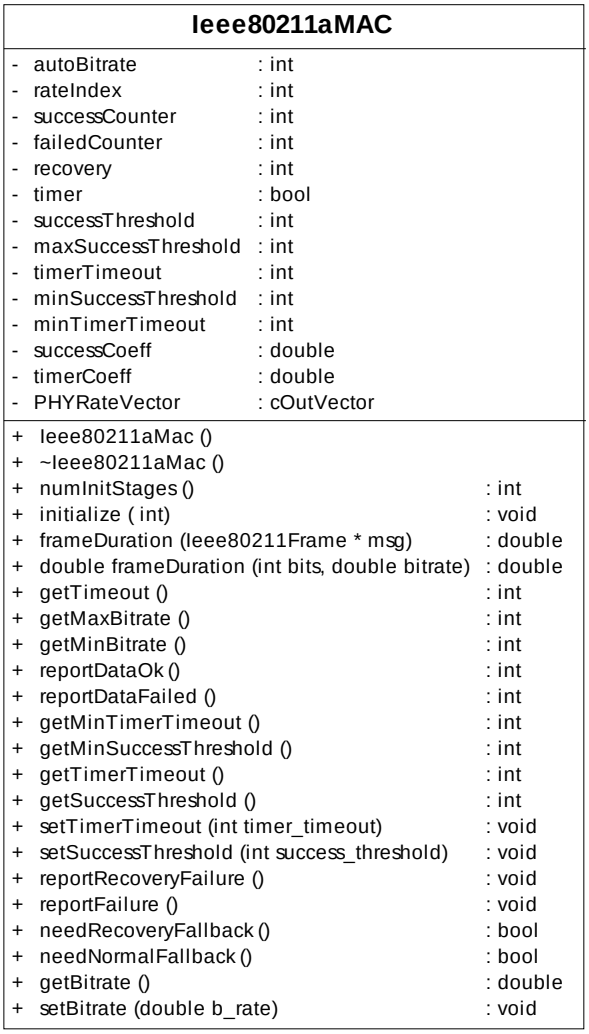


Figure 3.17 La classe Ieee80211aMAC

2.2.2.2. Classe Ieee80211MacLBMS

Cette classe correspond à l'implémentation de la couche MAC pour un point d'accès. Cette classe permet de faire le filtrage des trames multipoint reçues. Nous avons modifié dans cette classe la machine à états finis déjà implémentée. Nos avons ajouté un nouvel état appelé WAITMULTICAST permettant la station de distinguer les messages multipoint ses autres types de messages. Parmi les attributs que nous avons ajouté se trouve MulticastGroupList qui enregistre pour chaque adresse MAC multicast les adresses MAC des stations associées avec ce groupe et l'adresse MAC du Leader.

Ieee80211MacLBMS		
- MulticastGrouplist	: MulticastGroupList	
- numSentMulticast	: int	
- numReceivedMulticast	: int	
+ Ieee80211MacLBMS ()		
+ ~Ieee80211MacLBMS ()		
+ numInitStages ()		: int
+ initialize (int)		: int
+ handleWithFSM ()		: void
+ scheduleMulticastTimeoutPeriod (Ieee80211DataOrMgmtFrame * frame)		: void
+ sendMulticastFrame (Ieee80211DataOrMgmtFrame * frameToSend)		: void
+ buildMulticastFrame (Ieee80211DataOrMgmtFrame * frameToSend)		: Ieee80211DataOrMgmtFrame
+ isMulticast (Ieee80211Frame * msg)		: bool
+ isForUs (int Ieee80211Frame *msg)		: bool
+ isNewGroup (Ieee80211Frame * frameToSend)		: bool
+ isAssociatedGroup (Ieee80211DataOrMgmtFrame * frameToSend)		: bool
+ setLeader (Ieee80211DataOrMgmtFrame * frameToSend)		: void
+ setNonLeader (Ieee80211DataOrMgmtFrame * frameToSend)		: void
+ processIncomingLBMS (Ieee80211Frame * frameToSend)		: void
+ processIncomingMulticastFrame (Ieee80211Frame * frameToSend)		: void
+ addNewMulticastGroup (Ieee80211Frame * frameToSend)		: void
+ myMulticastGroup (Ieee80211Frame * frame)		: void
+ isLBMSFrame (Ieee80211Frame * frame)		: bool

Figure 3.18 La classe Ieee80211MacLBMS

2.2.2.3. Classe Ieee80211MacLBMSnonAP

La classe Ieee80211MacLBMSnonAP correspond à la couche MAC d'une station réceptrice d'un flux multipoint. Cette classe permet de filtrer les trames multipoint reçues. Elle enregistre les adresses groupes multicast avec lesquelles elle est associée. Pour chaque flux multipoint, si elle correspond au Leader, elle acquitte la trame multipoint reçue.

Ieee80211MacLBMSnonAP		
- multicastGroupList	: MulticastMACAddressList	
- numSentMulticast	: long	
- numReceivedMulticast	: long	
+ Ieee80211MacLBMSnonAP ()		
+ ~Ieee80211MacLBMSnonAP ()		
+ numInitStages ()		: int
+ initialize (int)		: void
+ handleWithFSM (cMessage * msg)		: void
+ scheduleMulticastTimeoutPeriod (Ieee80211DataOrMgmtFrame * frame)		: void
+ sendMulticastFrame (Ieee80211DataOrMgmtFrame * frameToSend)		: void
+ sendLBMSFrame (Ieee80211DataOrMgmtFrame * frameToSend)		: void
+ buildMulticastFrame (Ieee80211DataOrMgmtFrame * frameToSend)		: Ieee80211DataOrMgmtFrame *
+ isMulticast (Ieee80211Frame * msg)		: bool
+ isForUs (int Ieee80211Frame *msg)		: bool
+ isLeader (Ieee80211Frame * frameToSend)		: bool
+ setLeader (Ieee80211Frame * frameToSend)		: void
+ joinMulticastGroup (Ieee80211Frame * frameToSend)		: void
+ leaveMulticastGroup (Ieee80211Frame * frameToSend)		: void
+ isNewGroup (Ieee80211Frame * frame)		: bool

Figure 3.19 La classe Ieee80211MacLBMSnonAP

2.3. La couche RTP/RTCP

Nous sommes basés sur le travail de Mathias Oppitz [Opp, 02] pour implémenter la couche RTP. Comme nous allons expliquer ci-après, cette couche permet de construire des trames vidéo et les envoyés à une destination ou un groupe multipoint. La figure 3.21 représente le diagramme de classes de cette couche protocolaire.

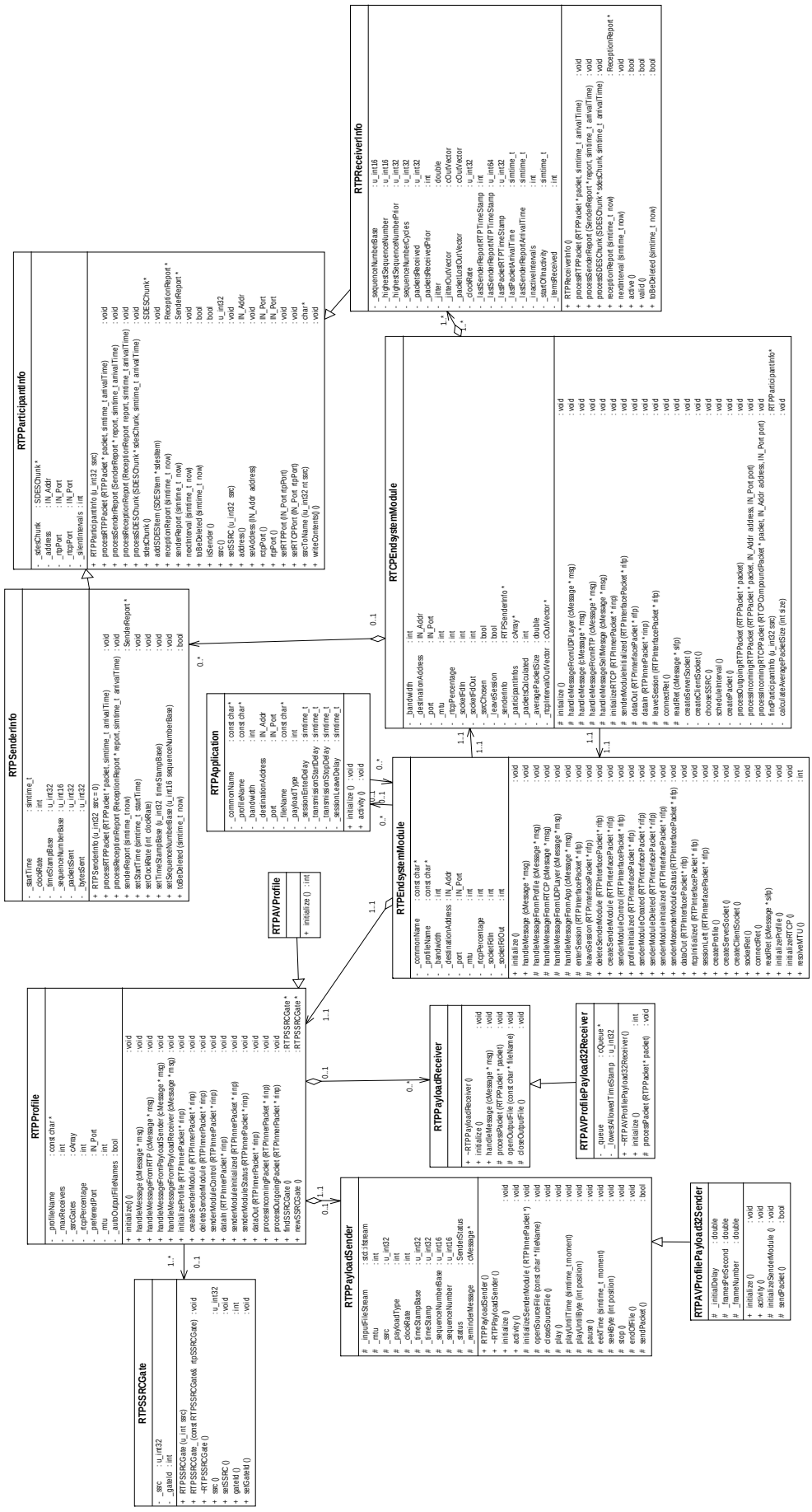


Figure 3.20 Diagramme de classe de la couche RTP/RTCP

2.3.1. RTPEndSystemModel

Cette classe représente le module RTP. Cette classe reçoit des messages de commandes (ouvrir une session, démarrer la transmission des données, quitter une session etc.) à partir du module RTPApplication. Elle ouvre une socket UDP au début d'une session, crée un profil et l'initialise. Au cours d'une session, elle decapsule les paquets RTP reçu de la couche UDP et les envoie au module RTPProfile et RTCPEndModule. A la fin de la session, elle ferme la socket et supprime le module Profile qu'elle a crée et informe le module RTCPEndSystemModule d'envoyer un paquet BYE.

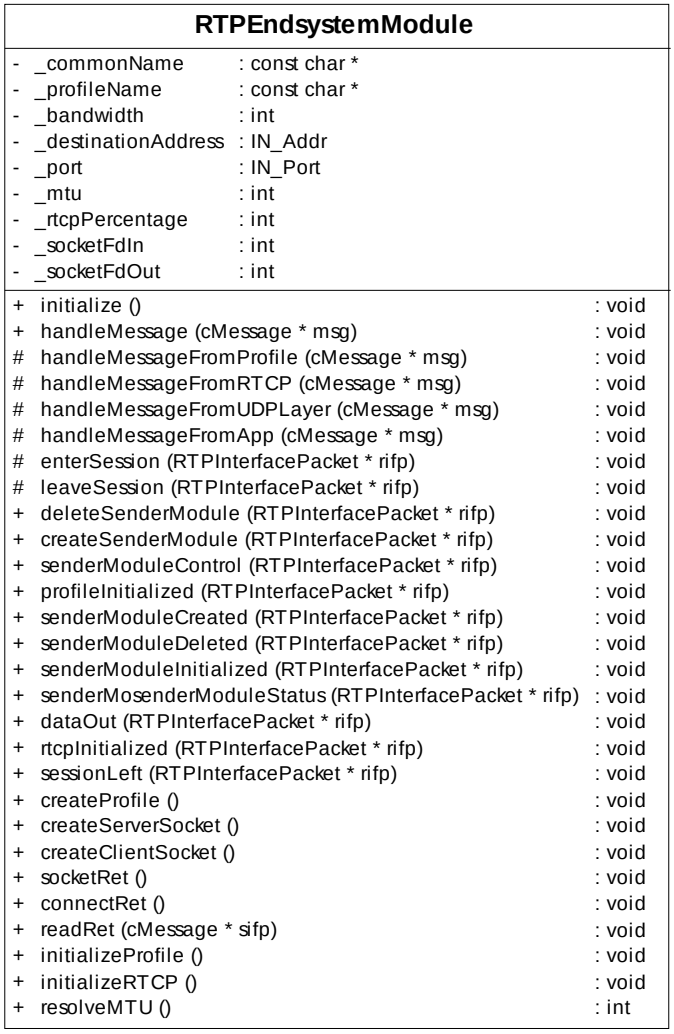


Figure 3.21 La classe RTPEndSystemModel

2.3.2. RTCPEndSystemModel

.

Cette classe ouvre une session RTCP, construit les paquets RTCP et traite les paquets RTCP reçues à partir des modules RTCP des autres participants.

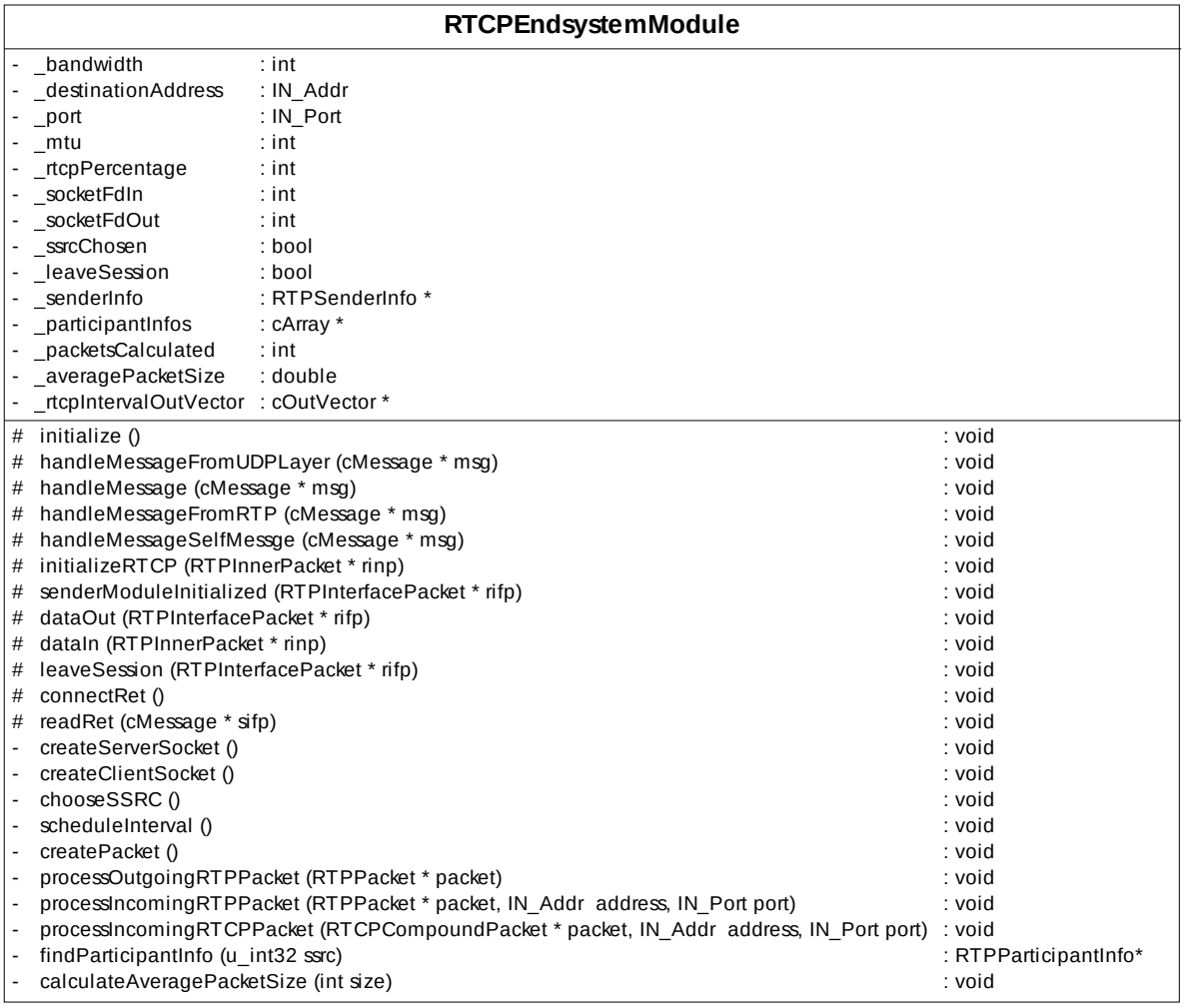


Figure 3.22 La classe RTCPEndSystemModel

2.3.3. RTPParticipantInfo

C'est est une classe abstraite. Elle permet d'enregistrer les informations des participants d'une session comme l'adresse IP, les numéros de port RTP et RTCP. Deux autres classes sont dérives de cette classe RTPSenderInfo et RTPReceiverInfo.

RTPParticipantInfo	
- _sdesChunk	: SDESChunk *
- _address	: IN_Addr
- _rtpPort	: IN_Port
- _rtcpPort	: IN_Port
- _silentIntervals	: int
+ RTPParticipantInfo (u_int32 ssrc)	
+ processRTPPacket (RTPPacket * packet, simtime_t arrivalTime)	: void
+ processSenderReport (SenderReport * report, simtime_t arrivalTime)	: void
+ processReceptionReport (ReceptionReport report, simtime_t arrivalTime)	: void
+ processSDESChunk (SDESChunk * sdesChunk, simtime_t arrivalTime)	: void
+ sdesChunk ()	: SDESChunk *
+ addSDESItem (SDESItem * sdesItem)	: void
+ receptionReport (simtime_t now)	: ReceptionReport *
+ senderReport (simtime_t now)	: SenderReport *
+ nextInterval (simtime_t now)	: void
+ toBeDeleted (simtime_t now)	: bool
+ isSender ()	: bool
+ ssrc ()	: u_int32
+ setSSRC (u_int32 ssrc)	: void
+ address ()	: IN_Addr
+ setAddress (IN_Addr address)	: void
+ rctpPort ()	: IN_Port
+ rtpPort ()	: IN_Port
+ setRTPPort (IN_Port rtpPort)	: void
+ setRTCPPort (IN_Port rtpPort)	: void
+ ssrcToName (iu_int32 nt ssrc)	: char *
+ writeContents() ()	: void

Figure 3.23 La classe RTCPEndSystemModel

2.3.4. RTPProfile

Cette classe est créée par le module RTPEndSystemModule. Elle permet de créer RTPPayloadSender or RTPPayloadReceiver. Elle reçoit les paquets RTP générés par RTPPayloadSender et les envoie au Module RTPEndSystemModule. Elle permet aussi d'envoyer les paquets RTP reçus de RTPEndSystemModule vers RTPPayloasReceiver.

RTPProfile	
- _profileName	: const char *
- _maxReceivers	: int
- _ssrcGates	: cArray
- _rtcpPercentage	: int
- _preferredPort	: IN_Port
- _mtu	: int
- _autoOutputFileNames	: bool
+ initialize() ()	: void
+ handleMessage (cMessage * msg)	: void
+ handleMessageFromRTP (cMessage * msg)	: void
+ handleMessageFromPayloadSender (cMessage * msg)	: void
+ handleMessageFromPayloadReceiver (cMessage * msg)	: void
+ initializeProfile (RTPInnerPacket * rinp)	: void
+ createSenderModule (RTPInnerPacket * rinp)	: void
+ deleteSenderModule (RTPInnerPacket * rinp)	: void
+ senderModuleControl (RTPInnerPacket * rinp)	: void
+ dataIn (RTPInnerPacket * rinp)	: void
+ senderModuleInitialized (RTPInnerPacket * rinp)	: void
+ senderModuleStatus (RTPInnerPacket * rinp)	: void
+ dataOut (RTPInnerPacket * rinp)	: void
+ processIncomingPacket (RTPInnerPacket * rinp)	: void
+ processOutgoingPacket (RTPInnerPacket * rinp)	: void
+ findSSRCGate ()	: RTPSSRCGate *
+ newSSRCGate ()	: RTPSSRCGate *

Figure 3.24 La classe RTCPEndSystemModel

2.3.5. RTPPayloadSender

RTPPayloadSender permet de construire les paquets RTP à partir d'un fichier vidéo lors d'une session. Elle parcourt le fichier et construit les paquets RTP. En tenant compte de la valeur MTU, elle fragmente les trames vidéo en plusieurs trames RTP.

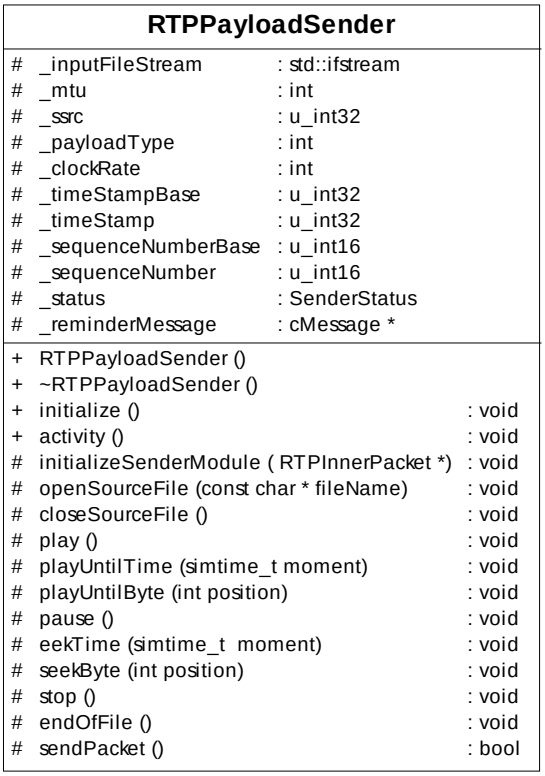


Figure 3.25 La classe RTCPEndSystemModel

2.3.6. RTPPayloadReceiver

La classe RTPPayloadReceiver reçoit les paquets à partir du module RTPProfile. Elle construit le fichier vidéo à partir des trames reçues.

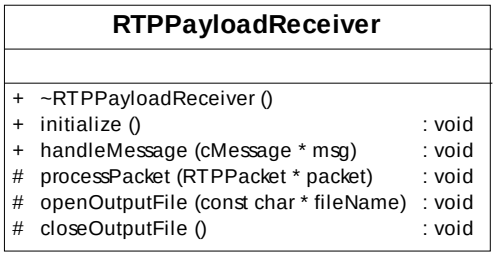


Figure 3.26 La classe RTCPEndSystemModel

2.4. RTPApplication

Cette classe permet d'envoyer des messages de contrôle à la couche RTP (ouvrir et quitter une session, commencer et arrêter la transmission vidéo).

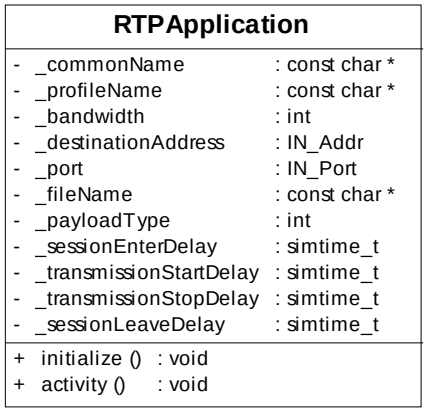


Figure 3.27 La classe RTPApplication

Le module RTPApplication contient des paramètres comme commonName qui représente le nom du participant d'une session, destinationAddress qui peut être une adresse point-a-point ou une adresse d'un groupe multipoint. Les paramètres sessionEnterDelay et sessionLeaveDealy permettent d'indiquer les instants d'entré et de départ d'une session une vidéo. TransmissionStartDelay et transmissionStopDelay permettent de paramétrer les instants de démarrage et d'arrêt d'une transmission.

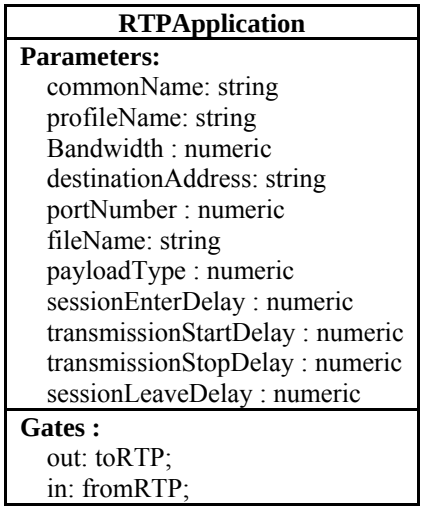


Figure 3.28 Le module RTPApplication

Nous avons détaillé notre conception des classes dans cette deuxième partie, nous décrivons dans la deuxième partie l'interaction entre elles avec les diagrammes de séquence.

3. Diagrammes de séquences

Dans cette partie nous présentons quelques exemples de diagramme de séquence qui expliquent les interactions entre les classes que nous avons créées.

3.1. Calcul de la puissance reçue

La figure 3.30 représente un des scénarios possibles pour le calcul de la puissance reçue. Dans ce diagramme l'objet radio, instance de la classe Ieee80211aRadio, fait appel à la méthode calculateReceivedPower() de l'objet receptionModel. Ce dernier est une instance de

la classe ShadowingMode. Ce diagramme reste valable pour les autres modèle Free-Space et Two-Ray.

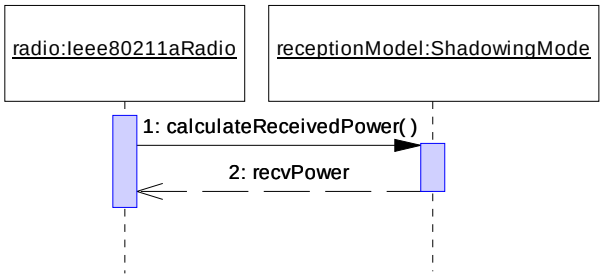


Figure 3.29 Diagramme de séquence du calcul de la puissance reçue

3.2. Calcul de la durée de transmission

Avant de transmettre un paquet, l'objet radio calcule la durée de transmission. Il fait appel à la méthode calculateDuration() de l'objet radioModel comme le montre la figure 3.31.

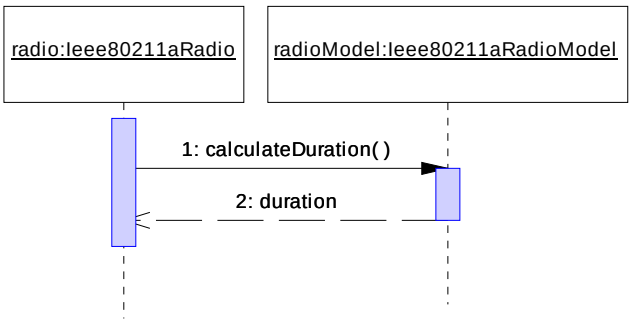


Figure 3.30 Diagramme de séquence du calcul de la durée de transmission

3.3. Calcul de PER

A chaque réception d'un nouveau message, l'objet radio fait appel à la méthode isReceivedCorrectly() de l'objet radioModel. Ce dernier fait appel à la méthode packetOK() qui calcule le CSR de chaque morceau du message reçue. En général, l'objet radio fait appel à la méthode getChrunkSuccessRate() pour chaque morceau. Ensuite, il calcule le PSR qui sera le produit des deux valeurs déjà calculées et PER qui sera 1-PSR. Puis, Il génère un nombre aléatoire entre 0 et 1 et le compare avec la valeur PER. Enfin, il retourne true si le nombre aléatoire est inférieur à PER et false sinon.

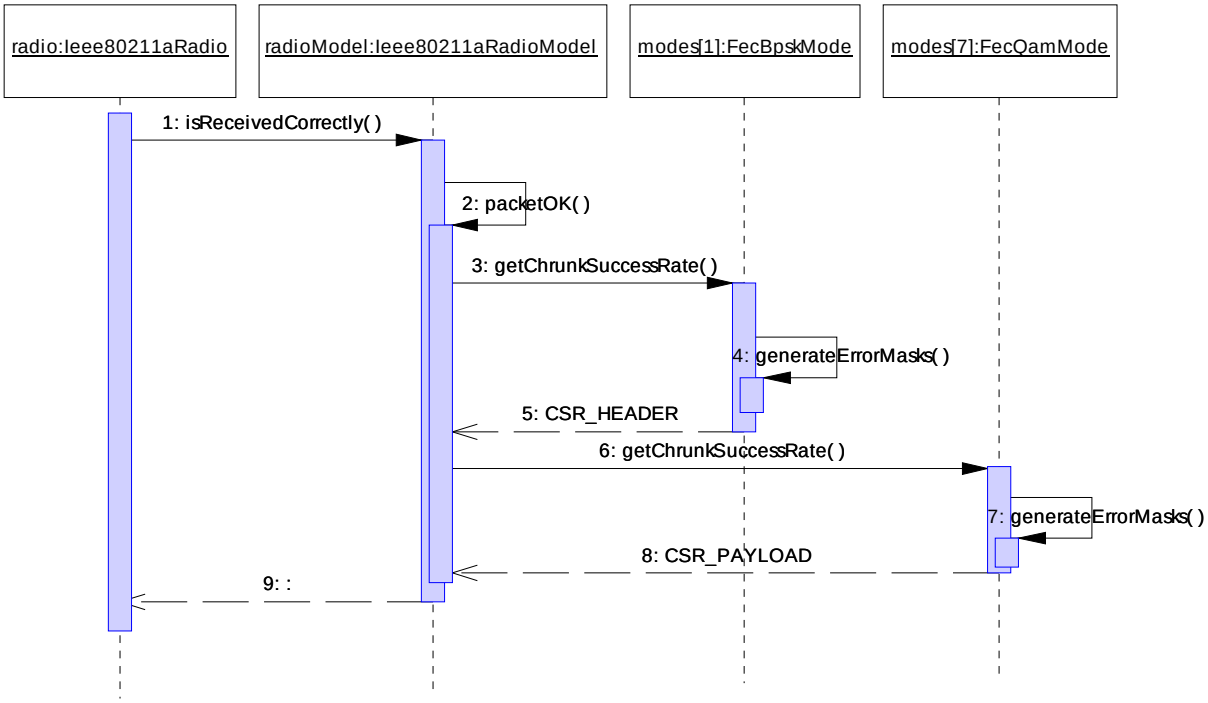


Figure 3.31 Diagramme de séquence du calcul du PER

3.4. Transmission vidéo

A l'instant sessionEnterDelay, l'objet rtpApplication, instance de la classe RTPApplication, envoie un message à l'objet rtpModule, instance de la classe RTPEndSystemModule, pour ouvrir une nouvelle session. rtpModule crée l'objet profile instance de la classe RTPProfile, ouvre une socket, et initialise l'objet rtcpModule. Une fois l'objet rtcpModule est initialisé, il renvoie un message à rtpModule. rtpModule renvoie ensuite un message au module rtpApplication lui informant que tous les modules sont initialisés.

Ensuite, si l'attribut fileName est différent de la chaîne vide, c'est à dire que rtpApplication est l'application source, il envoie un message à rtpModule afin de créer un rtpPayloadSender. A la réception de ce message, rtpModule crée un nouveau objet rtpPayloadSender et renvoie un message à rtpApplication pour lui informer que l'objet rtpPayloadSender est déjà crée et qu'il peut commencé la transmission.

A l'instant transmissionStartDelay, rtpApplication envoie un message ce contrôle "PLAY" à rtpModule qui à son tour l'envoie à rtpPayloadSender. Ce dernier ouvre le fichier dont le nom est fileName,t commence à construire les paquets RTP et les envoie au module rtpModule. A chaque réception d'un paquet RTP, rtpModule envoie une copie au module rtcpModule et l'envoie à la couche inférieure UDP.

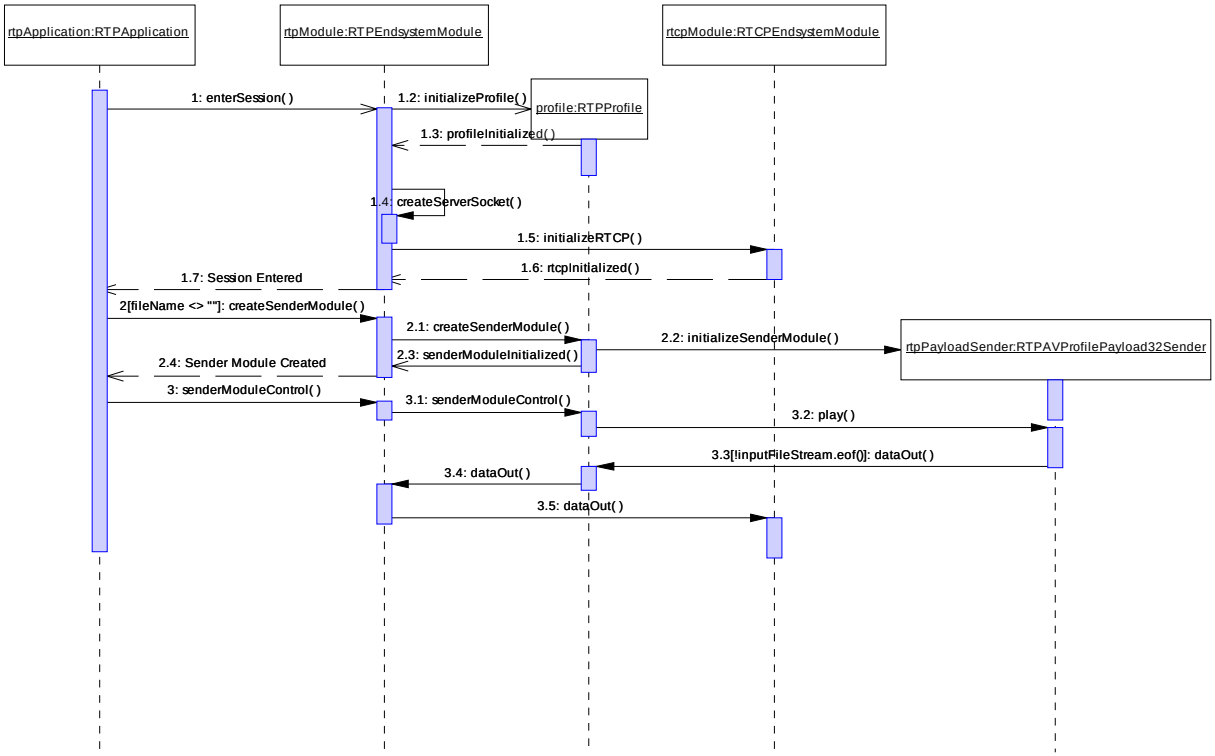


Figure 3.32 Transmission d'un flux vidéo

Conclusion

Au cours de ce chapitre, nous avons abordé la partie conceptuelle en définissant les différentes couches protocolaires ainsi que les différentes classes associées. En plus, nous avons détaillé les fonctionnalités de chaque classe.

Dans le prochain chapitre, nous décrivons la réalisation et les différentes configurations pour notre solution.

Chapitre 4 Réalisation

Introduction

Au cours de ce chapitre, nous allons décrire l'environnement matériel et logiciel de notre application. Dans une deuxième partie, nous présenterons les résultats obtenus de nos simulations. Nous finissons par donner l'état d'avancement de la réalisation ainsi que l'évolution chronologique des étapes du projet.

1. Environnement de travail

Dans ce paragraphe, nous présentons les outils matériels et logiciels nécessaires à la mise en place de l'application.

1.1. Environnement matériel

Mon projet a été réalisé en utilisant l'ordinateur « TAC » bureau 135 du département Lagrange. TAC est un ordinateur « Dell Precision 360 » dont la configuration est décrite par le tableau 4.1 :

Processeur	Intel® Pentium ® IV CPU 3 GHz
Mémoire	1Go DDR
Disque dur	160 Go

Tableau 4.1 Configuration de l'ordinateur de développement

1.2. Environnement logiciel

Notre projet a été réalisé dans l'environnement logiciel suivant :

- Système d'exploitation : Le système d'exploitation installé sur TAC est Linux KDE 3.3.2.
- Le simulateur OMNET++ : est un simulateur à évènements discrets orienté objet, basé sur C++. Il a été conçu pour simuler les systèmes réseaux de communications, les systèmes multi processeurs, et d'autres systèmes distribués. OMNET++ est un projet open source dont le développement a commencé en 1992 par Andras Vargas à l'université de Budapest. Nous avons utilisé dans nos travaux la version 3.2 de ce simulateur.
- La librairie MF : est une extension du simulateur OMNET++. Elle a été développée par une équipe de chercheurs à l'université de Berlin. Ca dernière version a été proposée par Marc Loebbers en Octobre 2003. Il est un bon support pour la simulation des réseaux sans infrastructure et mobile.

- La librairie INET : est une librairie open-source pour la simulation des réseaux informatiques dans l'environnement OMNET++. Elle contient des modules pour plusieurs protocoles comme TCP, IP, UDP, Ethernet, PPP, IEEE 802.11, MPLS, RSVP et beaucoup d'autres protocoles.
- La libraire IT++ : Nous avons utilisé cette librairie pour l'implémentation des modèles propagation. Elle est une librairie C++ pour le calcul mathématique, le traitement du signal, le traitement de la parole et d'autre classe de communication. Elle a été développée par plusieurs chercheurs pendant leurs travaux de thèse. Une description de l'utilisation de cette librairie pour ajouter un modèle de fading est détaillé dans le travail de mastère de M. Khosroshahy [Khos, 06].
- La librairie tcl/tk
- Outils de développement : Nous avons utilisé l'éditeur Kate 3.3.2 comme outil de développement open source.
- Outil de conception : Nos diagrammes de cas d'utilisation, de séquences et de classes ont été réalisés à l'aide de Power AMC version 11.1.0.1547
- Autres outils : Nous avons utilisé Plove qui est un outil OMNET++ pour tracer les graphes.

1.3. Installation des différentes librairies

1.3.1. Installation de Omnet ++

Avant de commencer l'installation, il faut tout d'accord extraire omnetpp-3.3-src.tgz à l'aide de la commande suivante

```
tar zxvf omnetpp-3.3-src.tgz
```

Ensuite, Il faut modifié les variables de l'environnement PATH et LD_LIBRARY_PATH comme indiqué

```
export PATH=$PATH:~/omnetpp-3.3/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/omnetpp-3.3/lib
```

Puis, Il faut vérifier le fichier configure.user surtout quand l'utilisateur n'est pas *root*.

```
vi configure.user
```

A la fin de la configuration, commence l'installation avec l'exécution des deux commandes suivantes

```
./configure
make
```

Pour verifier l'installation, il faut tester l'exemple dyna.

```
cd ~/omnetpp-3.3/samples/dyna
./dyna
```

Une interface graphique doit apparaître. Pour modifier la configuration de OMNET++, il faut éditer le fichier configure .user et ensuite exécuter les trois commande suivante :

```
./configure
make clean
make
```

Après chaque modification Il faut recompiler en exécutant les deux commandes

```
make clean
make
```

1.3.2. Installation de INET

Installation de INET demande une installation déjà de OMNET++. Avant d'installer INET, il faut tester que les exemples de OMNET++ fonctionnent correctement. La décompression de INET-20061020-src.tgz se fait à l'aide de la commande suivante

```
make tar zxvf  INET-20061020-src.tgz
```

Il faut éditer ensuite le fichier inetconfig en précisant le chemin ROOT pour INET. Pour créer les makefile et omnetppconfig, il faut executer la commande suivante

```
./makemake
```

Ensuite, pour compiler la librairie INET il faut executer la commande

```
make
```

Il faut aussi modifier la variable d'environnement LD_LIBRARY_PATH comme indiqué

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/INET-20061020/bin
```

Enfin, pour vérifier l'installation, il exécuter un exemple

```
~/INET-20061020/Examples/rundemo
```

1.3.3. Installation de IT++

Pour utiliser la librairie IT++ avec le simulateur OMNET++, il faut éditer le fichier omnetppconfig comme indiqué dans l'annexe C. Ensuite, il suffit d'exécuter

```
make
```

1.3.4. Integration de la couche RTP

Pour utiliser la couche RTP dans le simulateur OMNET++, il faut éditer le fichier makemakefiles comme l'exemple de l'annexe D. Puis, il faut exécuter les deux commandes

```
makemake
make
```

2. Simulations et Résultats

Le but de cette section est de présenter les résultats des simulations que nous avons effectués.

2.1 Comparaison entre les différents modèles de propagation physique

Nous allons nous intéressés à la comparaison des modèles physiques de propagation FreeSpace, Two Ray et Shadowing. Nos comparaisons sont faîtes en fonction du PER. Notre réseau est composé comme le montre la figure 4.1 essentiellement d'un serveur « source vidéo », d'un point d'accès et une station. Le point d'accès est connecté à un réseau filaire dans lequel un serveur transmet un flux vidéo vers la station mobile. La station mobile se déplace avec une vitesse constante de 1m/s en s'éloignant du point d'accès avec une direction linéaire.

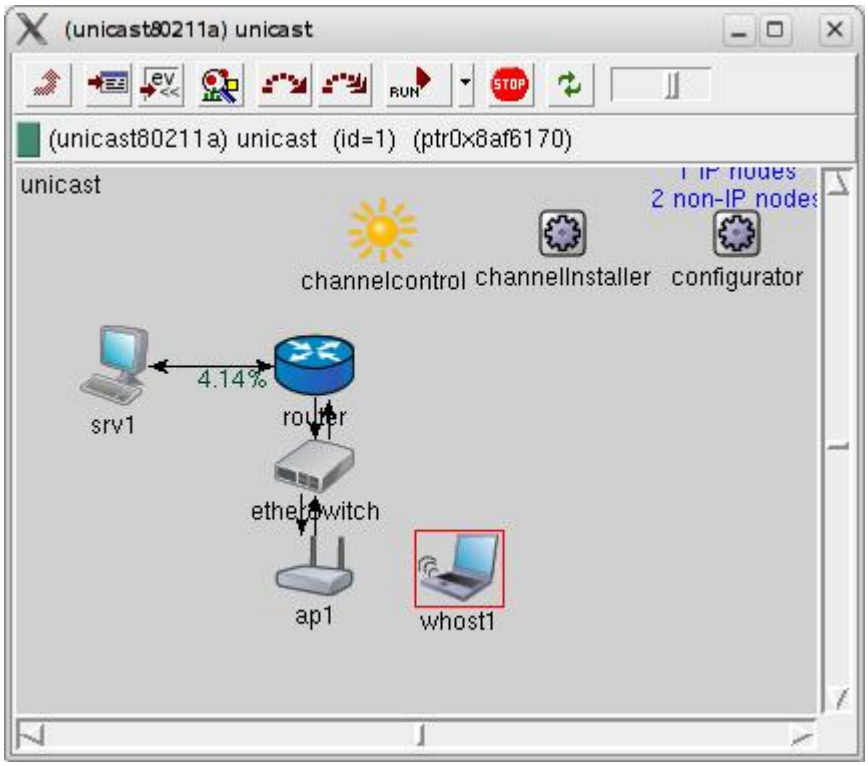


Figure 4.1

Nous avons réalisé des simulations en utilisant un réseau de type IEEE 802.11a avec les paramètres décrits dans le tableau 4.2.

Paramètres	Valeur
bitrate	6E+6
transmitterPower	200.0 [mW]
carrierFrequency	5E+9
thermalNoise	-72 db
sensitivity	-65 dB

Tableau 4.2 Paramètres de configuration

Les tableaux 4.3 et 4.4 montre les paramètres de configuration des deux modèles Two Ray et Shadowing que nous avons utilisés dans nos simulations.

Paramètres	Valeur
ht	3
hr	1

Tableau 4.3 Les paramètres du modèle Two-Ray

Paramètres	Valeur
pathLossExponent	3.3
shadowingVariance	3.0
shadowingNumberofSamples	1000

Tableau 4.4 Les paramètres du modèles Shadowing

La figure 4.2 montre la variation de PER en fonction de la distance dans un cas ou nous ne tenons pas compte du fading. Notre première remarque est que le modèle Two-Ray donne un faible taux de perte. Notre explication est que ce modèle n'est pas le mieux adopté pour les réseaux IEEE 802.11 car il est basé sur l'hypothèse que la distance entre l'émetteur

et le récepteur doit être très grande devant (ht*hr) ². Ce qui n'est pas toujours vrai dans le réseau IEEE 802.11 dont la dimension n'est pas grande. Notre deuxième remarque est que dans tous les modèles le taux de perte augmente quand la distance entre l'émetteur et le récepteur augmente. Enfin, nous remarquons qu'il y a des pertes de paquets même à une petite distance du AP dans le modèle Shadowing.

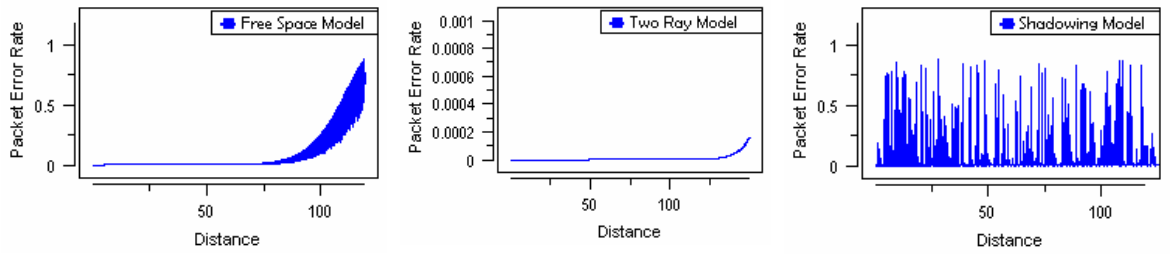


Figure 4.2 PER en fonction de la distance

Nous avons aussi simulé le même scénario avec le modèle FreeSpace en ajoutant l'effet de fading. Nous avons utilisé les même les paramètres utilisé par [Khos, 06] qui sont dans le tableau 4.5.

Paramètres	Valeur
fadingNumberOfSamples	20000
simulationBaudRate	1125000
normalizedDopplerFrequency	0.01
averagePowerProfileDb	0
fadingChannelRicianFactor	0
typeOfChannelForBER	Slow-Fading Channel
minSnrForOutageProbInSlowFading	1
perCalculationMethod	Non-Uniform

Tableau 4.5 Les paramètres de configuration du fading

La figure 4.3 montre que en ajoutant l'effet fading au modèle Free Space les pertes commencent même à une petite distance du point d'accès. Nous remarquons aussi que à 50 mètres du point d'accès les paquets sont presque tous perdus.

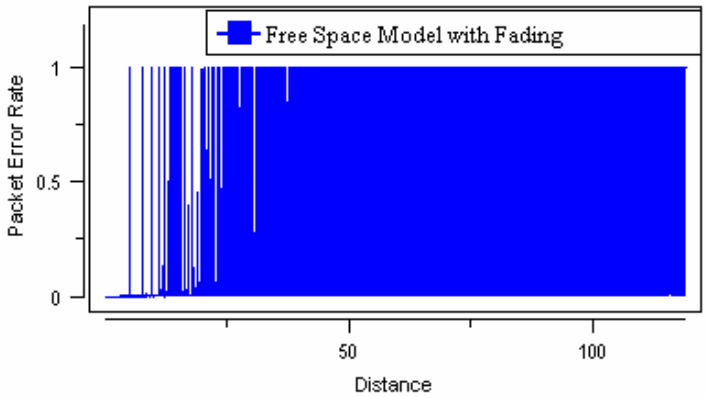
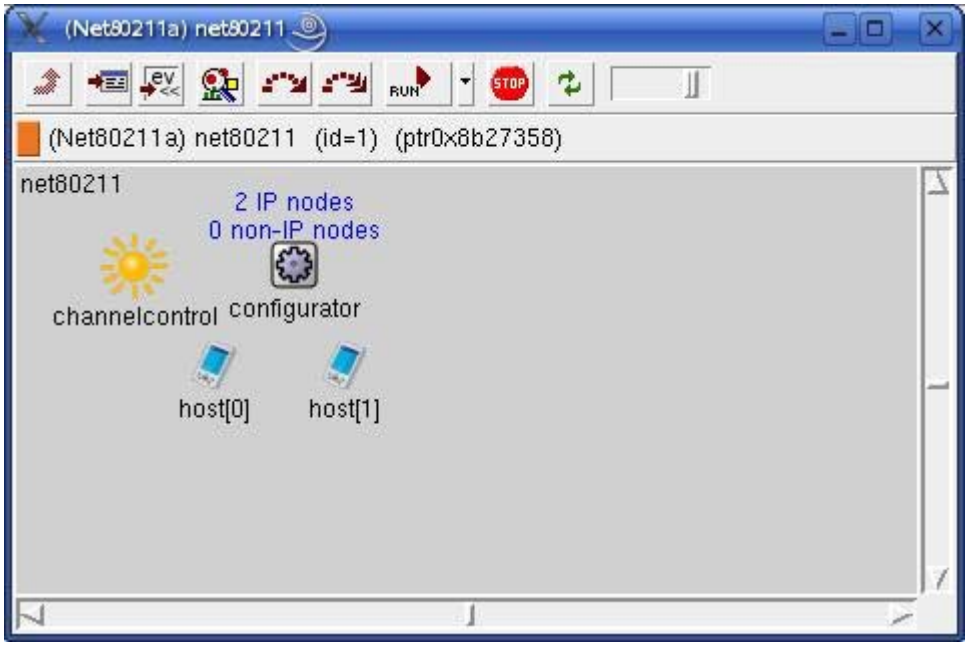


Figure 4.3 Influence du fading sur la perte des paquets

2.2 Comparaison entre les deux algorithmes d'adaptation du débit physique

Pour mettre en évidence l'apport de l'algorithme AARF par rapport à ARF, nous avons simulé un scénario avec deux stations A et B formant un réseau Ad-hoc distantes de 100 mètres. La station A envoie des paquets ICMP vers la station B. Dans ce scénario nous avons utilisé le modèle FreeSpace avec le modèle *fading*.



D'après la figure 4.4, Nous constatons que le meilleur débit PHY entre ces deux stations est 6Mbps. ARF augmente le débit après 10 transmissions successifs et par la suite il cause plus de perte de paquets.

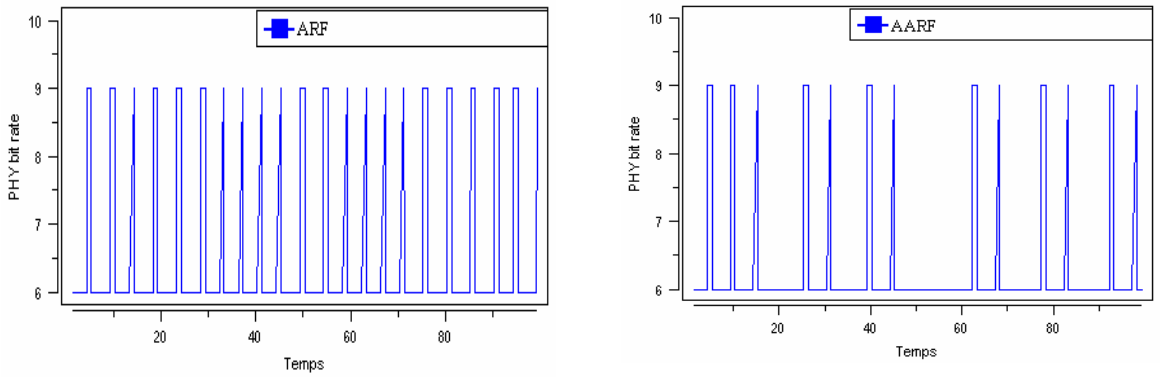


Figure 4.4 Comparaison entre les deux approches ARF et AARF

2.3 Comparaison entre le multicast standard et l'approche Leader

Afin de montrer l'apport de l'approche Leader dans la transmission multimédia dans les réseaux IEEE 802.11, nous avons simulé deux scénarios de transmission vidéo dans un réseau IEEE 802.11b. Le premier scénario avec la transmission multicast standard alors que le deuxième avec l'approche Leader. Le modèle physique utilisé dans ce scénario est le

FreeSpace. Comme le montre le figure 4.4, ce scénario consiste à deux réseau l'un de type Ethernet et l'autre de type IEEE 802.11.

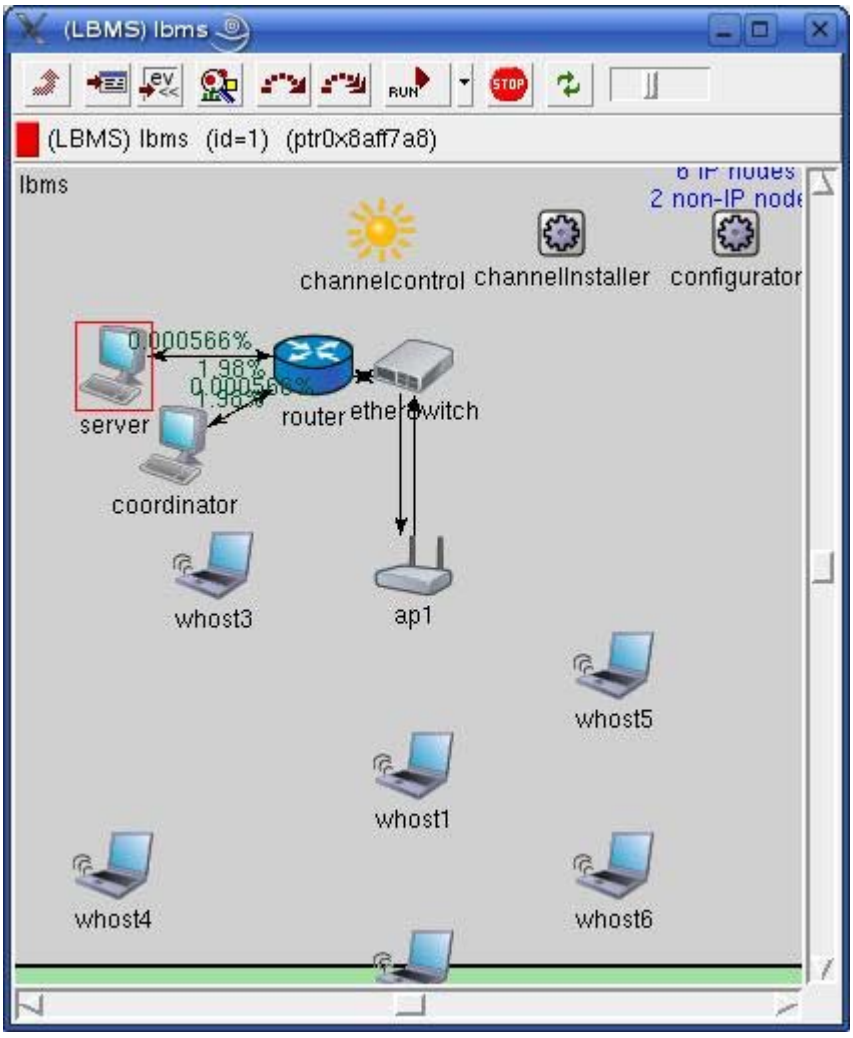


Figure 4.5 Scénario de transmission de la vidéo dans un réseau IEEE 802.11

Le tableau 4.6 montre le taux de perte des paquets RTP au niveau de chaque station, Nous constatons que le taux de perte est très élevé dans la station 4 dans notre première simulation. Cette perte est due à la distance qui sépare la station 4 au point d'accès. Nous avons cette station comme Leader dans notre deuxième simulation. Nous constatons que le taux de perte a diminué pour cette station de 16.37% à 9.09%. Le taux de pertes a baissé aussi pour les autres stations. Le taux de pertes 9.09% est due au perte des paquets après trois retransmissions successifs et aussi au perte dans la queue du point d'accès a cause de la retransmission des anciens paquets.

	Station 1	Station 2	Station 3	Station 4	Station 5	Station 6
Multicast standard	4.3	15.33	5.59	16.37	4.48	14.85
Station 4 Leader	3.81	7.8	3.71	9.09	10.73	4.73

Tableau 4.6 Le taux de perte des paquets

3. Difficultés rencontrés

Durant la réalisation, nous avons rencontré de nombreuses difficultés. La première difficulté est l'installation des librairies OMNeT++ et la librairie IT++. Le nombre important de librairies à installer et leurs dépendances rendent l'installation très ardue. La deuxième difficulté est la compilation des librairies OMNET++, IT++ et INET qui demandent de nombreuses modifications dans divers fichiers de configuration pour chaque librairie. De plus, l'environnement matériel nous a posé un certain nombre de contraintes comme le temps de compilation et le temps de simulation.

Enfin, nous avons rencontré de nombreux bugs dans la librairie INET que nous avons résolu à l'aide de Andras Vargas et les membres du mailing list de OMNET++ [].

4. Etat courant du travail

Notre nouvelle version du simulateur OMNET++ contient tous les modèles de transmission physiques implémentés. Nous avons aussi ajouté les deux modèles de la couche MAC IEEE 802.11a et b avec les deux algorithmes d'adaptation du débit physique ARF et AARF. Nous avons intégré la couche RTP dans la librairie INET et la transmission multipoint dans les réseaux IEEE 802.11.

Notre travail sera intégré avec les modules développés par les autres partenaires du projet Divine avec quelques modifications et sera proposé comme une contribution pour les prochaines versions de INET.

5. Chronogrammes

Nous avons commencé notre stage le 15 février 2007 et nous l'avons fini le 11 Juin 2007. Notre travail contient de nombreuses parties indépendantes. C'est la raison pour la quel nous avons travaillé partie par partie.

	Semaines																			
	Février				Mars				Avril				Mai				Juin			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Etat de l'art																				
Spécification																				
Installation et configuration																				
Conception																				
Réalisation et Test																				
Rédaction du rapport																				

Figure 4.6 Chronogramme du projet

Conclusion

Ce chapitre a été consacré à la présentation des résultats de notre projet. Nous avons décrit au début l'environnement de réalisation de notre projet. Nous avons présenté ensuite les simulations que nous avons réalisées dans notre stage et enfin nous avons expliqué les différentes contraintes de réalisation et l'état actuel de notre projet.

Conclusion & Perspective

Le succès grandissant des applications multimédia sur Internet rend indispensable l'amélioration de la diffusion vidéo dans les réseaux 802.11. A cet effet, la méthode de diffusion multipoint avec l'élection d'un leader apporte plusieurs améliorations permettant d'enregistrer de nouveaux progrès dans la diffusion de la vidéo sur les réseaux sans fil. En outre, elle adapte le débit physique suivant l'état du support de transmission. Enfin, elle retransmet au niveau de la couche MAC les paquets perdus.

Cependant, si le Leader a un taux de perte très important il peut nuire à tout le groupe. Ainsi, le nombre important de retransmission peut être l'origine d'autre perte dans la queue du point d'accès.

Ce travail sera suivi par une deuxième partie où nous allons ajouter un autre nœud « Coordinateur » dans le réseau local dans notre scénario de type musée. Cette nouvelle station aura pour rôle d'optimiser la transmission vidéo, c'est-à-dire de calculer le nombre optimal de couches, le débit d'émission de chaque couche et éventuellement le taux de redondance par couche en fonction des rapports de réception des stations. L'algorithme du coordinateur fera l'objet de recherches à venir dans notre travail de mastère.

Ce projet nous a permis de toucher à la réalité du travail dans un laboratoire, dans un environnement orienté recherche. Nous avons appris à considérer les notions d'un oeil critique afin de vouloir toujours améliorer, toujours essayer de changer et d'innover. Cette expérience est la première dans notre jeune formation, elle fût réussie à plusieurs points de vue. Sur le plan concret, ce projet nous a offert l'occasion de programmer en C++ sous l'environnement Linux, décortiquer dans la norme IEEE 802.11 et les protocoles temps réels RTP et RTCP, implémenter de nouveaux protocoles et simuler quelques exemples scénarios.

Bibliographie

[802.11, 99] IEEE 802.11 WG, “Reference number ISO/IEC 8802-11:1999(E) IEEE Std International Standard [for] Information Technology-Telecommunications and information systems-Local and metropolitan area networks-Specific Requirements-Part Access Control (MAC) and Physical Layer (PHY) specifications,” 1999.

[802.11a, 00] “ISO/IEC 8802-11:1999/Amd 1:2000(E); IEEE Std 802.11a-1999. Supplement to IEEE Standard for Information technology. Telecommunications and information exchange between systems. Local and metropolitan area networks. Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications High-speed Physical Layer in the 5 GHz Band,” 1999.

[Ben Slimane, 05] Abdelrahmen Ben Slimane “Multicast multimedia sur Internet” Lavoisier, 2005.

[Dujovne et al., 06] Diego Dujovne, Thierry Turletti, “Multicast in 802.11 WLANs : An experimental Study”, OFMSWIM, 2006

[Ergen, 02] Mustafa Ergen, “IEEE 802.11 Tutorial”, Department of Electrical Engineering and Computer Science, University of California Berkeley, 2002.

[Holland et al, 01] Gavin Holland, Nitin Vaidya, Paramvir Bahl “A Rate Adaptive MAC Protocol for MultiHop Wireless Networks”.

[IT++, 07] http://itpp.sourceforge.net/ version 3.10.5, 21 Février 2006.

[INET, 07] http://www.omnetpp.org/doc/INET/neddoc/, le 25 Février 2006.

[Khosroshahy, 06] M. Khosroshahy, “Study and implementation of IEEE 802.11 Physical Layer Model in YANS”, Master Thesis, Dec 2006.

[Khalili et al., 06] Ramin Khalili, Kavé Salamatian, “An Analytical Model for Residual Errors in Convolutional Codes”, LIP6- CNRS, Université Pierre et Marie Curie, France, Technical Report, Paper to be submitted, 2006

[Kamerman et al., 97] A. Kamerman and L. Monteban. “WaveLAN-II: A High-performance wireless LAN for the unlicensed band”. Bell Lab Technical Journal, pages 118{133, Summer 1997.

[Lacage et al., 04] M. Lacage, M.H. Manshaei, T. Turletti, “A Practical Approach to Rate Adaptation”, INRIA Research Report, No 5208, May 2004.

[Löbbers et al., 07] Marc Löbbers, Daniel Willkomm, “A Mobility Framework for OMNeT++ User Manual”, January 2007.

[MF, 07] http://mobility-fw.sourceforge.net/ le 6 Mars 2007.

[Manshaei et al., 05] Mohammad Hossein Manshaei, Gion Reto Cantieni, Chadi Barakat, and Thierry Turletti, “Performance Analysis of the IEEE 802.11 MAC and Physical Layer Protocol”, 2004.

[Manshaei, 05] Mohammad Hossein Manshaei, “Cross layer interactions for adaptive communications in IEEE 802.11 wireless lans”, thèse, Ecole Doctorale STIC, Université de Nice - Sophia Antipolis Décembre 2005.

[Mélin, 06] Jeans-Louis Mélin, “Quality of service sur IP”, edition eyrolles, 2001.

[O’Hara, 05] Bob O’Hara, Al Petrick “IEEE 802.11 handbook” Published Mars 2005.

[Omnet, 07] http://www.omnetpp.org/, le 2 Mars 2007.

[Oppitz, 02] Matthias Oppitz, “Entwurf und implementierung einer Simulation des Real-time Transport Protocol unter OMNET ++”, Mars 2002.

[Perkins, 03] Colin Perkins “RTP audio and video for the Internet”, Addition-Westley, first printing, Juin 2003.

[Rappaport, 02] Theodore S. Rappaport “Wireless Communications, Principles and Practice” 2nd ed., Prentice Hall, 2002.

[Seok et al., 07] Y. Seok, T. Turletti ”Mécanismes de Transmission Multipoint pour Réseaux Locaux Sans Fil IEEE 802.11”, INRIA Research Report No RR-5993, 2006.

[Seok et al., 07] Y. Seok, T. Turletti, D. Dujovne, E. Qi, P. Cuenca, “Leader based multicast proposal”, IEEE 802.11-07/0144r3

[Seok , 07] IEEE P802.11v/D0.08, “Draft amendment v: Wireless Network Management”, May 2007.

[Schulzrinne, et al., 03] H. Schulzrinne, et al., “RTP: A Transport Protocol for Real-Time Applications”, RFC-3550, July 2003.

[Van Steyvoort, 06] Van Steyvoort Thomas, “Détection distribuée des paramètres de propagation indoor dans les réseaux sans-fils”, 2006.

Glossaire

AARF	Adaptive Auto Rate Fallback
ACK	ACKnowledgment
AP	Access Point
ARF	Auto Rate Fallback
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
BS	Base Station
BSS	Basic Service Set
CA	Collision avoidance
CBR	Constant Bit Rate
CCK	Complementary Code Keying
CD	Collision Detection
CDMA	Code Division Multiple Access
CFP	Contention Free Period
CNAME	Canonical NAME
CRC	Cyclic Redundancy Check
CS	Carrier Sense
CSMA	Carrier Sense Multiple Access
CSR	Chunk Success Rate
CTS	Clear To Send
CW	Contention Window
DBPSK	Differential Binary Phase Shift Keying
DCF	Distributed Coordination Function
DQPSK	Differential Quadrature Phase Shift Keying
DS	Direct Sequence
DSSS	Direct Sequence Spread Spectrum
ESS	Extended Service System
FEC	Forward Error Correction
FHSS	Frequency Hopping Spread Spectrum
IBSS	Independent BSS
ID	Identification
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Multicast Protocol
INRIA	Institut National de Recherche en Informatique et Automatique
IP	Internet Protocol
IR	Infra-Red
ISO	International Organization for Standardization
LAN	Local Area Network
LBMS	Leader Based Mechanism Services
LEP	Leader Election Protocol
LLC	Logical Link Control
LOS	Line Of Sight
MAC	Medium Access Control
MF	Mobility Framework
MH	Mobile Host

MONET	MObile ad hoc NETwork
NAV	Network Allocation Vector
OFDM	Orthogonal Frequency Division Multiplexing
OSI	Open System Interconnection
PCF	Point Coordination Function
PDA	Personal Digital Assistant
PDU	Packet Data Unit
PER	Packet Error Rate
PHY	PHYsical layer
PLCP	Physical Layer Convergence Procedure
PMD	Physical Medium Dependent
PSR	Packet Success Rate
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RBAR	Receiver Based Auto Rate
RR	Receiver Report
RTP	Real-time Transport Protocol
RTCP	Real-time Transport Control Protocol
RTS	Request To Send
SIFS	Short IFS
SNR	Signal to Noise Ratio
SNIR	Signal to Noise and Interference Ratio
SR	Sender Report
SYNC	Synchronization
TCP	Transport Control Protocol
UDP	User Datagram Protocol
WLAN	Wireless Local Area Network

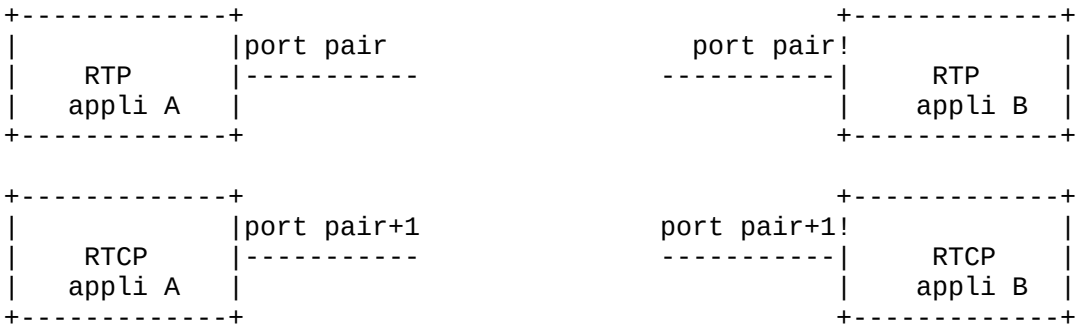
Annexe A : Les formats des paquets RTP et RTCP

RTP (Realtime Transport Protocol) et son compagnon RTCP (Realtime Transport Control Protocol) permettent respectivement de transporter et de contrôler des flots de données qui ont des propriétés temps-réel.

RTP et RTCP sont des protocoles qui se situent au niveau de l'application et utilisent les protocoles sous-jacents de transport TCP ou UDP. Mais l'utilisation de RTP/RTCP se fait généralement au-dessus de UDP.

RTP et RTCP peuvent utiliser aussi bien le mode point à point que le mode multipoint.

Chacun d'eux utilise un port séparé d'une paire de ports. RTP utilise le port pair et RTCP le port impair immédiatement supérieur.



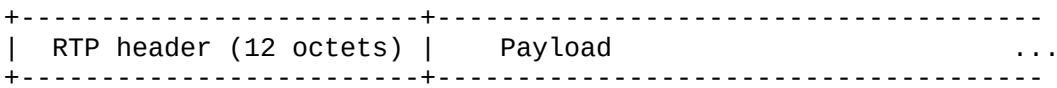
1. RTP

1.1. Rôle

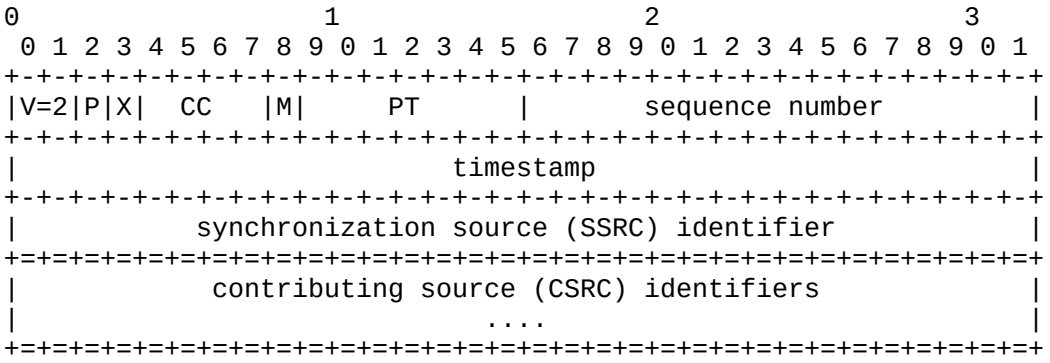
Le but du protocole RTP est de transporter des données qui ont des propriétés temps-réel.
Format du paquet

L'entête d'un paquet RTP est obligatoirement constituée de 12 octets, éventuellement suivie d'une liste d'identificateurs de sources contributeurs CSRCs dans le cas d'un mixer. Cette entête précède le "payload" qui représente les données utiles.

Les types de payload sont standardisés.



1.2. Format de l'entête RTP



- version V : 2 bits, V=2
- padding P : 1 bit, si P=1 le paquet contient des octets additionnels de bourrage (padding) pour finir le dernier paquet.
- extension X : 1 bit, si X=1 l'entête est suivie d'un paquet d'extension
- CSRC count CC : 4 bits, contient le nombre de CSRC qui suivent l'entête
- marker M : 1 bit, son interprétation est définie par un profil d'application (profile)
- payload type PT : 7 bits, ce champ identifie le type du payload (audio, vidéo, image, texte, html, etc.)
- sequence number : 16 bits, sa valeur initiale est aléatoire et il s'incrémente de 1 à chaque paquet envoyé, il peut servir à détecter des paquets perdus
- timestamp : 32 bits, reflète l'instant d'échantillonnage du premier octet du paquet
- SSRC: 32 bits, identifie de manière unique la source et sa valeur est choisie de manière aléatoire par l'application
- CSRC : 32 bits, identifie les sources contribuant.

2. RTCP

2.1. Rôle

Le protocole RTCP est basé sur des transmissions périodiques de paquets de contrôle par tous les participants dans la session.

2.2. Format des paquets

Il existe 5 types de paquets RTCP pour transporter des informations de contrôle :

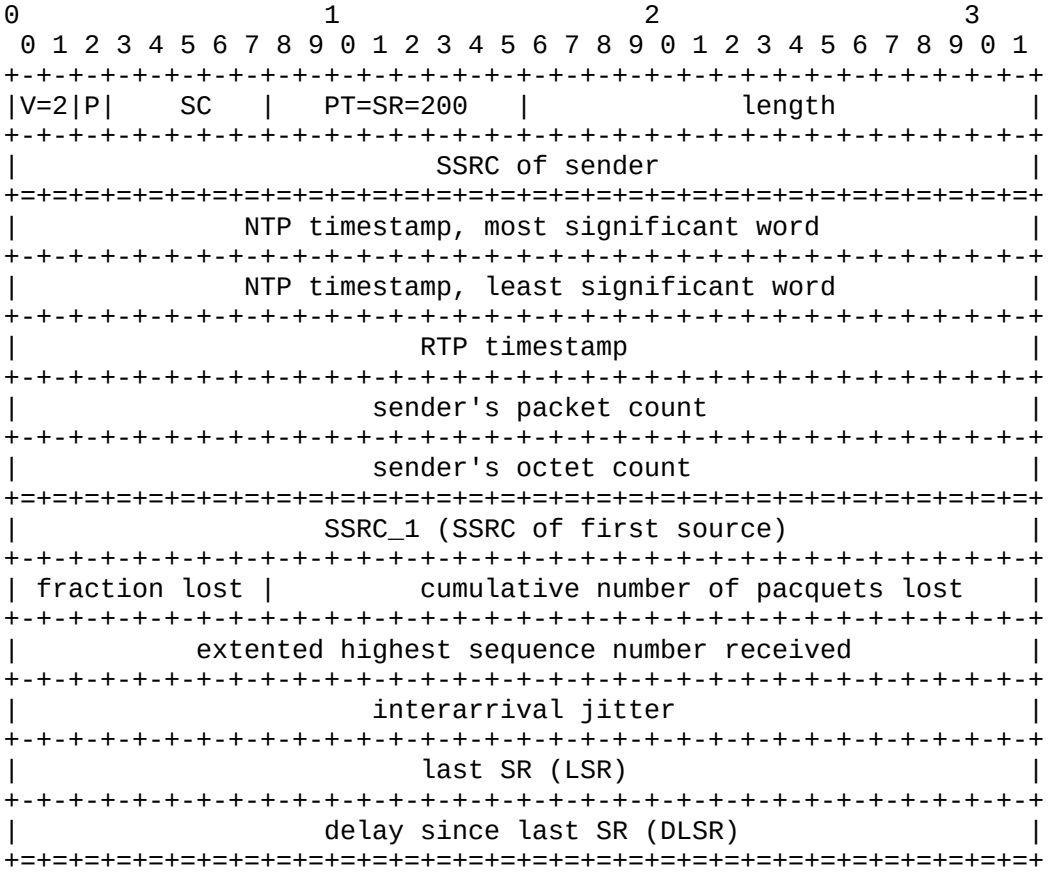
- SR : Sender Report, transmission de statistiques des participants actifs en émission
- RR : Receiver Report, transmission de statistiques des participants passifs
- SDES : Source Description items (CNAME, NAME, EMAIL, PHONE,...)
- BYE : Fin de participation

- APP : Fonctions spécifiques à l'application

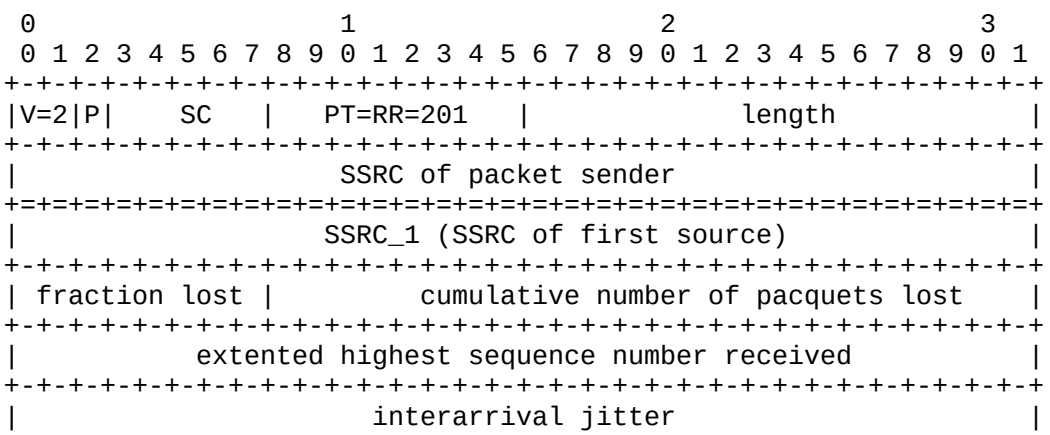
Chaque paquet RTCP commence par une partie fixe, suivie par des éléments structurés qui peuvent être de longueur variable selon le type de paquet, mais toujours terminé sur une frontière de 32 bits.

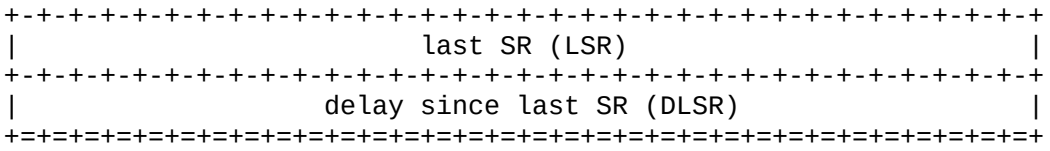
SR (Sender Report)

La partie fixe minimum d'un paquet SR est de 4 octets. Un paquet SR peut contenir des informations de la partie émetteur (sender info) sur 6 mots de 32 bits (48 octets) et ou des informations de la partie récepteur (report block) sur 6 mots de 32 bits (48 octets).

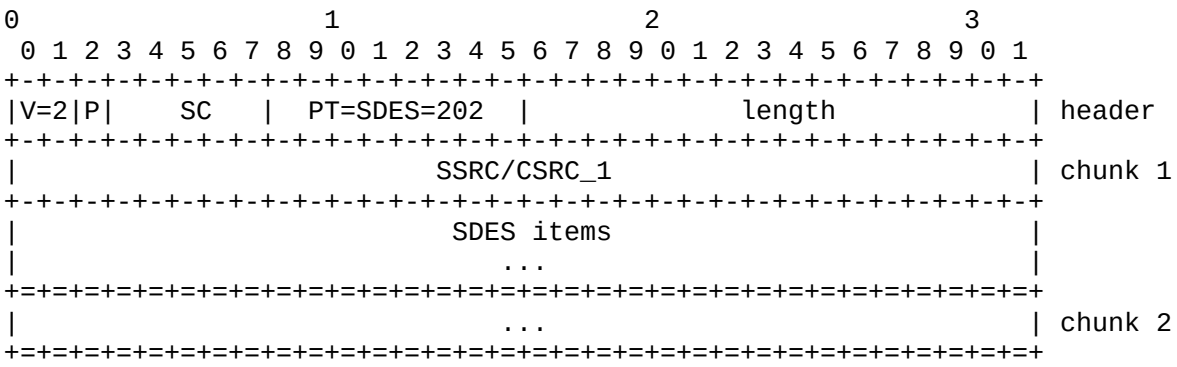


RR (Receiver Report)



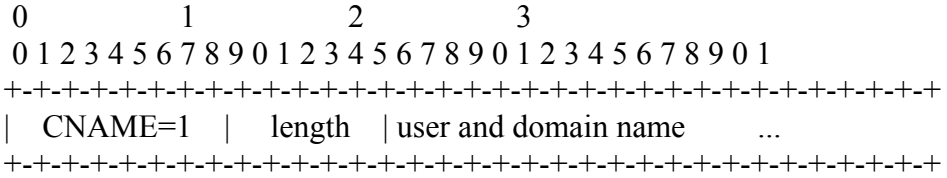


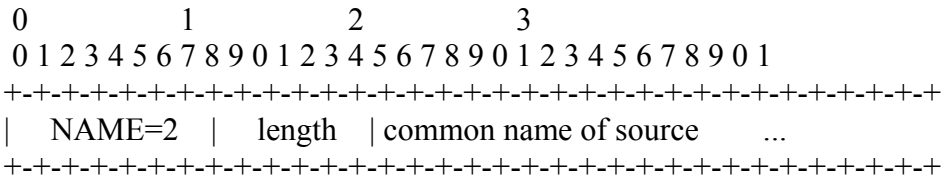
SDES (Source Description)

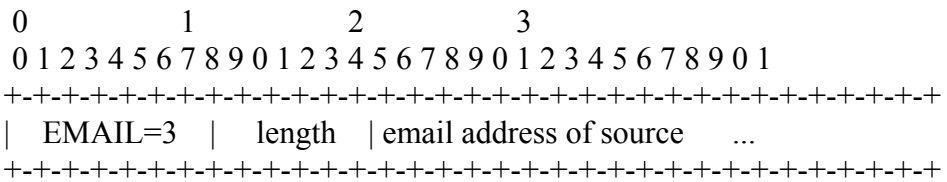


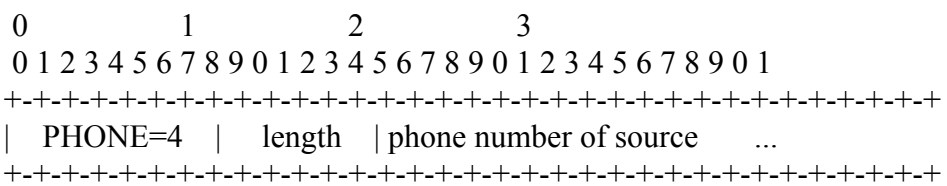
Format des SDES items

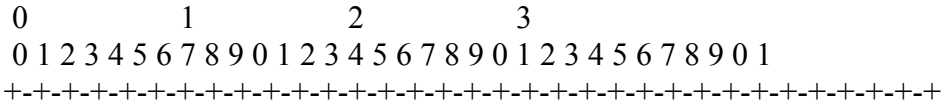
Le premier octet définit le type de l'item. Le second octet donne la longueur en octets de la chaîne de caractères de taille variable qui suit, mais arrondie sur une frontière de mot.

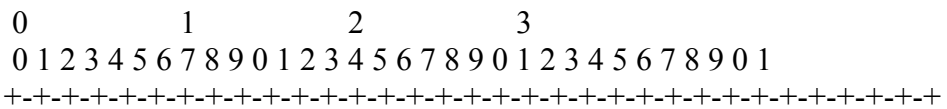


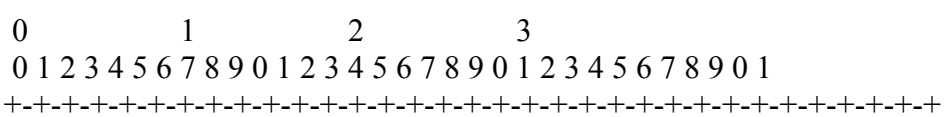






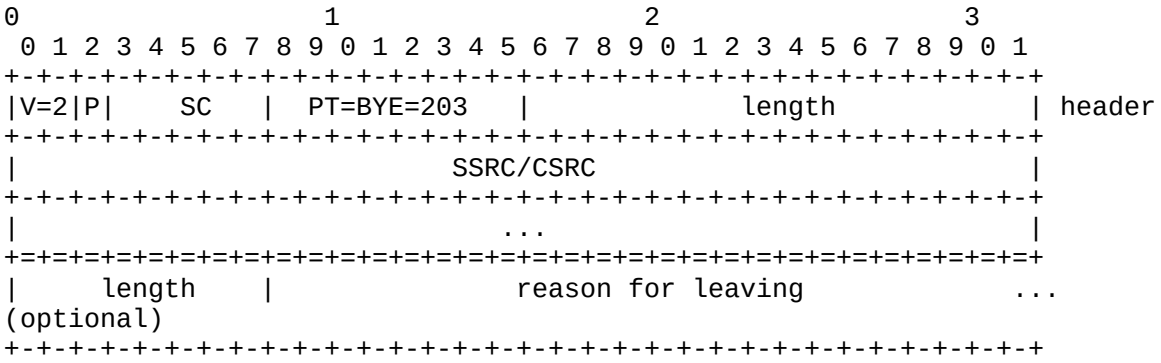






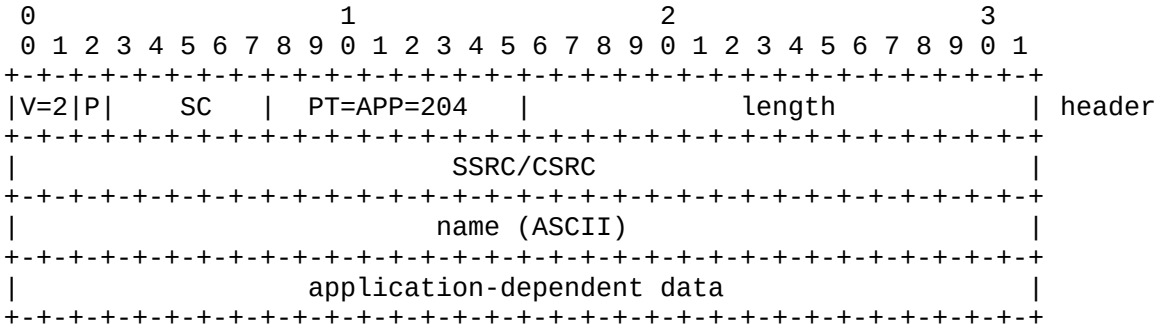
BYE

Ce paquet permet d'indiquer aux autres participants que le membre quitte la session.



APP

Ce paquet permet d'ajouter des fonctions spécifiques à une application.



Annexe B : Le fichier masques d'erreur

L'utilisateur aura à la fin de la simulation un fichier contenant les masques d'erreur pour chaque paquet simulé. Le suivant est un exemple de masque d'erreur généré pour un paquet.

```
[Masks after decoder. 1st Part: PHY PLCP header -2nd Part: MAC Header &
Payload
| 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Annexe C : Le fichier omnetconfig

```
#
# OMNeT++ simulation model configuration file
# (included in makefiles generated with -c option)
#
# Generated by opp_makemake --genconfig omnetppconfig
#

#
# Local configuration
#
NEDC=/home/ahmed/omnetpp-3.3/bin/nedtool
MSGC=opp_msgc
CXX=g++
CC=gcc
AR=ar cr
SHLIB_LD=g++ -shared
MAKEDEPEND=opp_makedep -Y --objdirtree

CFLAGS=-O2 -DNDEBUG=1   -DWITH_PARSIM -DWITH_NETBUILDER
NEDCFLAGS=-Wno-unused
LDFLAGS= -Wl,--export-dynamic
EXE_SUFFIX=

WITH_PARSIM=yes
WITH_NETBUILDER=yes

OMNETPP_INCL_DIR=/home/ahmed/omnetpp-3.3/include
OMNETPP_LIB_DIR=/home/ahmed/omnetpp-3.3/lib

TK_LIBS=-L/usr/X11R6/lib -lX11 -ltk8.4 -ltcl8.4
MPI_LIBS=
XML_LIBS=-lxml2
SYS_LIBS=-ldl -lstdc++
SYS_LIBS_PURE=-ldl -lsocket -lnsl -lm $(shell $(CXX) -print-file-
name=libstdc++.a)

#
# Definitions for models
#

# User interface libs
CMDENV_LIBS=-lenvir -lcmdenv
TKENV_LIBS=-lenvir -ltkenv $(TK_LIBS)

# Simulation kernel
KERNEL_LIBS=-lsim_std
```

```
ifeq ($(WITH_NETBUILDER),yes)
KERNEL_LIBS += -lnedxml $(XML_LIBS)
endif

ifeq ($(WITH_PARSIM),yes)
KERNEL_LIBS += $(MPI_LIBS)
endif

# Simulation kernel and user interface libraries
OMNETPP_LIBS=-L$(OMNETPP_LIB_DIR) $(USERIF_LIBS) $(KERNEL_LIBS) $(SYS_LIBS)

COPTS=$(CFLAGS) $(INCLUDE_PATH) -I$(OMNETPP_INCL_DIR)
NEDCOPTS=$(COPTS) $(NEDCFLAGS)
```

Annexe D : Le fichier makemakefile

```
#
# Makefile to create all other makefiles for the project.
#
# CAREFUL: This file has to remain portable across Unix make and Windows
nmake!
#

# The vars ROOT, MAKEMAKE and EXT have to be specified externally, on the
'make' command line.
#ROOT=d:/home/IPv6SuiteWithINET
#MAKEMAKE=opp_nmakemake
#EXT=.vc

# for compiled-in NED files, remove -N from OPTS, and switch to the longer
version of ALL_MODEL_OPTS below
# with omnetpp-3.2 or newer, if you want to build windows dlls, add this to
OPTS below: -PINET_API
OPTS=-f -N -b $(ROOT) -c $(ROOT)/inetconfig$(EXT) -I.

CONTRACT_INCLUDES=-I$(ROOT)/Transport/Contract -I$(ROOT)/Network/Contract -
I$(ROOT)/NetworkInterfaces/Contract -I$(ROOT)/Base -I$(ROOT)/Util

ALL_INET_INCLUDES=$(CONTRACT_INCLUDES) -I$(ROOT)/Network/IPv4 -
I$(ROOT)/Network/AutoRouting -I$(ROOT)/Network/Queue -I$(ROOT)/Network/Quagga
-I$(ROOT)/Network/OSPFv2 -I$(ROOT)/Network/OSPFv2/Interface -
I$(ROOT)/Network/OSPFv2/MessageHandler -I$(ROOT)/Network/OSPFv2/Neighbor -
I$(ROOT)/Network/OSPFv2/Router -I$(ROOT)/Transport/TCP -
I$(ROOT)/Transport/TCP/flavours -I$(ROOT)/Transport/TCP/queues -
I$(ROOT)/Transport/RTP  -I$(ROOT)/Transport/UDP  -I$(ROOT)/NetworkInterfaces -
I$(ROOT)/Network/ARP -I$(ROOT)/NetworkInterfaces/Ethernet -
I$(ROOT)/NetworkInterfaces/EtherSwitch -I$(ROOT)/NetworkInterfaces/PPP -
I$(ROOT)/Applications/Generic -I$(ROOT)/Applications/Ethernet -
I$(ROOT)/Applications/TCPApp -I$(ROOT)/Applications/UDPApp -
I$(ROOT)/Applications/PingApp -I$(ROOT)/Util/HeaderSerializers -
I$(ROOT)/Nodes/INET -I$(ROOT)/Nodes/Wireless -I$(ROOT)/Nodes/Adhoc
# -I$(ROOT)/Applications/BRModel
ALL_MPLS_INCLUDES=-I$(ROOT)/Network/MPLS -I$(ROOT)/Network/LDP -
I$(ROOT)/Network/RSVP_TE -I$(ROOT)/Network/TED -I$(ROOT)/Network/Extras -
I$(ROOT)/Nodes/MPLS
ALL_IPv6_INCLUDES=-I$(ROOT)/Network/IPv6 -I$(ROOT)/Network/ICMPv6 -
I$(ROOT)/Nodes/IPv6
ALL_MF_INCLUDES=-I$(ROOT)/World -I$(ROOT)/Mobility -
I$(ROOT)/NetworkInterfaces/MFCore -I$(ROOT)/NetworkInterfaces/MF80211 -
I$(ROOT)/NetworkInterfaces/MF80211/macLayer -
I$(ROOT)/NetworkInterfaces/MF80211/phyLayer -
I$(ROOT)/NetworkInterfaces/MF80211/phyLayer/decider -
I$(ROOT)/NetworkInterfaces/MF80211/phyLayer/snrEval -
```

```
I$(ROOT)/NetworkInterfaces/Ieee80211 -I$(ROOT)/NetworkInterfaces/Ieee80211/Mac
-I$(ROOT)/NetworkInterfaces/Ieee80211/Mgmt -I$(ROOT)/NetworkInterfaces/Radio

#ALL_MODEL_OPTS=$(OPTS) -w $(ALL_INET_INCLUDES) $(ALL_MPLS_INCLUDES)
$(ALL_IPv6_INCLUDES)
ALL_MODEL_OPTS=$(OPTS) -n

#
# Example simulations don't contain C++ code, only NED, ini and other config
# files, so they don't need Makefiles. They all invoke the bin/INET
executable.
#

all: inetbase

inetbase:
      $(MAKEMAKE) $(OPTS) -n -r -X Documentation -X Etc -X Examples -X Tests -
X Experimental -X Obsolete -X tmp

      cd bin && $(MAKEMAKE) $(OPTS) -w -o INET $(ALL_INET_INCLUDES)
$(ALL_IPv6_INCLUDES) $(ALL_MPLS_INCLUDES) $(ALL_MF_INCLUDES)

      cd Applications && $(MAKEMAKE) $(OPTS) -n -r
      cd Network && $(MAKEMAKE) $(OPTS) -n -r
      cd NetworkInterfaces && $(MAKEMAKE) $(OPTS) -n -r
      cd Nodes && $(MAKEMAKE) $(OPTS) -n -r
      cd Transport && $(MAKEMAKE) $(OPTS) -n -r #-X RTP
      cd Base && $(MAKEMAKE) $(OPTS) -n -r -I../Util
      cd Util && $(MAKEMAKE) $(OPTS) -n -r $(ALL_INET_INCLUDES)
$(ALL_IPv6_INCLUDES) $(ALL_MPLS_INCLUDES) $(ALL_MF_INCLUDES)

      : # MF stuff follows
      cd World && $(MAKEMAKE) $(OPTS) -n -r -I../Util -I../Base
      cd Mobility && $(MAKEMAKE) $(OPTS) -n -r -I../Util -I../Base -I../World
      cd NetworkInterfaces/MFCore && $(MAKEMAKE) $(OPTS) -n -r
$(CONTRACT_INCLUDES) -I../../World
      cd NetworkInterfaces/MF80211 && $(MAKEMAKE) $(OPTS) -n -r
      cd NetworkInterfaces/MF80211/macLayer && $(MAKEMAKE) $(OPTS) -n
$(CONTRACT_INCLUDES) -I../../MFCore -I../phyLayer/snrEval
      cd NetworkInterfaces/MF80211/phyLayer && $(MAKEMAKE) $(OPTS) -n -r
      cd NetworkInterfaces/MF80211/phyLayer/decider && $(MAKEMAKE) $(OPTS) -n
$(CONTRACT_INCLUDES) -I../../../MFCore -I../../macLayer
      cd NetworkInterfaces/MF80211/phyLayer/snrEval && $(MAKEMAKE) $(OPTS) -n
$(CONTRACT_INCLUDES) -I../../../MFCore -I../../macLayer -I../../../../World
      cd NetworkInterfaces/Ieee80211 && $(MAKEMAKE) $(OPTS) -n -r
      cd NetworkInterfaces/Ieee80211/Mac && $(MAKEMAKE) $(OPTS) -n
$(CONTRACT_INCLUDES) -I../../MFCore
      cd NetworkInterfaces/Ieee80211/Mgmt && $(MAKEMAKE) $(OPTS) -n
$(CONTRACT_INCLUDES) -I../Mac -I../../Ethernet
      cd NetworkInterfaces/Radio && $(MAKEMAKE) $(OPTS) -n -n
$(CONTRACT_INCLUDES) -I../MFCore -I../Ieee80211/Mac -I../../World

      cd Nodes/IPv6 && $(MAKEMAKE) $(OPTS) -n -r $(ALL_INET_INCLUDES)
$(ALL_IPv6_INCLUDES)
```

```
	cd Applications/Generic && $(MAKEMAKE) $(OPTS) -n -r
$(CONTRACT_INCLUDES)
	cd Applications/Ethernet && $(MAKEMAKE) $(OPTS) -n -r
$(CONTRACT_INCLUDES) -I../../NetworkInterfaces/Ethernet
	cd Applications/PingApp && $(MAKEMAKE) $(OPTS) -n -r
$(CONTRACT_INCLUDES)
	cd Applications/TCPApp && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES)
-I../../Transport/TCP
	cd Applications/UDPApp && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES)

	cd Network/Contract && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES) -
I../IPv4 -I../IPv6 -I../ICMPv6
	cd Network/IPv4 && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES) -
I../ARP
	cd Network/Queue && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES) -
I../IPv4 -I../IPv6
	cd Network/ARP && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES) -
I../IPv4
	cd Network/AutoRouting && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES)
-I../IPv4 -I../IPv6 -I../ICMPv6
	cd Network/IPv6 && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES) -
I../ICMPv6
	cd Network/ICMPv6 && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES) -
I../IPv6

	: # FIXME reduce cross-dependency among MPLS, LDP, TED and RSVP_TE!
	cd Network/MPLS && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES) -
I../IPv4 -I../RSVP_TE -I../LDP -I../../Transport/TCP
	cd Network/LDP && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES) -
I../MPLS -I../IPv4 -I../TED -I../RSVP_TE -I../../Transport/UDP -
I../../Transport/TCP
	cd Network/RSVP_TE && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES) -
I../MPLS -I../IPv4 -I../TED
	cd Network/TED && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES) -
I../MPLS -I../IPv4 -I../RSVP_TE
	cd Network/Extras && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES)
	cd Network/Quagga && $(MAKEMAKE) $(OPTS) -n -r $(ALL_MPLS_INCLUDES)
$(ALL_INET_INCLUDES) $(ALL_MF_INCLUDES)

	cd Network/OSPFv2 && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES) -I.
-I../IPv4 -IRouter -IInterface -IMessageHandler -INeighbor
	cd Network/OSPFv2/Interface && $(MAKEMAKE) $(OPTS) -n -r
$(CONTRACT_INCLUDES) -I. -I.. -I../../IPv4 -I../Neighbor -I../Router -
I../MessageHandler
	cd Network/OSPFv2/MessageHandler && $(MAKEMAKE) $(OPTS) -n -r
$(CONTRACT_INCLUDES) -I. -I.. -I../../IPv4 -I../Interface -I../Neighbor -
I../Router
	cd Network/OSPFv2/Neighbor && $(MAKEMAKE) $(OPTS) -n -r
$(CONTRACT_INCLUDES) -I. -I.. -I../../IPv4 -I../MessageHandler -I../Interface
-I../Router
	cd Network/OSPFv2/Router && $(MAKEMAKE) $(OPTS) -n -r
$(CONTRACT_INCLUDES) -I. -I.. -I../../IPv4 -I../MessageHandler -I../Interface
-I../Neighbor

	cd NetworkInterfaces/Contract && $(MAKEMAKE) $(OPTS) -n
$(CONTRACT_INCLUDES)
```

```
	cd NetworkInterfaces/PPP && $(MAKEMAKE) $(OPTS) -n $(CONTRACT_INCLUDES)
-I../../Network/Queue
	cd NetworkInterfaces/Ethernet && $(MAKEMAKE) $(OPTS) -n
$(CONTRACT_INCLUDES) -I../../Network/Queue
	cd NetworkInterfaces/EtherSwitch && $(MAKEMAKE) $(OPTS) -n
$(CONTRACT_INCLUDES) -I../Ethernet

	cd Nodes/INET && $(MAKEMAKE) $(OPTS) -n -r $(ALL_INET_INCLUDES)
	cd Nodes/IPv6 && $(MAKEMAKE) $(OPTS) -n -r $(ALL_INET_INCLUDES)
	cd Nodes/Wireless && $(MAKEMAKE) $(OPTS) -n -r $(ALL_INET_INCLUDES)
	cd Nodes/Adhoc && $(MAKEMAKE) $(OPTS) -n -r $(ALL_INET_INCLUDES)
	cd Nodes/MPLS && $(MAKEMAKE) $(OPTS) -n -r $(ALL_INET_INCLUDES)
$(ALL_MPLS_INCLUDES)

	cd Transport/Contract && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES)
	cd Transport/UDP && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES) -
I../../Network/IPv4 -I../../Network/ICMPv6 -I../../Network/IPv6
	cd Transport/RTP && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES) -X
Profiles -X tmp
	cd Transport/RTP/Profiles/AVProfile && $(MAKEMAKE) $(OPTS) -n -r
$(CONTRACT_INCLUDES) -I../..
	cd Transport/TCP && $(MAKEMAKE) $(OPTS) -n -r $(CONTRACT_INCLUDES) -
I../../Network/IPv4 -I../../Network/ICMPv6 -I../../Network/IPv6
	cd Transport/TCP/flavours && $(MAKEMAKE) $(OPTS) -n -r
$(CONTRACT_INCLUDES) -I..
	cd Transport/TCP/queues && $(MAKEMAKE) $(OPTS) -n -r
$(CONTRACT_INCLUDES) -I..

	cd Util/HeaderSerializers && $(MAKEMAKE) $(OPTS) -n $(ALL_INET_INCLUDES)
$(ALL_IPv6_INCLUDES) $(ALL_MPLS_INCLUDES) $(ALL_MF_INCLUDES)

# UNUSED TARGET. It might only be needed with fully compiled-in NED files, or
when
# an example simulation contains C++ parts too (currently none have).
examples-dir:
	cd Examples && $(MAKEMAKE) $(OPTS) -n -r
	cd Examples/Ethernet && $(MAKEMAKE) $(OPTS) -n -r
	cd Examples/INET && $(MAKEMAKE) $(OPTS) -n -r
	cd Examples/OSPFv2 && $(MAKEMAKE) $(OPTS) -n -r
	cd Examples/Quagga && $(MAKEMAKE) $(OPTS) -n -r
	cd Examples/MPLS && $(MAKEMAKE) $(OPTS) -n -r
	cd Examples/IPv6 && $(MAKEMAKE) $(OPTS) -n -r
	cd Examples/RTP && $(MAKEMAKE) $(OPTS) -n -r #-X Data -X Multicast1 -X
Multicast2 -X Unicast
	cd Examples/Wireless && $(MAKEMAKE) $(OPTS) -n -r
	cd Examples/Adhoc && $(MAKEMAKE) $(OPTS) -n -r

	cd Examples/Ethernet/ARPTest && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/Ethernet/LANs && $(MAKEMAKE) $(ALL_MODEL_OPTS)

	cd Examples/INET/NClients && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/INET/FlatNet && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/INET/KIDSNw1 && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/INET/Multicast && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/INET/RouterPerf && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/INET/BulkTransfer && $(MAKEMAKE) $(ALL_MODEL_OPTS)
```

```
	cd Examples/INET/REDTest && $(MAKEMAKE) $(ALL_MODEL_OPTS)

	cd Examples/OSPFv2/Areas && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/OSPFv2/Backbone && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/OSPFv2/FullTest && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/OSPFv2/SimpleTest && $(MAKEMAKE) $(ALL_MODEL_OPTS)

	cd Examples/Quagga/FourRouters && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/Quagga/SimpleTest && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/Quagga/OSPFBackbone && $(MAKEMAKE) $(ALL_MODEL_OPTS)

	cd Examples/MPLS/LDP && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/MPLS/Net37 && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/MPLS/TestTE_Failure && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	: #cd Examples/MPLS/TestTE_Failure2 && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/MPLS/TestTE_Reroute && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/MPLS/TestTE_Routing && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/MPLS/TestTE_Tunnel  && $(MAKEMAKE) $(ALL_MODEL_OPTS)

	cd Examples/IPv6/NClients && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/IPv6/DemoNetworkEth && $(MAKEMAKE) $(ALL_MODEL_OPTS)

	cd Examples/Wireless/80211Lan && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/Wireless/Handover && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/Wireless/Throughput && $(MAKEMAKE) $(ALL_MODEL_OPTS)

	cd Examples/Adhoc/Mobility && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/Adhoc/Ieee80211 && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/Adhoc/MF80211 && $(MAKEMAKE) $(ALL_MODEL_OPTS)

	cd Examples/RTP/Data && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/RTP/Unicast && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/RTP/Multicast1 && $(MAKEMAKE) $(ALL_MODEL_OPTS)
	cd Examples/RTP/Multicast2 && $(MAKEMAKE) $(ALL_MODEL_OPTS)

tests-dir:
	cd Tests && $(MAKEMAKE) $(OPTS) -n -r -X IPv4 -X MPLS
	cd Tests/NewTCP && $(MAKEMAKE) $(OPTS) -w $(ALL_INET_INCLUDES)
$(ALL_IPv6_INCLUDES)
```