Qualité de Service et Performances des Protocoles de Transport dans l'UTRAN

Rani Makké

TELECOM PARIS
école nationale supérieure des télécommunications

Thèse

présentée pour obtenir le grade de docteur
de l'Ecole Nationale Supérieure des Télécommunications

Spécialité **: Informatique et Réseaux**

Rani MAKKE

Qualité de Service et Performances des Protocoles de Transport dans l'UTRAN

Soutenue le 3 juillet 2003 devant le jury composé de

Jean-Pierre Coudreuse	Président
Ken Chen	Rapporteurs
Guy Pujolle	
Jean-Yves Cochennec	Examinateurs
Bernard Cousin	
Samir Tohmé	Directeur de thèse

à mes parents,

à ma famille.

Résumé

Le 3GPP (*3rd Generation Partnership Project*) a choisi, dans sa Release 99, le protocole AAL2/ATM pour le transport des données sur les interfaces Iub et Iur du réseau d'accès de l'UMTS, nommé UTRAN (*UMTS Terrestrial Radio Access Network*). L'AAL-2 (*ATM Adaptation Layer - Type 2*) est une couche d'adaptation au-dessus de la couche ATM. L'AAL2 consiste à agréger plusieurs paquets de différents flux dans une même cellule ATM pour réduire le temps de remplissage surtout pour les applications temps-réel à bas débit. Les études menées sur l'AAL2 pour évaluer ses performances ne sont pas nombreuses, surtout dans le contexte de l'UTRAN où les contraintes temporelles sont strictes à cause des mécanismes de synchronisation des canaux de transport radio. Le lien entre le Node B et le RNC, où le protocole AAL2 est déployé, ne doit pas constituer le goulot d'étranglement (*Bottleneck*) pour les flux transportés. Le protocole AAL2 doit être bien étudié pour évaluer ses performances et ses capacités pour le transport des canaux radio.

L'AAL2 constitue un véritable protocole de transport qui se superpose au protocole de transport ATM. Or, dans les normes de l'AAL2, aucune notion de qualité de service (QoS) n'était définie pendant notre étude. En plus, les études faites étaient insuffisantes pour définir un modèle complet de la QoS dans l'AAL2. Nous avons alors contribué pour définir des classes de QoS au niveau AAL2, des paramètres de QoS, des capacités de transfert (AAL2-TC) ainsi que des critères de performance. Ensuite, nous proposons des schémas d'association (*mapping*) entre les différentes classes de service de l'UMTS et les classes de l'AAL2 d'une part et entre les classes de l'AAL2 et les classes de l'ATM d'autre part. Un schéma de partage de la bande passante entre les différentes classes est proposé ainsi qu'un schéma de contrôle d'admission des connexions AAL2. Dans le cas où plusieurs types de trafic seraient agrégés dans le même VC ATM, un mécanisme d'ordonnancement au niveau AAL2 est nécessaire pour pouvoir différencier entre les classes de services. Plusieurs algorithmes d'ordonnancement sont proposés au niveau de l'AAL2 et une comparaison entre ces mécanismes est réalisée dans plusieurs contextes pour évaluer les avantages et les inconvénients de chaque algorithme. Nous proposons un nouvel algorithme dynamique appelé DyWRR qui s'adapte avec le changement du trafic. Nous étudions aussi un paramètre important relatif au protocole AAL2 qui est le Timer-CU (*Timer - Combined Use*). Ce paramètre de temporisation a une influence sur le délai des paquets et sur le taux d'utilisation de la bande passante. Une étude fine et détaillée de ce paramètre est réalisée pour obtenir des résultats sur sa valeur optimale. Le choix de l'ATC (*ATM Transfer Capability*) pour le transport des connexions AAL2 fait une partie de notre étude. Le DBR et le SBR sont deux ATC proposées et une comparaison entre ces deux solutions est analysée. Quand on parle de l'AAL2 comme un protocole de transport, l'idée de la commutation AAL2 ne sera plus exclue. Au lieu de commuter les cellules ATM, on peut commuter les paquets AAL2 pour qu'on puisse séparer les connexions AAL2 dans un nœud donné du réseau. La commutation AAL2 présente des avantages mais également des inconvénients que nous traitons dans cette thèse.

Les résultats de l'étude sur l'AAL2 faite dans le cadre de cette thèse ont été utilisés pour une contribution à la normalisation au sein de l'ITU-T (*International Telecommunication Union -Telecommunication standardization sector*). Ces travaux de normalisation ont abouti à une nouvelle norme appelée ITU-T I.378 qui traite la problématique de la qualité de service et du contrôle de trafic au niveau de l'AAL2.

Dans la Release 5 du 3GPP, l'IP est introduit comme protocole de transport sur les interfaces Iub et Iur dans une optique de réaliser des réseaux "tout-IP". Ce protocole, dans sa version simple, ne peut pas garantir une qualité de service parce qu'il ne fournit qu'une seule classe de service pour tous les flux, la classe du meilleur effort (*Best Effort*). Des mécanismes de différenciation des services sont alors nécessaires comme *DiffServ* ou *IntServ* dans le contexte de l'UTRAN. Plusieurs architectures pour le transport en IP sur l'interface Iub sont présentées. D'ailleurs, ces solutions en IP introduisent une charge supplémentaire à cause de la longueur des en-têtes. Sachant que les opérateurs des télécommunications s'intéressent à l'optimisation de la bande passante sur les liens appelés *Last Mile Link*, une étude est alors réalisée dans cette thèse pour évaluer les performances des solutions en IP dans l'UTRAN.

Abstract

The 3GPP (3rd Generation Partneship Project) has chosen, in the Release 99, the AAL2/ATM protocol as a transport protocol on the Iub et Iur interfaces of the UMTS access network called UTRAN (UMTS Terrestrial Radio Access Network). The AAL-2 (ATM Adaptation Layer - Type 2) is an adaptation layer above the ATM layer. It consists in aggregating several packets from different flows in the same ATM cell in order to decrease the packetization delay especially for low bit rate real time applications. There are few studies about the performance of the AAL2 protocol but there are no enough studies of the AAL2 in the special context of the UTRAN where the time constraints become stringent because of the synchronisation mechanisms of the radio channels. The link between a Node B and a RNC, where the AAL2 protocol is deployed, should not be the bottleneck of the network. The AAL2 protocol should be well studied to evaluate its performance and its capabilities for the transport of radio channels.

The AAL2 is a real transport protocol but in the specifications, there is no quality of service (QoS) definition at the AAL2 layer. We contributed to define some classes of service, QoS parameters, AAL2 transfer capabilities (AAL2-TC) and performance criterions. Then, we propose a mapping scheme on the one hand between the different classes of service of the UMTS and the classes of the AAL2 layer and on the other hand between AAL2 classes and ATM classes. We propose a bandwidth sharing scheme between different classes as well as a connection admission control scheme for AAL2 connections. In the case where several types of flows are aggregated in the same ATM VC, a scheduling mechanism is needed to differentiate between the different classes of service. Several scheduling algorithms are proposed at the AAL2 layer and a comparison between them is done in several contexts in order to evaluate the advantages and drawbacks of each algorithm. We propose also a new dynamic scheduling algorithm called DyWRR. Then, we studied an important parameter of the AAL2 protocol that is the Timer-CU (Timer-Combined Use). This parameter has an impact on the transfer delay and on the bandwidth efficiency. A detailed study of this parameter is achieved in order to obtain results about the optimal value of this parameter. The choice of the ATC (ATM Transfer Capability) for the transport of AAL2 connections represents a part of our study. The DBR and the SBR are two proposed solutions and a comparison between their performances is done. The AAL2 switching technology is an important aspect in the AAL2 networks. Small and variable size packets (mini-cells) are switched rather than ATM cells. The AAL2 switching technology is studied to evaluate its advantages and drawbacks and to compare it with the ATM switching technology.

The results of our study on the AAL2 were used in a contribution to the standardization works of the ITU-T (International Telecommunication Union – Telecommunication standardization sector). These standardization efforts had leaded to a new standard called ITU-T I.378 which treats the problematic of the quality of service and traffic control at the AAL2 layer.

In the Release 5 of the 3GPP, the IP protocol was chosen to transport radio channels on Iub and Iur interfaces. The IP protocol cannot guarantee a quality of service without any special mechanisms like DiffServ or IntServ because in the present networks, only one class of service is used, the Best Effort class. Several architectures are presented for the use of the IP protocol on the Iub interface. These solutions introduce additional overhead because of the long size headers. The optimization of the bandwidth especially at the Last Mile Link is an important issue for the telecommunication operators. Then, we studied several solutions in order to evaluate their performance in the special context of the UTRAN.

Remerciements

Cette thèse s'est déroulée, au sein du département Informatique et Réseaux de l'Ecole Nationale Supérieure des Télécommunications (Télécom-Paris), sous la direction de Monsieur Samir Tohmé, professeur et animateur du groupe de recherche "Réseaux à Haut Débit". Je tiens à lui exprimer ma profonde reconnaissance pour m'avoir fait confiance en acceptant de diriger cette thèse ainsi que pour ses encouragements et ses conseils tout au long de ces trois années.

Mes remerciements chaleureux vont ensuite à toutes les personnes avec qui j'ai été amené à collaborer à travers cette thèse, particulièrement aux équipes de France Télécom R&D (Lannion et Issy les Moulineaux) qui ont montré leur intérêt à nos travaux à l'ENST et avec qui nous avons participé aux projets MINICEL puis TIPS. Je remercie également les équipes de Mitsubishi Electric ITE (Rennes) qui ont participé au projet MINICEL et avec qui nous avons collaboré pendant un an et demi.

Je remercie vivement Monsieur Ken Chen, professeur à l'Université de Paris 13, et Monsieur Guy Pujolle, professeur à l'Université Pierre et Marie Curie, d'avoir accepté la lourde tâche d'être rapporteurs de cette thèse.

Je tiens à remercier Monsieur Jean-Pierre Coudreuse, Papa de l'ATM et directeur de Mitsubshi Electric ITE (Rennes), d'avoir accepté de présider le jury de cette thèse.

Je remercie également Monsieur Jean-Yves Cochennec, expert senior à France Télécom R&D (Lannion) et coordinateur des projets avec l'ENST ainsi que Monsieur Bernard Cousin, professeur à l'université de Rennes 1, de m'avoir fait l'honneur de bien vouloir participer au jury de cette thèse.

Les membres du département Informatique et Réseaux de l'ENST, les professeurs, les collègues et les amis ainsi que les membres de l'administration de l'école sont priés de trouver ici l'expression de ma sincère gratitude.

Je garde une place toute particulière à ma famille : mon père Fawzi, ma mère Jamilé, ma sœur Rana et mon frère Ali. Je voudrais leur exprimer toute ma profonde reconnaissance et je leur remercie du fond de mon cœur parce qu'ils m'ont constamment aidé, malgré la distance, par leur soutien moral et leurs encouragements pour achever cette thèse.

Mes derniers remerciements vont à tous les membres de ma grande famille pour leurs encouragements et pour la confiance qu'ils m'ont accordée. Enfin, je remercie chaleureusement ma grand-mère, qui ne lira jamais ces pages et qui y a pourtant tant contribué, pour n'avoir jamais douté de moi et m'avoir donné son entière confiance.

Table des matières

Liste des figures

Liste des tableaux

Chapitre 1

1 Introduction générale

1.1 Historique

Pendant les dernières décennies, le marché de la téléphonie mobile a connu une grande évolution surtout avec l'apparition des systèmes radio-mobiles cellulaires. Les premiers réseaux cellulaires ont été déployés aux Etats-Unis à partir de 1978 avec le système AMPS (*Advanced Mobile Phone System*) et en Europe en 1981 avec le système NMT (*Nordic Mobile Telephone*). Ces réseaux, dits de première génération, utilisaient un système de transmission analogique et un multiplexage fréquentiel FDMA (*Frequency Division Multiple Access*). La densité d'abonnés restait relativement faible et la mobilité était facile à gérer puisque les cellules étaient de grande taille. Une cellule radio est une zone géographique couverte par une antenne de transmission. Un utilisateur est alors en mesure de passer d'une cellule à une autre sans coupure de communication. Ce passage, appelé *handover*, correspond à la fonction permettant au terminal de changer de cellule sans interruption. Les réseaux cellulaires de première génération ont été les premiers à permettre à un utilisateur mobile d'utiliser un téléphone de façon continue, n'importe où dans la zone de service d'un opérateur.

Les réseaux cellulaires de deuxième génération ont été conçus au milieu des années 80. Ils utilisent une transmission numérique qui a l'avantage d'augmenter le débit grâce aux codes correcteurs d'erreurs. Le principal système de deuxième génération est le GSM (*Global System for Mobile communications*) qui est basé sur une technique d'accès FDMA/TDMA (*Frequency Division Multiple Access/Time Division Multiple Access*). Le GSM fonctionne dans la bande de fréquence de 900 MHz. Il existe d'autres systèmes de deuxième génération comme le DCS1800 (*Digital Cellular System 1800*) qui fonctionne dans la bande de fréquence de 1800 MHz. Aux Etats-Unis, le système utilisé est le PCS (*Personal Communications Services*) fonctionnant dans la bande de 1900 MHz. Au Japon, le système déployé est le PDC (*Personal Digital Cellular*).

Jusqu'à la fin de l'année 1999, les services de la parole ont représenté la majorité du trafic dans les réseaux GSM. La transmission des données reste marginale et les débits ne peuvent pas dépasser 9,6 kbit/s. L'organisme de normalisation ETSI (*European Telecommunication Standards Institut*) a standardisé deux nouveaux services pour le GSM, le HSCSD (*High Speed Circuit Switched Data*) et le GPRS (*General Packet Radio Service*). Dans le premier, le débit peut atteindre 64 kbit/s et dans le deuxième on peut atteindre des débits de l'ordre de 160 kbit/s pour la transmission des données. Une évolution de la norme GPRS a mené à un nouveau service appelé EDGE (*Enhanced Data rates for the GSM Evolution*) qui envisage des débits de transmission de l'ordre de 384 kbit/s.

En 1985, l'Union Internationale des Télécommunications (UIT) a commencé ses études sur les réseaux FPLMTS (*Future Public Land Mobile Telecommunication System*), renommés IMT-2000 (*International Mobile Telecommunications for the year 2000*) en 1993. De son coté, l'ETSI a commencé en 1990 ses études sur les réseaux de mobiles pour l'Europe sous le nom de UMTS (*Universal Mobile Telecommunication System*). L'UMTS n'est qu'un élément de la famille IMT-2000 dite de troisième génération.

Les réseaux mobiles de troisième génération offrent des services de voix ainsi que des services de transmission des données jusqu'à 384 kbit/s avec une vitesse de déplacement de plus de 500 km/h, jusqu'à 512 kbit/s en extérieur urbain avec une vitesse de 120 km/h et jusqu'à 2 Mbit/s à moins de 10

km/h. Grâce aux débits élevés de cette génération, des services multimédia sophistiqués peuvent être fournis.

En Europe, une interface radio appelée UTRA (*UMTS Terrestrial Radio Access*) a été choisie par l'ETSI pour la troisième génération. Cette interface utilise la technique d'accès CDMA (*Code Division Multiple Access*). Le réseau d'accès global en UMTS est appelé UTRAN (*UMTS Terrestrial Radio Access Network*) qui est divisé en deux parties : la partie radio UTRA et une partie qui interconnecte les différents nœuds de ce réseau.

1.2 Contexte des études

Les réseaux UMTS doivent fournir des nouveaux services de communication allant de la téléphonie classique jusqu'aux services multimédia. Plusieurs classes de services ont été définies dans le cadre des travaux du 3GPP (*3rd Generation Partnership Project*) [23] : *Conversational, Streaming, Interactive, Background*. Chacune de ces classes garantit le transport de bout en bout d'un certain type de trafic. Ces classes de service sont transportées par plusieurs sous-réseaux interposés, particulièrement le réseau d'accès UTRAN qui est le sujet de notre étude.

L'infrastructure du réseau d'accès radio terrestre de l'UMTS, appelé UTRAN (*UMTS Terrestrial Radio Access Network*), doit être capable de gérer les flux transportés par les canaux radio qui sont étendus entre le terminal mobile et le RNC (*Radio Network Controller*). La liaison entre le terminal et le Node B (équivalent au BTS dans les systèmes GSM) est une interface radio dont les ressources sont rares. L'interconnexion entre le Node B et le RNC est une interface appelée Iub. Deux RNC sont interconnectés par une interface appelée Iur. Sur ces deux interfaces, le protocole de transport utilisé ne doit pas constituer le goulot d'étranglement (*bottleneck*) une fois que le trafic a passé par la ressource chère qui est l'interface radio. Le protocole de transport sur les interfaces Iub et Iur doit être bien étudié pour pouvoir garantir les services offerts par l'UMTS.

Dans sa Release 99, le 3GPP a choisi l'AAL-2 (*ATM Adaptation Layer – Type 2*) [64] comme protocole de transport sur les interfaces Iub et Iur de l'UTRAN. L'AAL2 a été normalisé par l'UIT (Union Internationale des Télécommunications) en 1997 dans une nouvelle version adaptée au transport des applications à bas débits. En effet, pour réduire le temps de remplissage des cellules ATM, qui peut atteindre 6 ms pour un débit de 64 kbit/s, on multiplexe plusieurs petits paquets (les mini-cellules) dans une même cellule ATM. Plusieurs flux sont alors multiplexés dans un même VC (*Virtual Channel*) et la séparation des flux se fait à l'aide d'un identificateur placé dans l'en-tête au début de chaque mini-cellule. Un mécanisme de temporisation est mis en place pour réduire le temps de remplissage en cas où le débit est faible. La valeur de ce temporisateur nommé Timer-CU (*Timer-Combined Use*) a un impact sur l'efficacité de la bande passante ainsi que sur le délai de transfert des mini-cellules et la variation du délai (la gigue). Une faible valeur du Timer-CU peut entraîner une perte de bande passante et une valeur élevée introduit des délais parfois trop élevés. Dans l'UTRAN, les contraintes temporelles sont strictes à cause de la synchronisation des couches radio et par suite, le transport sur les interfaces Iub et Iur ne doit pas introduire des délais supplémentaires élevés. L'immigration du système GSM au système UMTS va se faire progressivement, et le réseau d'accès de l'UMTS va utiliser au départ l'infrastructure déjà existante du GSM. Les deux systèmes vont coexister pour une certaine période et vont partager les mêmes ressources physiques, c'est pourquoi l'optimisation de l'utilisation des liens physiques constitue un problème essentiel, d'où l'étude de la valeur optimale du Timer-CU est importante.

L'AAL-2 est en soi un nouveau protocole de transport de mini-cellules qui se superpose au transport de cellules ATM. Une séparation entre les différents flux est possible grâce à l'en-tête au début de chaque mini-cellule. La définition de la qualité de service (QoS : *Quality of Service*) est alors possible au niveau de l'AAL2 et par suite des paramètres de QoS sont à définir. Des classes de service AAL2 sont à définir pour séparer entre les différents types de flux et un *mapping* entre les classes de l'UMTS et celles de l'AAL2 est aussi à définir. Des mécanismes d'ordonnancement sont nécessaires

pour différencier les classes de service au niveau de l'AAL2. Un mécanisme d'ordonnancement bien adapté au cas de l'AAL2 doit être choisi.

L'AAL2 requiert une signalisation qui lui est propre afin d'établir et de libérer les connexions AAL2 multiplexées dans les connexions ATM. L'établissement d'une connexion AAL2 doit être basé sur un mécanisme de contrôle d'admission (CAC : *Connexion Admission Control*). Une fonction CAC bien adaptée au cas de l'AAL2 doit être définie pour pouvoir garantir la qualité de service négociée lors de l'établissement de la connexion. Un schéma de partage de la bande passante entre les différentes classes de l'AAL2 est encore à définir.

L'AAL2 en tant que protocole de transport conduit naturellement à la fonction de commutation. A la différence de la commutation ATM, les entités commutées ont des longueurs qui peuvent aller de 4 octets à 48 octets (en-têtes de 3 octets compris). La commutation AAL2 introduit un nouveau délai dû au processus d'extraction des mini-cellules à partir des cellules ATM à l'entrée du commutateur et au processus d'insertion des mini-cellules dans les cellules ATM à la sortie du commutateur. Cet inconvénient peut être récompensé par l'impact de la commutation AAL2 sur l'utilisation des liens surtout dans le cas où les liens entrants seraient faiblement chargés. La commutation AAL2 est indispensable dans le cas où les connexions AAL2 supportées par le même VC ATM auraient des destinations différentes. La commutation ATM peut remplacer la commutation AAL2 dans le cas où toutes les connexions AAL2 d'un même VC auraient la même destination. Une comparaison entre ces deux technologies de commutation est à évaluer pour déduire les avantages et les inconvénients de la commutation AAL2.

Dans sa Release 5, le 3GPP a choisi l'IP (*Internet Protocol*) [84] comme protocole de transport sur les interfaces Iub et Iur de l'UTRAN dans une optique de déploiement des réseaux tout-IP. L'introduction de l'IP dans l'UTRAN doit être faite avec précaution pour pouvoir garantir la qualité de service exigée par les flux transportés dans l'UTRAN. Plusieurs solutions sont proposées pour le transport en IP dans le but de garantir une qualité de service acceptable et d'assurer une utilisation optimale des liens de l'UTRAN.

1.3 Cadre général et objectifs de la thèse

Au cours des deux premières années de notre thèse, nous avons travaillé sur un projet en collaboration entre l'Ecole Nationale Supérieure des Télécommunications (ENST), France Télécom R&D et Mitsubishi Electric ITE. Ce projet a été labellisé par le RNRT (Réseau National de Recherche en Télécommunications) sous le nom du "projet MINICEL". Le but de ce projet était de réaliser une étude théorique et par simulation du protocole AAL2 dans le contexte de l'UTRAN. Les résultats de cette étude devaient être utilisés pour la réalisation d'une maquette de démonstration dans laquelle un simulateur de l'UTRAN avec un commutateur AAL2 et des générateurs de trafic sont implémentés.

Dans notre étude, nous nous étions intéressés à plusieurs problématiques concernant le transport des flux AAL2 dans l'UTRAN. Nous identifions ci-dessous les points essentiels de notre étude:

- La nature des flux transportés dans l'UTRAN et les modèles de trafic convenables.
- L'influence des couches RLC, MAC et FP sur le profil du trafic.
- Définition de la qualité de service (QoS) au niveau AAL2 avec des paramètres de QoS et des classes de QoS AAL2 ainsi que des capacités de transfert AAL2.
- Gestion des flux dans l'UTRAN et schéma de partage de la bande passante.
- La notion de la bande passante équivalente et la fonction de contrôle d'admission CAC.
- L'étude de la valeur optimale du Timer-CU.
- Les mécanismes d'ordonnancement dans la couche AAL2.

- Le choix du meilleur ATC au niveau ATM.
- Les avantages et les inconvénients de la commutation AAL2.
- Conception d'un modèle de simulation du réseau UTRAN et implémentation du protocole AAL2 ainsi que des couches radio et les modèles de trafic définis.

Pendant notre troisième année de thèse, nous avons travaillé sur un projet nommé TIPS en collaboration entre l'ENST et France Télécom R&D. Le but de ce projet était d'étudier les performances de différentes solutions pour le transport en IP (Release 5) au sein de l'UTRAN et de comparer ses solutions pour conclure les avantages et les inconvénients de chacune surtout sur le *Last Mile Link*. Ce dernier constitue la connexion entre le Node B et le *Edge Router* qui est le dernier nœud du réseau de transport avant le Node B. Un modèle de simulation est défini et implémenté avec toutes les couches protocolaires dans le contexte d'un UTRAN IP.

1.4 Plan du rapport

Notre rapport de thèse est divisé en plusieurs chapitres selon les sujets d'études. Dans le deuxième chapitre, nous introduisons les réseaux UMTS en focalisant la présentation sur l'architecture générale du réseau UTRAN ainsi que l'architecture des couches protocolaires. Nous présentons aussi le fonctionnement de quelques couches radio qui ont un impact sur le trafic transporté sur les interfaces Iub et Iur. Cette étude est nécessaire pour modéliser les flux transportés au niveau de la couche de transport AAL2/ATM.

Dans le chapitre 3, nous présentons les modèles des sources de trafic dans l'UTRAN, l'influence des couches radio sur le profil de ses flux pour en déduire des modèles réalistes au niveau de la couche AAL2.

Le chapitre 4 introduit les réseaux ATM et focalise l'étude sur la couche d'adaptation AAL2 avec les fonctions des sous-couches de l'AAL2 (la SSCS et ses dérivés et la CPS) ainsi que les mécanismes de multiplexage et de temporisation. Dans ce chapitre, nous définissons une notion de qualité de service (*QoS: Quality of Service*) au niveau AAL2 avec des paramètres de QoS, des classes de QoS et des capacités de transfert ATC (*ATM Transfer Capability*) au niveau AAL2 ainsi que des critères de performance du protocole AAL2.

Le chapitre 5 traite les aspects liés à la qualité de service (*QoS*) et la gestion des flux dans l'UTRAN. Nous présentons l'architecture générale de la qualité de service dans l'UMTS et nous focalisons l'étude sur la position de l'AAL2 dans cette architecture. Nous proposons une association ou une traduction (*mapping*) entre les classes de service de l'UMTS et celles de l'AAL2 ainsi qu'entre celles de l'AAL2 et celles de l'ATM sous-jacent. Nous proposons aussi un schéma de gestion des flux ainsi qu'un mécanisme de contrôle d'admission CAC (*Connection Admission Control*) pour les connexions AAL2.

Dans le chapitre 6, nous étudions l'impact du *Timer-CU* sur les performances du protocole AAL2 dans différents cas de figure en fonction de la valeur du PCR (*Peak Cell Rate*). Nous faisons la différenciation des classes au niveau AAL2 et nous comparons plusieurs mécanismes d'ordonnancement. Nous abordons aussi le sujet de la commutation AAL2 : sa nécessité, ses avantages et ses inconvénients et nous faisons par simulation une comparaison avec la commutation ATM. Une comparaison entre différentes capacités de transfert au niveau ATM (*ATC : ATM Transfer Capability*) est aussi réalisée.

Dans le chapitre 7, nous introduisons les différentes approches pour le transport des canaux radio sur l'interface Iub dans le cas d'un UTRAN-IP et nous étudions les performances du protocole IP en faisant une comparaison entre plusieurs solutions de transport en IP.

Dans le chapitre 8, nous présentons une conclusion générale et une synthèse sur l'étude faite dans le cadre de notre thèse et nous proposons des perspectives et des axes d'étude qui constituent une ouverture sur des nouveaux domaines de recherche.

Chapitre 2

2 Le réseau terrestre d'accès radio (UTRAN)

2.1 Introduction

L'UMTS (*Universal Mobile Telecommunication System*) fait partie de la famille IMT-2000 (*International Mobile Telecommunications for the year 2000*) des réseaux mobiles de troisième génération. Ces systèmes se présentent comme des concurrents des systèmes de deuxième génération déjà déployés. La troisième génération doit apporter un plus par rapport à la précédente qui peut s'exprimer par une qualité de service au moins comparable avec celle fournie par les réseaux fixes. De plus, les réseaux de troisième génération doivent fournir des nouvelles avancées incluant l'itinérance mondiale, une large gamme de services à haut débit, des services audio-visuels et l'utilisation d'un seul terminal dans différents environnements radio.

Le réseau d'accès radio de l'UMTS est une partie importante où les ressources sont chères. Dans les spécifications de l'UMTS, deux grandes catégories de réseaux d'accès existent : le réseau de satellites d'accès radio (*USRAN : UMTS Satellite Radio Access Network*) et le réseau terrestre d'accès radio (*UTRAN : UMTS Terrestrial Radio Access Network*). L'interface radio est une ressource rare ainsi que les liens entre les nœuds du réseau d'accès. Le réseau d'accès radio doit être bien dimensionné pour qu'il puisse garantir la qualité de service requise par les nouveaux services de l'UMTS. Notre étude s'intéresse à la qualité de service dans le réseau terrestre d'accès radio et aux performances des protocoles de transport sur les interfaces Iub et Iur de l'UTRAN. En effet, l'UMTS ne va pas remplacer les anciens systèmes de communication mobile, comme le GSM, dès les premiers jours de son déploiement, mais il va coexister, dans des îlots, avec ces systèmes et va prendre leur place progressivement. L'UTRAN va alors utiliser les infrastructures déjà existantes et doit optimiser l'utilisation des ressources pour pouvoir transporter les nouveaux services à haut débit de l'UMTS tel que le transfert des données et les services multimédias.

Dans ce chapitre, nous présentons de manière générale les réseaux UMTS et nous focalisons l'étude sur le réseau d'accès UTRAN et les fonctionnalités de ses nœuds ainsi que les mécanismes qui ont un impact direct sur le transport des flux dans l'UTRAN.

2.2 Architecture globale de l'UMTS

Le système UMTS fait distinction entre la partie accès et les autres parties. Globalement, il peut être divisé en trois domaines principaux comme le montre la figure 2.1.

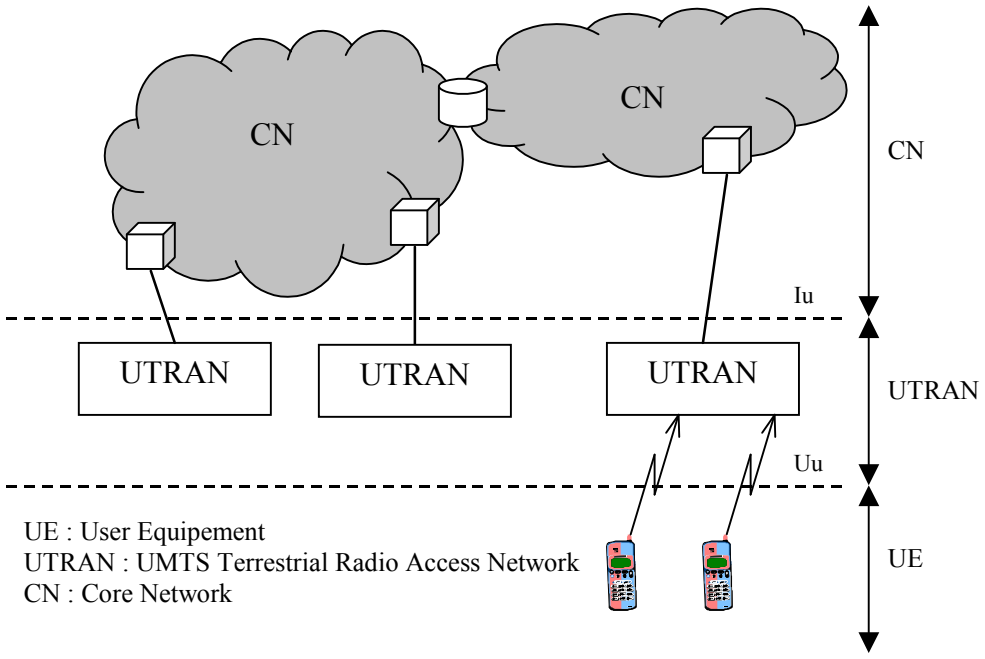


Figure 2.1 Architecture générale de l'UMTS

- **Le réseau cœur (*Core Network*) :** il assure la connexion entre les différents réseaux d'accès et entre le réseau UMTS et les autres réseaux comme le réseau téléphonique (*PSTN : Public Switched Telephone Network*), le réseau GSM, le réseau RNIS (Réseau Numérique à Intégration de Services ou *ISDN : Integrated Services Digital Network*), etc. Il fournit le support des services de télécommunications UMTS et gère les informations de localisation des utilisateurs mobiles ainsi qu'il contrôle les services et les caractéristiques du réseau. Le réseau cœur est composé de deux domaines : le domaine à commutation de circuits CS (*Circuit Switched domain*) et le domaine à commutation de paquets PS (*Packet Switched domain*).
- **Le réseau d'accès radio :** il gère les ressources radio, l'établissement, la maintenance et la libération des canaux radio entre le terminal et le réseau cœur (*Core Network*). IL permet aux utilisateurs mobiles de communiquer avec le réseau cœur. Deux catégories de réseau d'accès sont définies : le réseau de satellites d'accès radio USRAN et le réseau terrestre d'accès radio UTRAN.
- **L'équipement d'utilisateur (*User Equipement*) :** c'est le terminal mobile qui est en charge d'établir une communication entre l'utilisateur et le réseau. Il est connecté par une interface radio au réseau d'accès radio UTRAN.
- **Le système d'opération et de maintenance :** il est utilisé par l'opérateur du réseau pour configurer les équipements et les maintenir en cas de pannes.

L'interface Iu assure la connexion entre le réseau d'accès UTRAN et le réseau cœur CN. L'interface radio Uu assure la connexion entre le terminal UE et le réseau d'accès UTRAN.

2.3 Architecture globale de l'UTRAN

Le réseau terrestre d'accès radio [94] de l'UMTS (*UTRAN*) assure le transport des flux entre le terminal mobile et le réseau cœur. L'architecture globale de l'UTRAN est présentée dans la figure 2.2. Un réseau UTRAN est composé d'un ensemble de RNS reliés au réseau cœur à travers l'interface Iu.

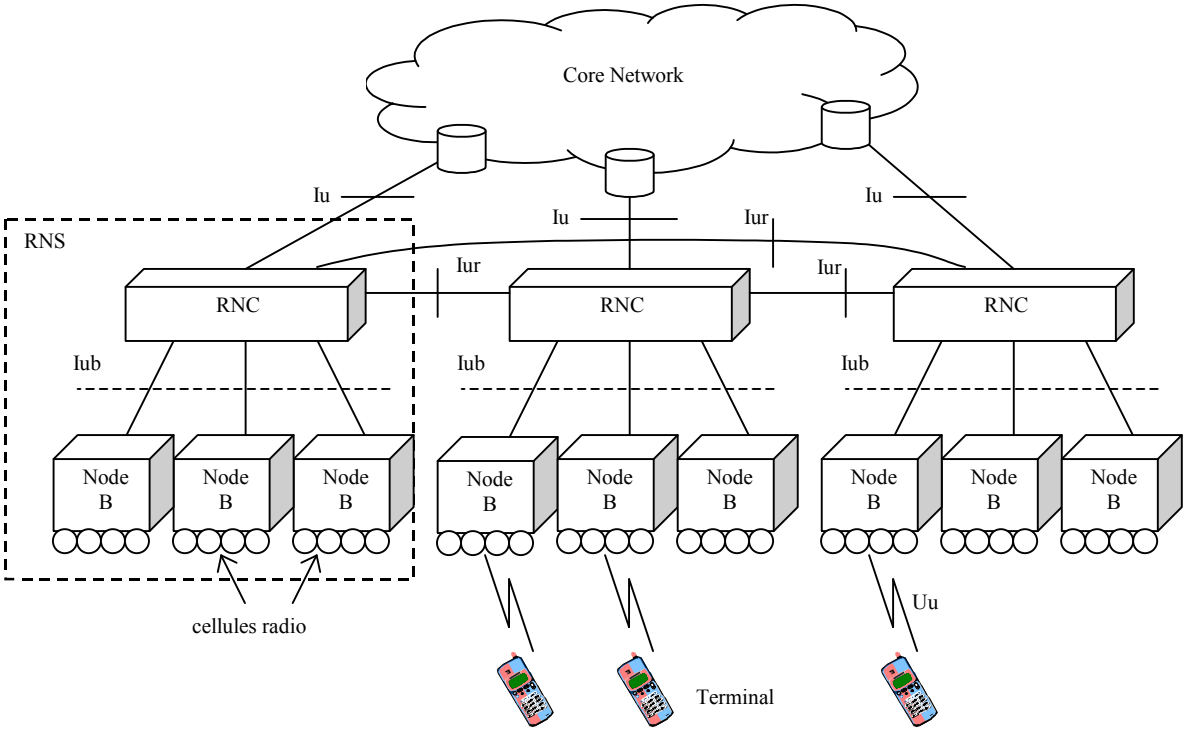


Figure 2.2 Architecture globale de l'UTRAN

2.3.1 Le RNS (*Radio Network Subsystem*)

C'est un sous-réseau constitué d'un seul RNC et de plusieurs Node B. Les Node B sont reliés entre eux par l'interface Iub et deux RNS sont reliés par l'interface Iur. Le RNS est connecté au réseau cœur par l'interface Iu.

2.3.2 Le Node B

Le modèle logique d'un Node B est représenté dans la figure 2.3. C'est une entité logique reliée à un RNC par l'interface Iub. Le Node B correspond à un BTS dans le système GSM. Il contient les fonctions de transmission radio (modulation, démodulation, codage, etc.). Il est responsable de la configuration des cellules radio (la gestion des fréquences porteuses, les codes des cellules, la configuration des canaux, etc.), de la gestion des canaux de transport communs et dédiés, de la synchronisation, de la gestion de la signalisation de l'interface Iub ainsi que du maintien des liens et du partage de la charge.

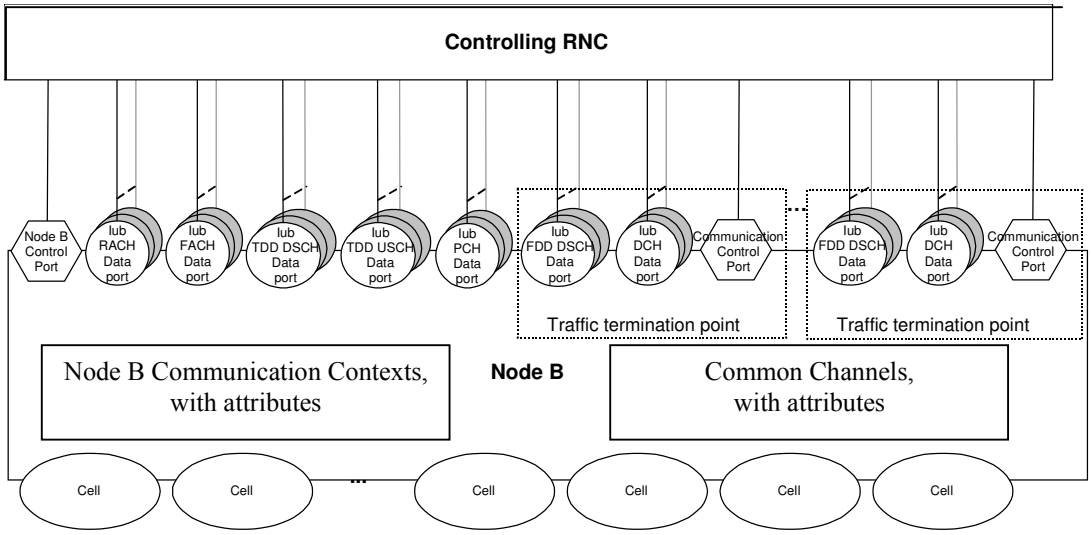


Figure 2.3 Modèle logique du Node B

Un Node B doit être capable de gérer jusqu'à quatre fréquences porteuses. Théoriquement, chaque porteuse fournit un débit de 2 Mbit/s par cellule radio. Ils existent plusieurs types de cellules radio selon l'environnement. La figure 2.4 représente les différents types de cellules radio.

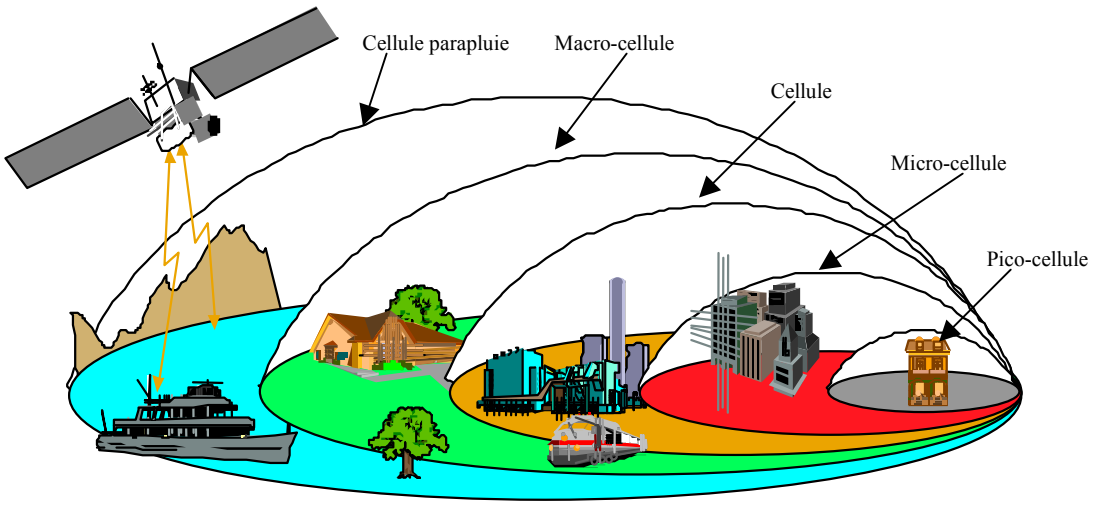


Figure 2.4 Les environnements définis dans les réseaux UMTS

- Pico-cellule : Quelques mètres de diamètre. Elle est utilisée dans les environnements de haute densité de population comme les bureaux, les supermarchés, etc.
- Micro-cellule : Quelques dizaines de mètres de diamètre. Elle est utilisée dans les rues et entre les immeubles des quartiers.
- Cellule : Ce type de cellule couvre les zones urbaines. Le diamètre d'une cellule varie entre quelques centaines de mètres et quelques kilomètres.
- Macro-cellule : C'est une grande cellule dont le diamètre peut atteindre quelques dizaines de kilomètres. Elle est utilisée sur les autoroutes et pour les véhicules à grande vitesse.

- Cellule parapluie : Son diamètre peut atteindre quelques centaines de kilomètres de diamètre. Elle couvre des très larges zones comme les océans et les déserts.

2.3.3 Le RNC (*Radio Network Controller*)

Le RNC est responsable de la gestion et du contrôle des canaux radio (établissement/maintien/libération des connexions radio). Il est aussi responsable de la gestion du *handover* quand un terminal mobile se déplace d'une cellule radio vers une autre. Il gère les mécanismes de contrôle de puissance dans les deux directions montante et descendante (*uplink* et *downlink*). Dans le cas du *soft handover* et de la macro-diversité, qu'on va expliquer dans le paragraphe 2.3.4, le *Serving*-RNC gère les mécanismes de division et de recombinaison des paquets FP-PDU (*Frame Protocol – Packet Data Unit*) ainsi que le contrôle de la qualité du lien radio. Les fonctions du contrôle d'admission sont gérées par le RNC.

Deux types de RNC sont définis :

- *Serving*-RNC (S-RNC) : C'est le RNC qui maintient la connexion avec le réseau cœur quand le mobile est en *soft handover.* Il assure les fonctions de division/recombinaison (*splitting/recombination*) dans le cas du *soft handover* pour acheminer un seul flux vers l'interface Iu.
- *Drift*-RNC (D-RNC) : Il achemine les flux du S-RNC vers le Node B qui gère la connexion avec le terminal mobile et vice-versa. Il assure la fonction de commutation pour garder un seul point d'interconnexion avec le réseau cœur. Le modèle logique du D-RNC est présenté dans la figure 2.5.

Le RNC contient les terminaisons des canaux et des couches radio (RLC et MAC) qui sont étendus jusqu'au terminal mobile. Il termine aussi les couches protocolaires de transport sur les interfaces Iub et Iur avec les Node B et les autres RNC respectivement.

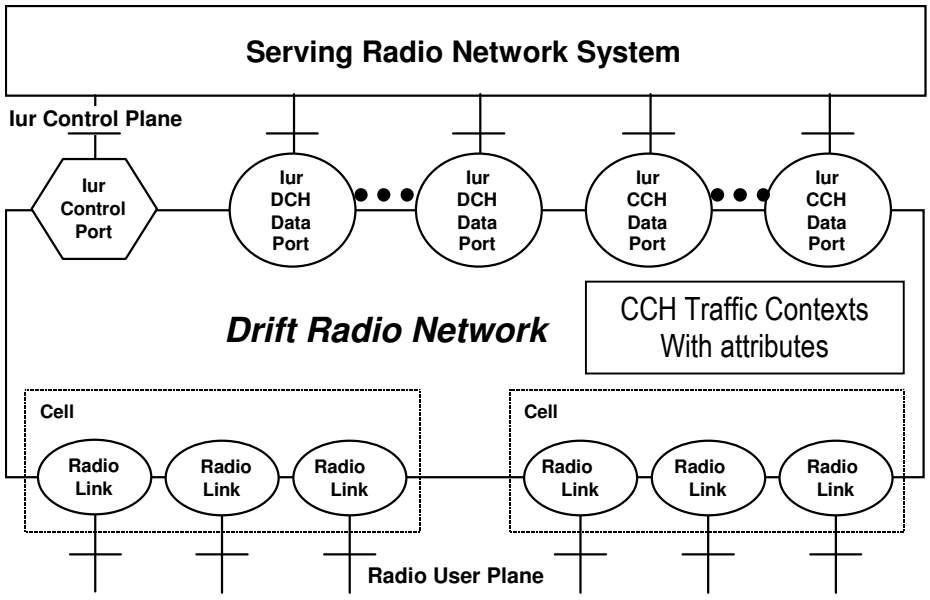


Figure 2.5 Modèle logique du RNC (vu de l'interface Iur : le D-RNC)

2.3.4 Le *soft handover* et la macro-diversité

La technique d'accès CDMA (*Code Division Multiple Access*) permet d'établir plusieurs connexions entre un terminal mobile et les stations de base (les Node B) afin de maintenir la communication en cas de passage d'une cellule radio à une autre (dans le cas d'un *handover*). Un terminal mobile peut être connecté en même temps à deux ou plusieurs cellules radio et quand il passe d'une cellule à une autre, il libère la connexion avec l'ancienne cellule sans interrompre la communication. Ce mécanisme de *handover* est appelé *soft handover* et il est différent du mécanisme de *hard handover* qui est utilisé dans les systèmes GSM. Dans le cas du *hard handover*, un mobile peut être connecté à une seule cellule radio à un moment donné, et quand il passe d'une cellule à une autre, il coupe sa connexion avec l'ancienne cellule et établit une nouvelle connexion avec la nouvelle cellule. Cette brève coupure de communication engendre des pertes de paquets dans le cas du transport des données. Le *soft handover* est plus souple et il est bien adapté au transport des données car il évite les pertes des paquets et par conséquent, les mécanismes de retransmission qui peuvent ralentir le transfert des données sont réduits.

Quand un mobile est en *soft handover,* il possède plusieurs connexions (*legs*) avec différentes cellules radio. Dans le sens montant, le terminal envoie les mêmes informations sur les différentes connexions. Un mécanisme de recombinaison des flux est établi dans le RNC pour obtenir un seul flux sortant. Inversement, dans le sens descendant, le flux entrant dans le RNC est divisé en plusieurs flux identiques qui sont envoyés sur les différentes connexions. Ce mécanisme est appelé la macro-diversité (*macro-diversity*). Si le terminal mobile a des connexions avec des cellules appartenant à un même RNC, la macro-diversité est établie dans ce RNC. Si le mobile est connecté à des cellules appartenant à deux RNC différents, un seul RNC (le *Serving-RNC*) garde le point d'interconnexion avec le réseau cœur. L'autre RNC joue le rôle du *Drift*-RNC. C'est le cas de la figure2.6 où la macro-diversité est établie dans le S-RNC pour former un flux unique sortant vers le réseau cœur.

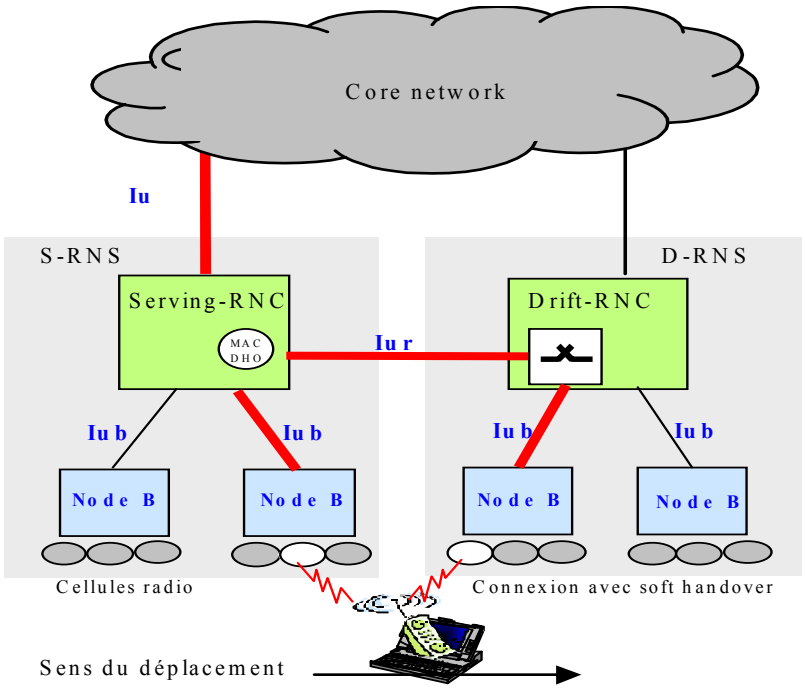


Figure 2.6 Schéma de *soft handover* et de macro-diversité

Sur le schéma de la figure 2.6, le terminal mobile est connecté à deux cellules appartenant à deux RNC différents. Le premier flux venant du terminal passe par l'interface Iub qui relie le Node B au S-

RNC. Le deuxième flux passe par une autre interface Iub qui est reliée au D-RNC, il est acheminé dans le D-RNC sur l'interface Iur reliée au S-RNC. Dans le S-RNC, un mécanisme de recombinaison entre les deux flux est établi et un seul flux est envoyé sur l'interface Iu vers le réseau cœur. Quand le terminal quitte la première cellule, il coupe sa connexion directe avec le S-RNC et il garde une seule connexion qui passe par le D-RNC vers le S-RNC. Si le mobile s'éloigne du S-RNC, le nombre de D-RNC qu'il traverse augmente et le chemin vers le réseau cœur sera plus long, alors un mécanisme de ré-localisation est mis en place. Le mécanisme de ré-localisation consiste à changer le point d'interconnexion avec le réseau cœur et par suite changer le S-RNC.

2.3.5 Les interfaces logiques dans l'UTRAN

L'architecture générique des interfaces logiques de l'UTRAN est représentée dans la figure 2.7. La structure de la pile protocolaire se divise en deux couches: la couche du réseau radio RNL (*Radio Network Layer*) et la couche du réseau de transport TNL (*Transport Network Layer*). Ces deux couches sont séparées dans le but de pouvoir modifier la couche de transport sans besoin de reconfigurer la couche radio.

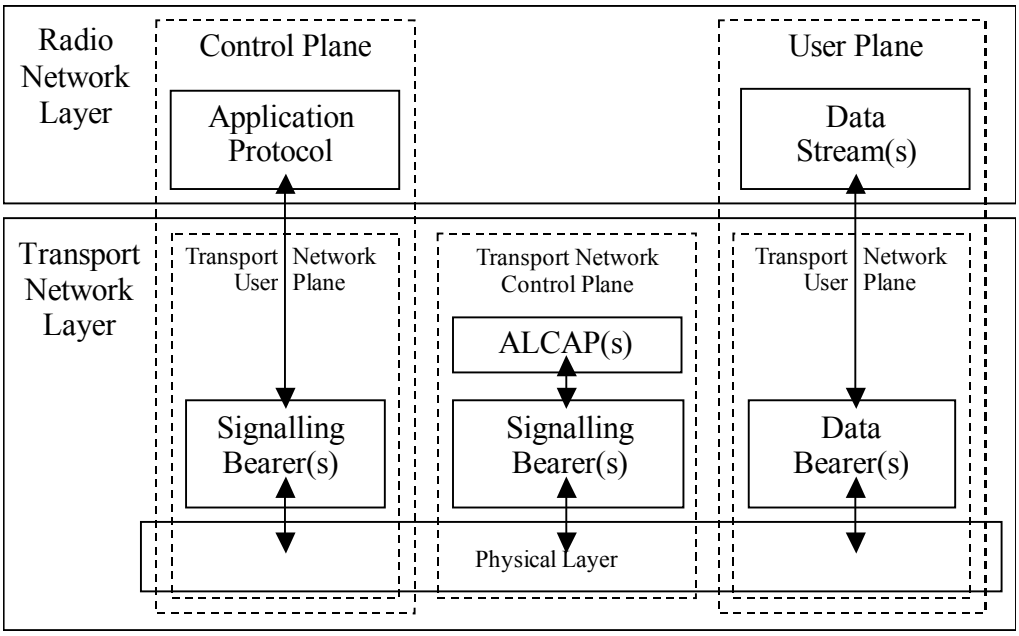


Figure 2.7 Architecture générique des interfaces de l'UTRAN

La couche réseau de transport est destinée à transporter les données de la couche radio au sein de l'UTRAN. La couche réseau radio assure la gestion des ressources de l'interface radio et les fonctions d'établissement et de libération des connexions entre le terminal mobile et le réseau UTRAN. Le plan de contrôle de la couche réseau radio gère les canaux de signalisation nécessaires pour transmettre les données des protocoles applications comme RANAP (*Radio Access Network Application Protocol*), RNSAP (*Radio Network Subsystem Application Part*) ou NBAP (*Node B Application Part*). Les protocoles du plan de contrôle de la couche réseau radio sont indépendants de la technologie employée dans le réseau de transport. Les protocoles ALCAP (*Access Link Control Application Part*) du plan de contrôle de la couche réseau de transport assurent les services de signalisation nécessaires pour l'établissement des connexions du plan utilisateur pour transporter les données de la couche réseau radio. Le plan utilisateur de la couche réseau radio contient les protocoles nécessaires pour transporter les flux des données de l'interface radio ainsi que les informations nécessaires aux mécanismes de synchronisation des trames et de macro-diversité.

2.3.5.1 L'interface Uu

L'interface logique Uu sert à connecter le terminal mobile à la station de base par l'intermédiaire d'une liaison radio. La couche physique de l'interface Uu est basée sur la technique d'accès multiple à répartition en codes CDMA (*Code Division Multiple Access*). Elle peut fonctionner en mode TDD (*Time Division Duplex*) ou en mode FDD (*Frequency Division Duplex*). On définit deux types de canaux radio : les canaux logiques comme DCCH (*Dedicated Control CHannel*), DTCH (*Dedicated Transport CHannel*), CTCH (*Common Traffic CHannel*), etc., et les canaux de transport comme DCH (*Dedicated CHannel*), RACH (*Random Access CHannel*), FACH (*Forward Access CHannel*), DSCH (*Downlink Shared CHannel*), etc. Nous ne détaillons pas le sujet de l'interface radio parce qu'il ne constitue pas le sujet principal de notre thèse, mais nous allons parler brièvement des protocoles des couches radio qui ont d'influence sur notre étude. En effet, la couche liaison de l'interface radio est divisée en deux sous-couches: la couche MAC (*Medium Access Control*) et la couche RLC (*Radio Link Control*) qu'on va présenter dans le paragraphe 2.4. La couche réseau est divisée en trois sous-couches : RRC (*Radio Resource Control*), MM (*Mobility Management*) et CC (*Call Control*). Puisque notre étude s'intéresse aux interfaces Iub et Iur uniquement, nous étudions seulement les mécanismes des couches RRC, RLC et MAC qui ont d'influence directe sur le trafic transporté sur les interfaces Iub et Iur.

2.3.5.2 L'interface Iu

C'est l'interface logique d'interconnexion entre le réseau d'accès radio et le réseau cœur. Pour que le plan utilisateur de l'interface Iu soit indépendant du domaine du réseau cœur (commutation de circuits ou commutation de paquets), deux types d'interfaces Iu ont été définis :

- L'interface Iu-CS qui connecte l'UTRAN avec le domaine à commutation de circuits (CS : *Circuit Switched domain*) du réseau cœur. La structure protocolaire de cette interface est représentée dans la figure 2.8.
- L'interface Iu-PS qui connecte l'UTRAN avec le domaine à commutation de paquets (PS : *Packet Switched domain*) du réseau cœur. La structure protocolaire de cette interface est représentée dans la figure 2.9.

Le 3GPP a choisi dans sa Release 99 le protocole AAL2/ATM comme protocole de transport sur l'interface Iu-CS et le protocole AAL5/ATM sur l'interface Iu-PS. Sur les schémas ci-dessous, nous présentons la structure protocolaire de l'interface Iu de la Release 99 puisque nous nous intéressons, dans la première partie de notre thèse, au cas de la Release 99.

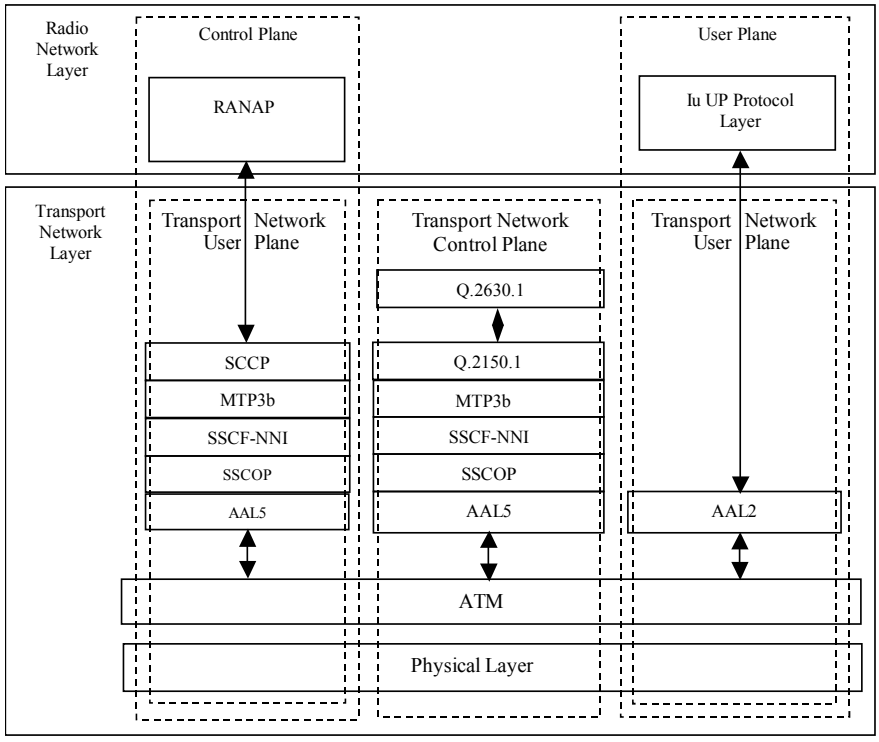


Figure 2.8 Structure protocolaire de l'interface Iu-CS

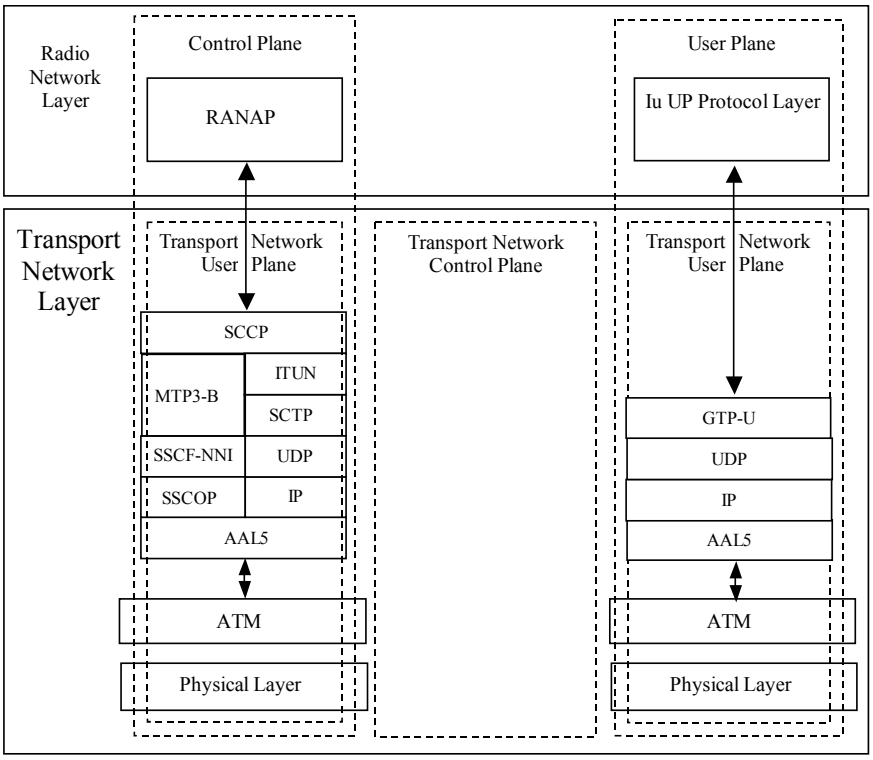


Figure 2.9 Structure protocolaire de l'interface Iu-PS

2.3.5.3 L'interface Iub

C'est l'interface logique d'interconnexion entre le Node B et le RNC. Sur cette interface, on ne distingue pas entre domaine CS et domaine PS. Tous les flux du plan utilisateur sont transportés par le

protocole AAL2/ATM choisi par le 3GPP dans sa Release 99 comme le protocole de transport sur l'interface Iub. Les flux du plan de contrôle sont acheminés sur les canaux AAL5. La figure 2.10 représente la structure protocolaire de l'interface Iub. C'est une interface UNI (*User-Network Interface*) au niveau ATM. Les canaux AAL2 servent de support de transmission aux données des protocoles du niveau supérieur (DCH-FP, RACH-FP, FACH-FP, PCH-FP, DSCH-FP, USCH-FP). Le protocole de trame (*FP: Frame Protocol*) a été conçu pour le transfert fiable des données sur un support de transmission non fiable, et cela grâce aux mécanismes de détection d'erreurs basés sur des CRC (*Cyclic Random Checksum*).

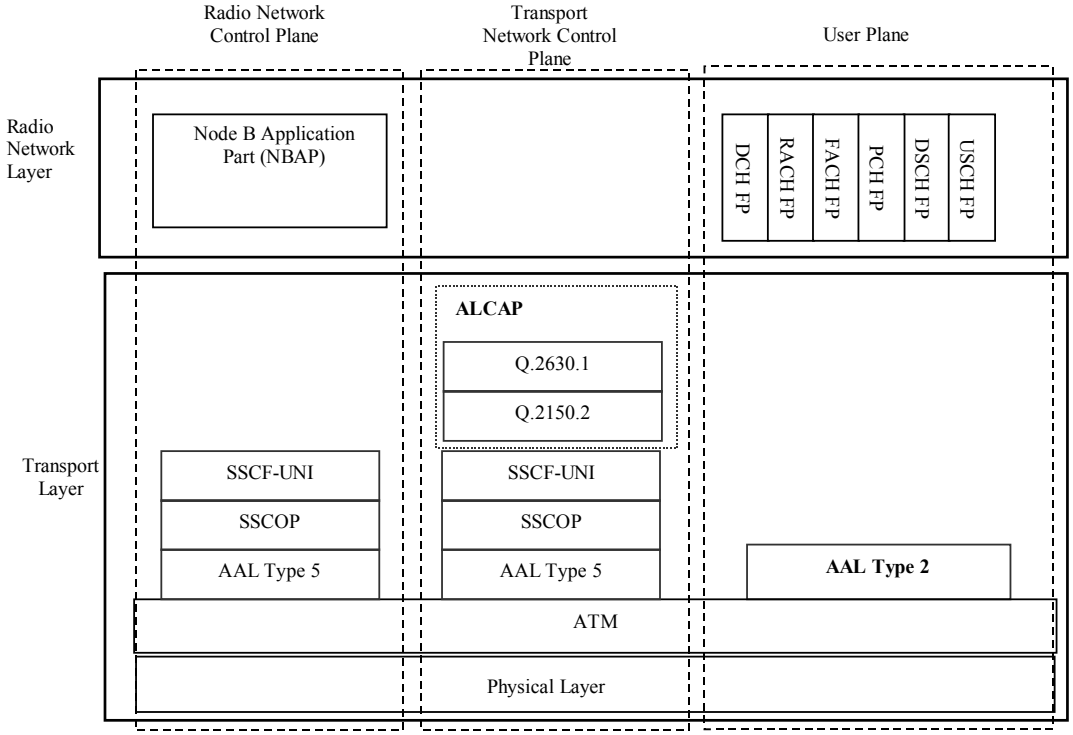


Figure 2.10 Structure protocolaire de l'interface Iub

2.3.5.4 L'interface Iur

C'est l'interface logique entre deux RNC. Dans sa Release 99, le 3GPP a choisi le protocole AAL2/ATM comme protocole de transport au niveau de la couche réseau de transport TNL pour le plan utilisateur et le protocole AAL5/ATM pour le plan de contrôle. Dans le cas où un mobile serait connecté à des cellules radio appartenant à des RNC différents, le flux passant par le D-RNC doit être acheminé à travers l'interface Iur vers le S-RNC. La figure 2.11 représente le schéma de la structure protocolaire de cette interface logique.

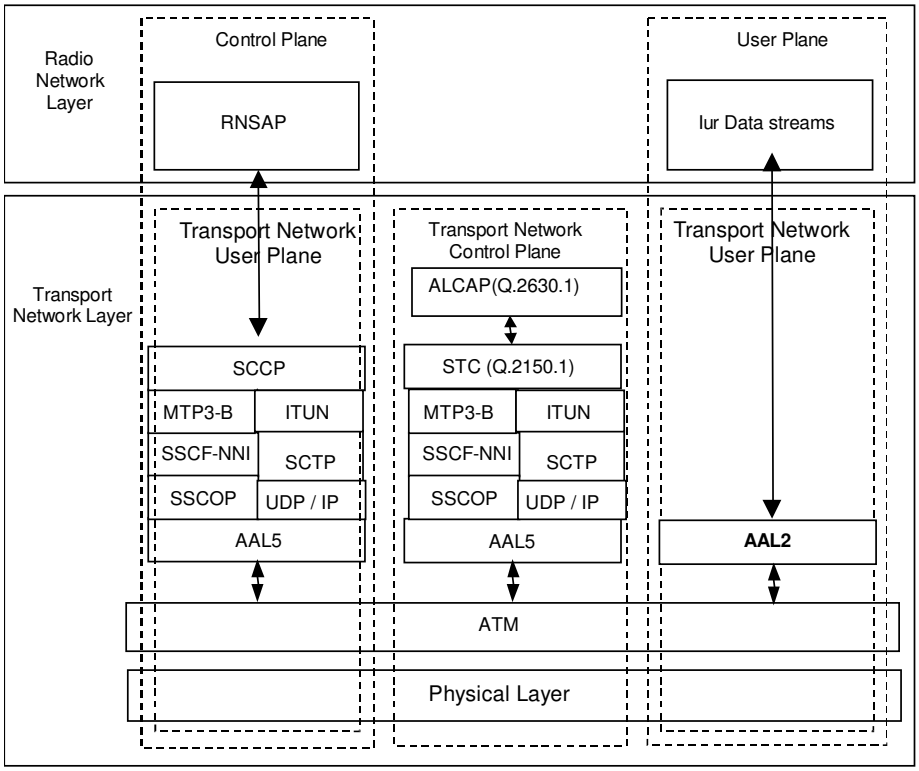


Figure 2.11 Structure protocolaire de l'interface Iur

2.4 Caractéristiques des couches protocolaires du RNL

Le sujet d'étude de notre thèse concerne les aspects de performance et de qualité de service dans le réseau de transport de l'UTRAN, le TNL. Puisque la pile protocolaire de la couche du réseau radio (le RNL) a un impact sur les flux venant des couches supérieures et entrant dans le TNL, nous avons besoin d'étudier brièvement les mécanismes des protocoles radio qui ont une influence directe sur le profil du trafic entrant dans le TNL. En fait, le modèle de trafic d'un service au niveau du TNL est différent du modèle du même service à l'entrée du RNL à cause des mécanismes de segmentation et d'ordonnancement des couches radio que nous allons présenter dans la suite. La structure protocolaire de l'interface radio du RNL est présentée sur la figure 2.12.

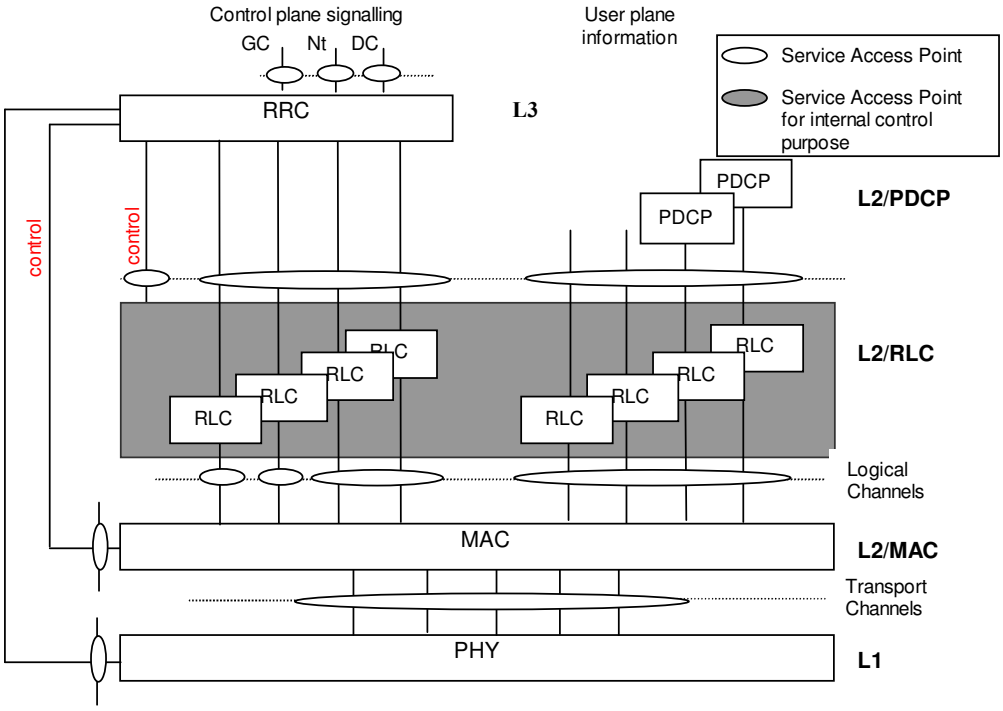


Figure 2.12 Structure protocolaire de l'interface radio

Dans la suite de ce paragraphe, nous allons présenter seulement les mécanismes des couches protocolaires qui ont une influence directe sur le profil du trafic passant par le RNL. Les protocoles concernés sont : RRC, RLC, MAC et FP.

2.4.1 La couche RRC (*Radio Resource Control*)

La couche RRC [100] a pour rôle de gérer la signalisation des connexions radio entre le mobile et l'UTRAN : établissement, libération et reconfiguration. Elle est responsable des fonctions de contrôle d'admission, de la gestion des ressources radio, du contrôle de puissance et de la gestion de la mobilité. Une seule connexion RRC est établie pour chaque mobile quel que soit le nombre des sessions et le mode (PS ou CS). Cette couche interagit avec les couches RLC et MAC pour déterminer la taille des RLC-PDU au niveau de la couche RLC [106] ainsi que le nombre de TB qui pourront être envoyés dans un même TTI au niveau de la couche MAC [105]. Dans les paragraphes suivants, nous allons présenter les mécanismes des couches RLC et MAC pour mieux comprendre les mécanismes des couches radio.

2.4.2 La couche RLC (*Radio Link Control*)

La couche RLC [106] contient des fonctions classiques du niveau 2 tel que le transfert des données sur l'interface radio. Elle réalise la fonction de segmentation des paquets en des unités de taille prédéterminée par la couche RRC. Ces unités sont appelées RLC-PDU (*RLC-Packet Data Unit*). Elle assure aussi le ré-assemblage des paquets à la réception. La couche RLC offre trois modes d'opération:

1. Le mode transparent TM (*Transparent Mode*): Dans ce mode de fonctionnement, la couche RLC réalise uniquement les opérations de segmentation et de ré-assemblage. Aucun en-tête RLC n'est ajouté aux paquets.
2. Le mode non-acquitté UM (*Unacknowledged Mode*): Dans ce mode de fonctionnement, la couche RLC réalise les mécanismes de segmentation/ré-assemblage ainsi que des mécanismes

de concaténation de plusieurs paquets (*RLC-SDU : RLC-Service data Unit*) dans un seul RLC-PDU. Un RLC-SDU correspond à un paquet du niveau supérieur. Le mode UM assure la détection d'erreurs et de pertes mais aucun mécanisme de retransmission n'est mis en place. Un en-tête est ajouté à chaque paquet. Le format d'un RLC-PDU en mode UM est présenté sur la figure 2.13.

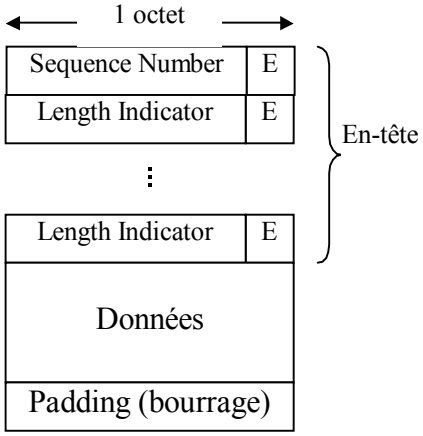


Figure 2.13 Format d'un paquet RLC-PDU en mode UM

- *Sequence Number* (SN): Ce champ de 7 bits indique le numéro de séquence du paquet RLC-PDU. Il est utilisé par l'entité de réception pour détecter les erreurs et les pertes et pour préserver l'ordre des paquets.
- *Length Indicator* (LI): Ce champ de 7 bits indique la longueur du paquet RLC-SDU, c'est à dire le champ de données suivant. Dans le cas de concaténation de plusieurs paquets, un champ LI est ajouté pour chaque RLC-SDU. Un seul champ SN est ajouté pour l'ensemble du RLC-PDU. Dans le cas où un seul RLC-SDU est inséré dans un RLC-PDU, le champ LI est éliminé.
- Le bit E : Ce bit indique la fin de chaque champ de l'en-tête.
- Le champ des données (*Data*): C'est le champ qui contient les informations de la couche supérieure.
- Le champ de bourrage (*Padding*): Ce sont des bits de bourrage qui sont ajoutés à la fin du RLC-PDU pour que sa taille soit égale à la taille imposée par la couche RRC.

3. Le mode acquitté AM (*Acknowledged M*ode): Dans ce mode de fonctionnement, la couche RLC assure les mêmes fonctions du mode UM (segmentation/ré-assemblage, concaténation, détection d'erreurs et de pertes) mais en plus, elle assure les fonctions de retransmission des paquets en cas de pertes ou d'erreurs. Ce mode de fonctionnement est recommandé pour les applications qui demandent un transfert fiable des données. Le format d'un RLC-PDU en mode AM est donné sur la figure 2.14.

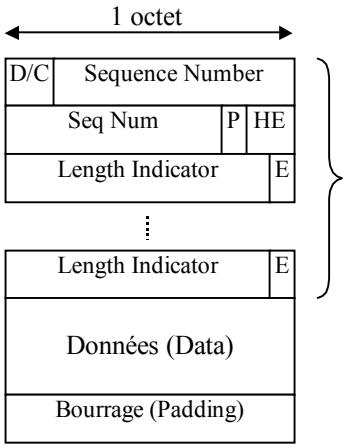


Figure 2.14 Format d'un RLC-PDU en mode AM

- Le champ D/C distingue entre les paquets de données (D) et les paquets de contrôle (C).
- Le bit P (*Polling*): ce bit est utilisé pour forcer l'entité de réception à envoyer un acquittement.
- Le champ HE de deux bits indique si l'octet suivant est un octet de données ou c'est un champ LI.
- Le champ SN est de 12 bits dans le mode AM.

2.4.3 La couche MAC (*Medium Access Control*)

La couche MAC [105] est en charge de partager les ressources radio. Ils existent plusieurs types de couches MAC tel que MAC-sh (*shared*), MAC-c (*common*) et MAC-d. La couche MAC-d gère les canaux dédiés de trafic (*DTCH : Dedicated Traffic CHannel*) et les canaux de transport dédiés (*DCH : Dedicated CHannel*). Chaque mobile contient une entité MAC-d et possède une entité MAC-d correspondante dans l'UTRAN. L'entité MAC-d transpose les canaux logiques DTCH ou DCCH *(Dedicated Common CHannel*) sur les canaux de transport DCH. Elle peut encore multiplexer plusieurs canaux DTCH ou DCCH sur un canal commun pour une cellule radio. Dans le premier cas, il n'y a pas d'en-tête MAC dans le paquet MAC-PDU (*MAC-Packet Data Unit*). Dans le deuxième cas, un en-tête est ajouté. Au niveau de la couche MAC, on définit les notions suivantes :

- TB (*Transport Block*) : c'est la plus petite entité de données au niveau du canal de transport. Un TB correspond à un MAC-PDU (*MAC-Packet Data Unit*) qui encapsule un RLC-PDU. La taille du TB est un paramètre dynamique.
- TBS (*Transport Block Set*) : c'est un ensemble de TB transmis par la couche MAC vers la couche physique dans un intervalle de temps TTI. Le nombre de TB dans un TBS est un paramètre dynamique qui dépend de la bande passante radio disponible et des exigences des services transportés.
- TTI (*Transmission Time Interval*) : c'est l'intervalle de temps entre deux TBS envoyés de la couche MAC vers la couche physique. Le TTI est un paramètre qui peut avoir des valeurs multiples de 10 millisecondes. Les valeurs typiques du TTI sont : 10, 20, 40 et 80 ms.
- TFS (*Transport Format Set*) : C'est un ensemble de TF (*Transport Format*) dont chacun définit une taille de TB et une taille de TBS. Le TFS contient toutes les valeurs possibles. Il contient aussi la valeur du TTI.

La couche MAC interagit avec la couche RRC qui lui communique la taille d'un TB, la taille d'un TBS et la valeur du TTI. Dans chaque TTI, la couche MAC envoie un nombre de TB inférieur ou égal à la taille d'un TBS. Ce mécanisme de fonctionnement rend pseudo-périodique le trafic sortant de la couche MAC.

2.4.4 La couche FP (*Frame Protocol*)

La couche FP (*Frame Protocol*) se situe dans le Node B et dans le RNC. C'est un protocole de transfert de trames sur les interfaces Iub et Iur. Ce protocole a un rôle très important dans le cas du *soft handover* et de la macro-diversité. En effet, ce protocole est responsable de la synchronisation des canaux radio et des fonctions de recombinaison dans le cas de la macro-diversité. Le schéma de la figure 2.15 représente la recombinaison de deux flux dans le RNC. Deux flux identiques venants du même terminal mobile en *soft handover* sont recombinés pour donner un seul flux sortant vers le réseau cœur. Dans l'exemple du schéma, les trames des deux flux contiennent les mêmes données. Dans chaque trame FP, plusieurs TB sont transportés. Il y a encore un champ qui donne une estimation de la qualité du lien radio (*QE : Quality Estimation*) en terme de taux d'erreurs de bit BER (*Bit Error Ratio*). En effet, il est possible que des erreurs ne soient pas corrigées un niveau physique et qu'elles soient transmises au niveau FP. Dans ce cas, il est possible que l'algorithme de macro-diversité élimine le TB1 venant du Node B 1 car son champ CRC est incorrect. Il est aussi possible de rejeter le TB2 venant du Node B 2 parce que la qualité de réception de ce flux est plus faible que celui du Node B 1 (BER plus élevé), ou même de faire une combinaison des deux trames (combinaison bit par bit).

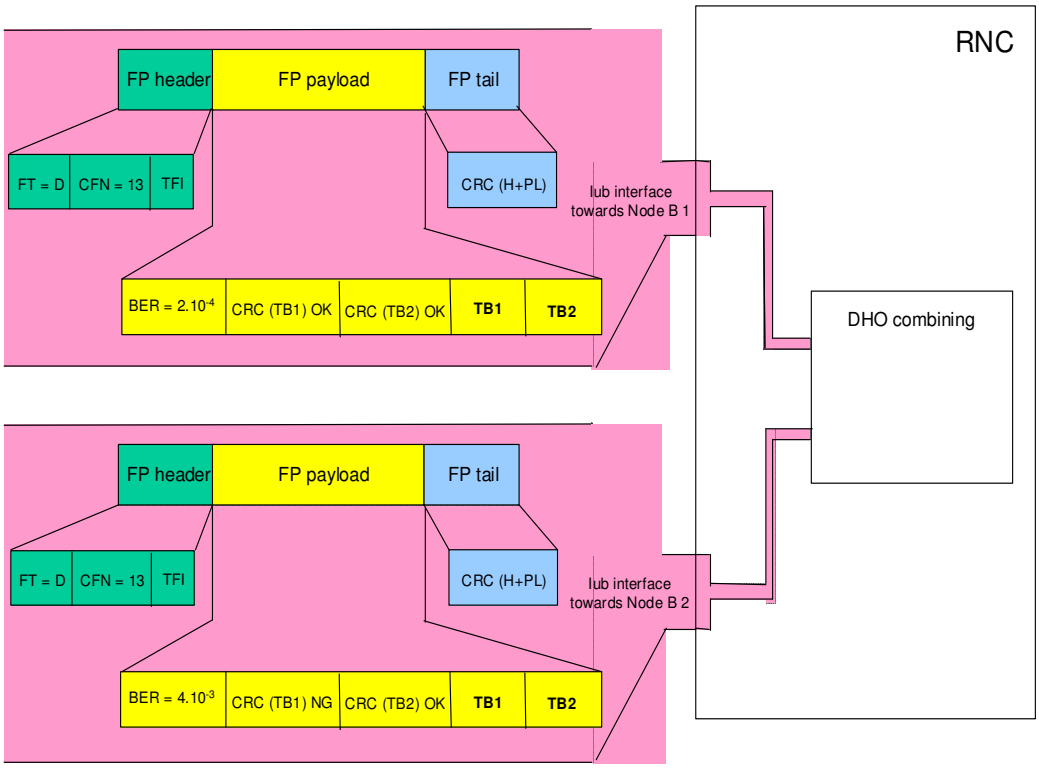


Figure 2.15 Processus de recombinaison dans le cas de la macro-diversité

Tous les TB envoyés par la couche MAC dans un même TTI et pour le même service sont assemblés au niveau FP dans une même trame appelée FP-PDU (*FP-Packet Data Unit*). Sur la figure 2.15, le CRC (H+PL) est le champ *Cyclic Redundancy Checksum* qui sert à détecter les erreurs dans l'en-tête et la *payload*. Le FT (*Frame Type*) indique si la trame est une trame de données ou de contrôle. Le CFN (*Connection Frame Number*) indique le numéro de la trame dans laquelle les

données doivent être envoyées sur le lien descendant (*downlink*) ou le numéro de la trame dans laquelle les données sont reçues dans le sens montant (*uplink*). Le champ TFI (*Transport Format Indicator*) indique le numéro local qui existe pour chaque canal DCH pour donner des informations à propos du TTI. Le champ QE (*Quality Estimation*) définit le BER du canal physique. Ce champ existe seulement dans le sens montant. Un CRC est présent pour chaque TB dans le sens montant uniquement. Il indique une erreur éventuelle dans le TB.

2.5 Structure protocolaire du TNL

Les canaux radio sont étendus entre le terminal mobile et le RNC. Ces canaux sont transportés entre le Node B et le RNC sur l'interface Iub. Au niveau de la couche réseau de transport (TNL), les canaux radio sont transportés sur les canaux de transport du TNL. Dans la Release 99 du 3GPP, le protocole AAL2/ATM a été choisi comme protocole de transport sur les interfaces Iub et Iur de l'UTRAN. La figure 2.16 représente la structure protocolaire de l'interface Iub. Les terminaisons des couche RLC et MAC se situent dans le terminal et le RNC. Dans le Node B, le protocole FP sert à transporter les canaux radio sur l'interface Iub vers le RNC en utilisant le protocole AAL2/ATM comme support de transport. Les connexions AAL2 transportent les paquets FP-PDU sur l'interface Iub. Le trafic entrant dans la couche AAL2 est affecté par les couches RLC, MAC et FP d'où la nécessité de connaître le fonctionnement des couches du RNL pour évaluer leur impact sur le trafic transporté par les canaux AAL2.

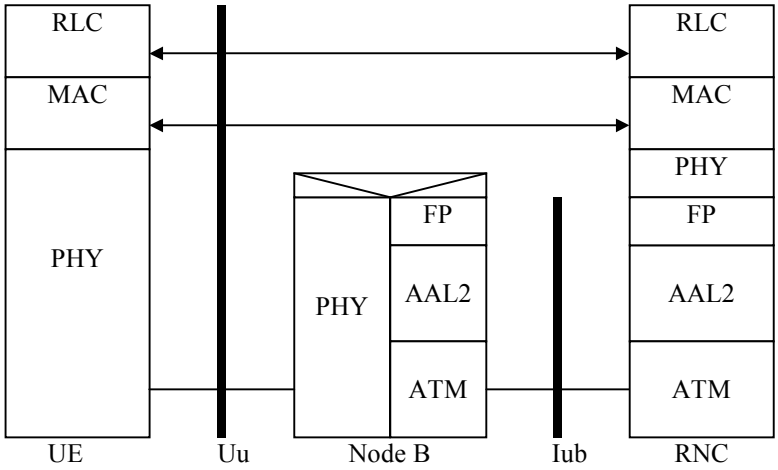


Figure 2.16 Structure protocolaire de l'interface Iub

La figure 2.17 représente la structure protocolaire des interfaces Iub et Iur dans le cas d'une connexion radio passant par un D-RNC vers un S-RNC. Dans le D-RNC, on doit monter jusqu'au niveau FP pour effectuer les opérations de macro-diversité entre différents Node B reliés au même RNC. Les terminaisons des canaux radio se situent dans le S-RNC.

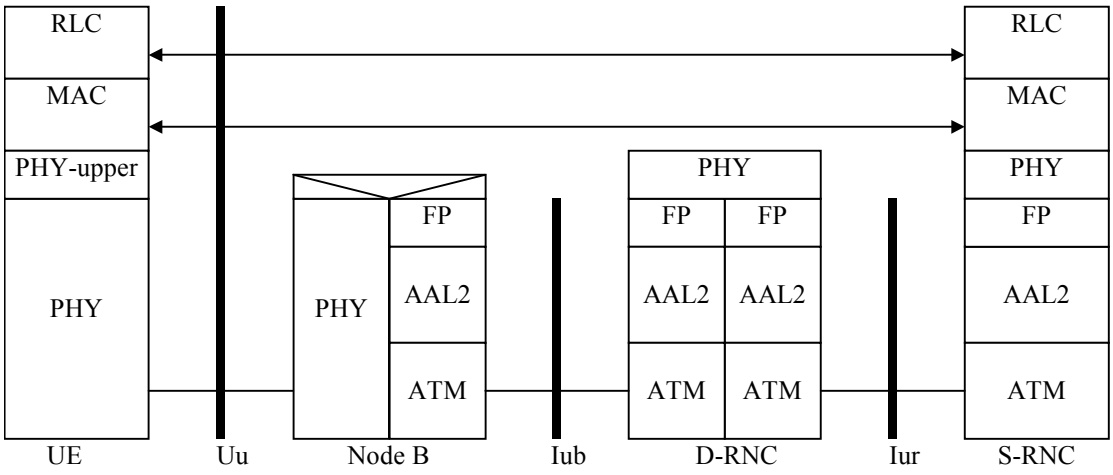


Figure 2.17 Structure protocolaire des interfaces Iub et Iur dans le cas d'un D-RNC (Release 99)

Dans une optique de déploiement des réseaux tout-IP, le 3GPP a choisi dans sa Release 5 le protocole IP comme technique de transport sur les interfaces Iub et Iur à la place de l'AAL2/ATM. Ce sujet sera détaillé plus loin dans le chapitre 7.

2.6 Les aspects des contraintes temporelles

Les spécifications du 3GPP définissent des bornes de délai pour le transfert des paquets. Le délai de bout en bout est divisé, en gros, en plusieurs parties tel que le délai de transit dans le réseau d'accès de départ, le délai de transfert dans le réseau cœur et le délai de transit dans le réseau d'accès d'arrivée. Notre étude s'intéresse au réseau d'accès dans lequel le délai de transfert peut être divisé en deux parties:

- Le délai dû aux mécanismes du réseau radio RNL dans les nœuds de l'UTRAN (Node B et RNC) tel que les traitements des couches RLC et MAC, le codage/décodage au niveau physique, la retransmission et les mécanismes liés à la macro-diversité.
- Le délai dans le réseau de transport TNL. Il est dû au temps d'attente dans les files d'attente des différentes couches protocolaires de transport, aux mécanismes de commutation dans le cas de passage par plusieurs nœuds ainsi qu'au délai de transmission sur les liens physiques du réseau terrestre.

Dans cette thèse, nous étudions les aspects temporels liés au délai de transfert dans le réseau de transport TNL. Les interfaces Iub et Iur de l'UTRAN transportent les canaux radio et doivent garantir un délai inférieur à une borne maximale pour respecter les mécanismes de synchronisation de l'interface radio [107]. En effet, quand un mobile est en *soft handover*, il possède plusieurs liens radio avec le réseau d'accès. Chaque canal radio est transporté sur une connexion de transport. Dans le sens descendant par exemple, quand le RNC décide d'envoyer un paquet vers le mobile, il envoie une copie de ce paquet sur chaque canal radio. Le mobile doit recevoir ces paquets en même temps (une petite marge est acceptée) pour qu'il puisse effectuer les mécanismes de recombinaison. C'est l'un des problèmes de la synchronisation dans l'UTRAN. Les canaux de transport doivent garantir une borne maximale du délai de transfert pour que tous les paquets arrivent dans l'intervalle de temps correspondant. En outre, même dans le cas où le mobile posséderait une seule connexion, quand la couche MAC du RNC décide d'envoyer un paquet sur l'interface Iub, ce paquet doit être envoyé sur l'interface radio dans un intervalle de temps précis, c'est à dire dans la trame radio correspondante. Le numéro de la trame radio dans laquelle le paquet doit être envoyé est noté dans l'en-tête du paquet. Si ce paquet arrive au Node B après la date d'échéance, c'est à dire au moment d'émission de la trame

radio suivante, il ne pourra pas être envoyé sur l'interface radio parce qu'il n'est pas en synchronisation avec le canal radio et par suite, il sera rejeté. Donc, le délai de transfert dans le réseau de transport doit être bien maîtrisé pour que ce réseau ne constitue pas le goulot d'étranglement (*bottleneck*) après que le flux a traversé l'interface radio qui est la ressource la plus chère. Les liens du réseau de transport doivent être bien dimensionnés pour respecter les délais exigés par le RNL. A cause des mécanismes du RNL que nous avons cité ci-dessus, tous les flux sur les interfaces Iub et Iur deviennent des flux temps-réels, même si l'application elle-même n'est pas exigeante en terme de délai de transfert.

2.7 Conclusion

Dans ce chapitre, nous avons présenté une vue globale de l'UTRAN et des mécanismes de la couche réseau radio pour évaluer leur influence sur le trafic acheminé dans le réseau de transport sur les interfaces Iub et Iur. Nous observons que les mécanismes de la couche MAC rendent pseudo-périodique le trafic entrant dans la couche AAL2. Dans le chapitre 3, nous allons présenter les modèles des sources de trafic dans l'UTRAN pour en tirer des modèles applicables au niveau de la couche de transport AAL2.

Nous avons vu dans ce chapitre que les contraintes temporelles dans l'UTRAN sont sévères à cause des mécanismes de synchronisation des couches radio. Même les flux non temps-réel comme les services *background (email, sms, etc.),* qui tolèrent en principe des délais assez élevés, deviennent très exigeants en terme de délai de transfert dès qu'ils seront transportés sur les canaux radio qui sont à leur tour supportés par les canaux de transport du TNL. L'étude des performances du protocole AAL2 en tant que protocole de transport et ses capacités à satisfaire les contraintes de l'UTRAN constitue l'objectif principal de notre thèse.

Chapitre 3

3 Modélisation des flux transportés dans l'UTRAN

3.1 Introduction

Les systèmes de mobiles de troisième génération tel que l'UMTS doivent offrir de nouveaux services en plus du service de voix déjà offert par les systèmes de deuxième génération. Grâce à la technique d'accès CDMA, les réseaux UMTS peuvent fournir des débits assez élevés pour permettre le transport des applications à haut débit tel que la vidéo et les services multimédia. Des modèles de trafic adéquats sont nécessaires pour modéliser les différents services offerts par l'UMTS. En effet, l'étude des performances des protocoles de transport dans l'UTRAN nécessite la connaissance du profil du trafic entrant dans les couches protocolaires du réseau de transport. Les flux passent, avant d'entrer dans les couches du TNL, par les couches radio (RLC, MAC) et la couche FP. Les modèles de trafic à l'entrée de la couche RLC sont différents de ceux à la sortie de la couche FP. Le schéma de la figure 3.1 représente la pile protocolaire traversée par les flux avant d'entrer dans les couches de transport.

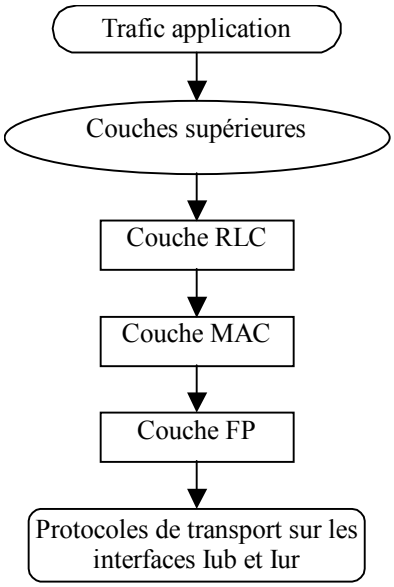


Figure 3.1 Pile protocolaire traversée par les flux dans l'UTRAN

En effet, le comportement du trafic change quand il traverse la pile protocolaire à cause des mécanismes des protocoles des couches. Dans ce chapitre, nous présentons quelques modèles de trafic des applications à l'entrée de la couche RLC ainsi qu'à la sortie de la couche FP, c'est à dire à l'entrée de la couche de transport.

3.2 Modèle de trafic de voix

Avec les nouvelles techniques de codage de voix, on peut arriver à des débits assez bas tout en conservant une bonne qualité de voix. Plusieurs codeurs existent et sont déjà normalisés. Le 3GPP a choisi un codeur à débit variable et à bande étroite appelé AMR (*Adaptative Multi-Rate*) [97, 110, 111]. Le codeur AMR change de débit en temps-réel en fonction de la qualité du canal. La longueur d'une trame de parole nécessaire au codage est de 20 ms, c'est-à-dire une trame est envoyée toutes les 20 ms. Les débits standardisés du codeur AMR ainsi que les tailles des trames sont représentés dans le tableau 3.1 .

Débit (kbit/s)	12,20	10,20	7,95	7,40	6,70	5,90	5,15	4,75
Taille de trame (bit)	244	204	159	148	134	118	103	95

Tableau 3.1 Les modes du codeur AMR

Le signal entrant dans le codeur est échantillonné et les bits de sortie sont réordonnés en trois classes (A, B et C) [97]. Les bits de chaque classe sont transportés sur un canal DCH séparé. La couche RLC est transparente par rapport aux flux AMR. Le TTI de la couche MAC est de 20 ms et par suite les bits de parole sont envoyés vers la couche FP toutes les 20 ms. La couche FP ajoute des bits de bourrage pour obtenir dans chaque paquet FP-PDU un nombre entier d'octet (*octet alignment*). Le tableau 3.2 présente le nombre de bits dans les paquets de chaque classe pour les différents modes du codeur AMR.

Nombre de bit	Classe A		Classe B		Classe C	
	trame	bourrage	trame	bourrage	trame	bourrage
Mode 12,20	81	7	103	1	60	4
Mode 10,20	65	7	99	5	40	0
Mode 7,95	75	5	84	4	-	-
Mode 7,40	61	3	87	1	-	-
Mode 6,70	55	1	79	1	-	-
Mode 5,90	55	1	63	1	-	-
Mode 5,15	49	7	54	2	-	-
Mode 4,75	39	1	56	0	-	-

Tableau 3.2 Tailles des paquets de voix AMR

Les octets des classes A, B et C sont assemblés par la couche FP en un paquet FP-PDU. Pour chaque classe, un en-tête d'un octet est ajouté pour indiquer le format de la séquence de bits. Un CRC

(*Cyclic Redundancy Check*) de 2 octets est ajouté pour protéger l'ensemble des en-têtes des classes. Un autre champ de 2 octets est ajouté à la fin de chaque FP-PDU pour protéger le contenu du paquet (*QE : Quality Estimation*). Ce champ est utilisé uniquement pour les flux dans le sens montant.

Après la fin de la période d'activité de la source de voix, le codeur AMR envoie un paquet appelé SID (*SIlence Descriptor*) pour indiquer le début d'une période de silence. Ce paquet est envoyé après 20 ms de l'envoi du dernier paquet de voix. Pendant la période de silence, des SID sont envoyés pour indiquer les changements du niveau du bruit de fond. Si ce niveau ne change pas pendant cette période, aucun SID n'est envoyé. La taille d'un SID est de 39 bits auxquels 1 bit de bourrage est ajouté au niveau de la couche FP. L'en-tête FP d'un SID est de 3 octets et le CRC est ajouté dans le sens montant uniquement.

Dans notre modèle, nous considérons que le codeur AMR fonctionne en mode 12,2 kbit/s. En effet, notre étude s'intéresse aux performances des protocoles de transport sur les interfaces Iub et Iur qui transportent les canaux radio. Le goulot d'étranglement (*bottleneck*) dans l'UTRAN est l'interface radio et par suite, si le trafic trouve les ressources nécessaires sur cette interface, il ne doit pas être bloqué par le réseau de transport. Pour cela, nous considérons que le codeur fonctionne à plein régime et que le débit de sortie est constant (12,2 kbit/s). Dans ce cas la taille d'un paquet AMR avec les *overhead* de la couche FP est de 37 octets dans le sens descendant et de 39 octets dans le sens montant. La taille du SID avec les *overhead* FP est de 8 octets dans les sens descendant et de 10 octets dans le sens montant.

Une source de voix AMR peut alors être représentée par un modèle ON/OFF où les périodes ON et OFF correspondent respectivement aux périodes d'activité et de silence de la source [87]. Ces deux périodes sont des variables aléatoires exponentiellement distribuées avec une moyenne de 3 ms. Une variable aléatoire "X" qui suit une loi exponentielle est définie comme suit:

$$f(x)=\begin{cases}\mu.e^{-\mu x}; & six\geq 0\\ 0; & sinon\end{cases} \qquad F(x)=\begin{cases}1-e^{-\mu x}; & si\ x\geq 0\\ 0; & sinon\end{cases} \qquad \textbf{Équation 3.1}$$

où f(x) et F(x) sont respectivement la densité de probabilité et la fonction de répartition de la variable "X" qui suit la loi exponentielle. μ est la valeur moyenne.

Pendant la période ON, des paquets de taille constante sont envoyés toutes les 20 ms. Dans notre modèle, nous considérons que les SID sont envoyés toutes les 160 ms pendant la période de silence. La figure 3.2 représente le modèle d'une source de voix AMR.

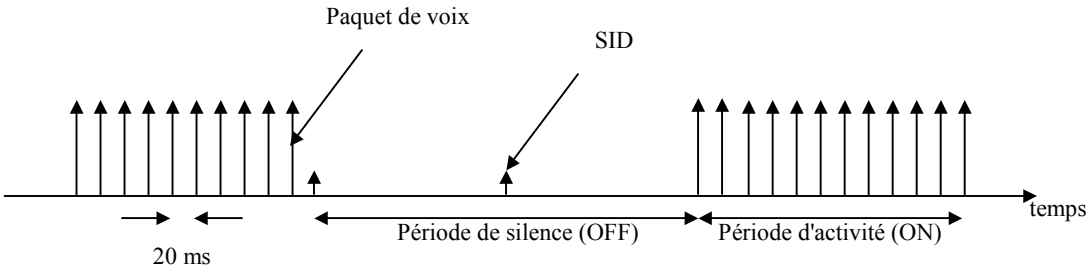


Figure 3.2 Modèle d'une source de voix AMR

La durée d'une session AMR est une variable aléatoire exponentiellement distribuée de valeur moyenne de 3 minutes [87, 92].

3.3 Modèle de trafic Web (WWW)

Une source de trafic WWW (*World Wide Web*) est représentée par un ensemble de sessions de consultation de sites web sur internet [87]. Une session WWW est constituée d'une suite d'appels de paquets (*packet-calls*). Un *packet-call* correspond au téléchargement d'un fichier HTML. La taille d'un *packet-call* est une variable aléatoire "S" qui suit une loi de Pareto bornée (*Pareto with cut-off distribution*) [87, 92]. Une variable aléatoire "X" qui suit une loi de Pareto normale est définie de la manière suivante :

$$\begin{cases} f_X(x)=\dfrac{\alpha.k^{\alpha}}{x^{\alpha+1}} \quad ; \quad \forall x \geq k \\ F_X(x)=1-\left(\dfrac{k}{x}\right)^{\alpha} \quad ; \quad \forall x \geq k \\ \mu=\dfrac{k.\alpha}{\alpha-1} \quad ; \quad \alpha>1 \\ \sigma^2=\dfrac{k^2.\alpha}{(\alpha-2)(\alpha-1)^2} \quad ; \quad \alpha>2 \end{cases}$$

Équation 3.2 Loi de Pareto normale

Où Fx et fx représentent respectivement la fonction de répartition et la densité de probabilité de la variable aléatoire qui suit la loi de Pareto normale. α est le facteur de Fano. μ et σ² représentent respectivement la moyenne et la variance de la distribution de Pareto. k représente la valeur minimale de la variable "X".

La taille d'un *packet-call* est alors définie comme suit : S = min(X,m) où "m" est la taille maximale d'un *packet-call*. La densité de probabilité de la taille d'un *packet-call* est alors donnée par la formule suivante :

$$f_S(x)=\begin{cases} \dfrac{\alpha.k^{\alpha}}{x^{\alpha+1}} \; ; \quad k \leq x < m \\ \dfrac{k^{\alpha}}{m^{\alpha}} \; ; \quad x \geq m \end{cases} \qquad \mu=\frac{\alpha.k-m\left(\frac{k}{m}\right)^{\alpha}}{\alpha-1} \; ; \; \alpha>1$$

Équation 3.3 Loi de Pareto bornée

μ est la taille moyenne d'un *packet-call*. k est la taille minimale d'un *packet-call*.

Dans notre modèle, nous considérons les valeurs des paramètres données dans le rapport technique du 3GPP TR 25.933 [92] : α = 1,1; k = 1858 octets et m = 5000 000 octets. Par un simple calcul en utilisant l'équation 3.3, on trouve que la taille moyenne d'un *packet-call* est de 12 000 octets.

Deux *packet-calls* successifs sont séparés par un intervalle de temps appelé "*reading-time*" ou "*thinking-time*". La longueur de cet intervalle est une variable aléatoire qui suit une loi exponentielle d'une valeur moyenne de 12 secondes [92]. Le nombre de *packet-calls* dans une session WWW est géométriquement distribué avec une moyenne de 5 [92]. Une loi géométrique est définie comme suit:

$$p(x)=\begin{cases} p.(1-p)^x \quad ; \quad si \; x \in \{0,1,...\} \\ 0 \quad ; \quad sinon \end{cases} ; \; F(x)=\begin{cases} 1-(1-p)^{\lfloor x \rfloor+1} \quad ; \quad six \geq 0 \\ 0 \; ; \; sinon \end{cases}$$

$$\mu=\frac{1-p}{p} \; ; \; \sigma^2=\frac{1-p}{p^2} \; ; \; \textbf{avec} \; p \in (0,1)$$

Équation 3.4 Loi géométrique

où F(x) est la fonction de distribution et p(x) est la fonction de masse. μ et σ^2 sont respectivement la moyenne et la variance. p est le paramètre de la loi géométrique qui est égale à $\frac{1}{6}$ dans notre modèle.

Le modèle WWW à la sortie de la couche FP

Le modèle présenté ci-dessus est applicable à l'entrée de la couche RLC. Ce modèle n'est pas applicable à l'entrée des couches de transport (AAL2/ATM) à cause des mécanismes de contrôle de flux de la couche RLC et de la nature pseudo-périodique des canaux de transport imposée par la valeur du TTI de la couche MAC.

En effet, la couche RLC fournit des services en mode assuré (AM) pour les trafics en mode paquets comme WWW. Ces trafics sont aussi appelés UDD (*Unconstrained Delay Data*) parce que les contraintes temporelles ne sont pas très strictes. La couche RLC assure la segmentation des paquets dont la taille dépasse la taille fixe déterminée par la couche RRC. Elle assure aussi la concaténation des petits paquets et elle ajoute des informations de bourrage (*padding*) aux RLC-PDU partiellement remplis pour les compléter et obtenir des RLC-PDU de tailles fixes. Ces RLC-PDU sont envoyés vers la couche MAC où chaque RLC-PDU est encapsulé dans un TB (*Transport Block*). La couche MAC envoie dans chaque TTI un nombre déterminé de TB de telle façon que le débit sur le support radio ne dépasse pas la valeur maximale du mode UDD utilisé. La taille d'un RLC-PDU et le nombre de TB dans chaque TTI sont alors liés au débit du canal radio. Les débits UDD définis dans les normes du 3GPP sont : 8, 32, 64, 144, 384 et 2048 kbit/s. Dans notre thèse, nous utilisons seulement les modes 64, 144 et 384 kbit/s. La valeur du TTI choisie est de 40 ms. La taille maximale d'un RLC-PDU est de 40 octets. Quand un *packet-call* arrive dans la couche RLC, il est segmenté en des RLC-PDU auxquels des en-têtes de 2 octets sont ajoutés parce qu'on est en mode AM. Chaque RLC-PDU forme un TB au niveau de la couche MAC et plusieurs TB sont envoyés dans un même TTI. Nous considérons que le canal utilise son débit maximal, c'est à dire le débit correspondant au mode UDD choisi, et par suite le nombre de TB dans un TTI est maximal. Evidemment, le dernier groupe de TB peut contenir un nombre plus petit de TB s'il ne reste pas de données dans le *buffer* RLC. Dans la couche FP, tous les TB envoyés dans un même TTI et appartenant à un même service sont regroupés dans une même trame appelée FP-PDU à laquelle un en-tête FP de 3 octets est ajouté. Un CRC de 2 octets est ajouté dans le sens montant uniquement. Les trames FP sont toutes de même taille, à l'exception de la dernière trame du *packet-call*, elles sont envoyées périodiquement tous les TTI. La figure 3.3 représente le profil du trafic WWW aux différents niveaux.

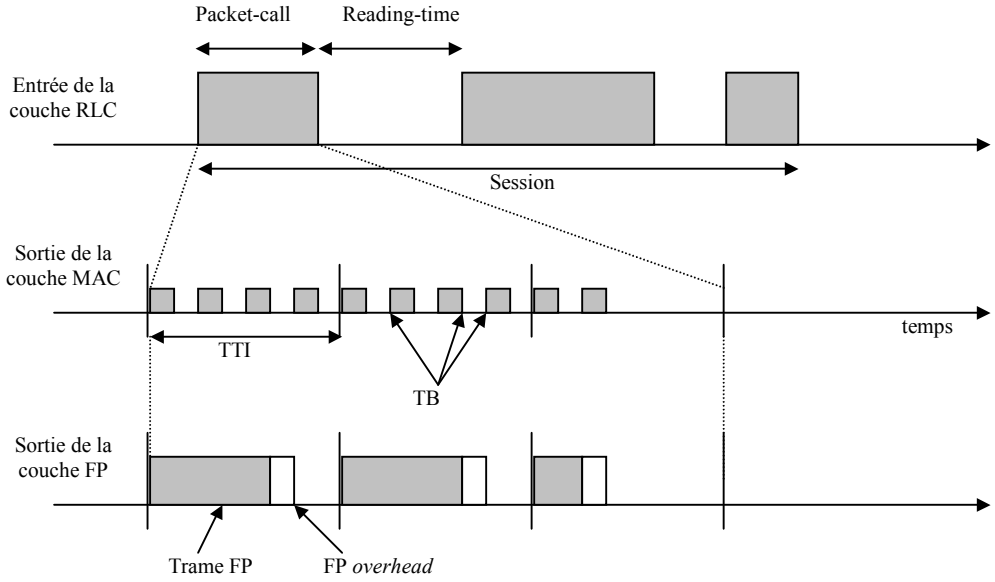


Figure 3.3 Le profil du trafic WWW (UDD)

Le trafic entrant dans les couches de transport a un comportement quasi-périodique. Le tableau 3.3 représente la taille maximale d'une trame FP-PDU pour différents modes UDD et avec tous les en-têtes.

Mode UDD	Taille d'un RLC-PDU avec l'en-tête RLC (en octets)	Nombre maximale de TB dans un TTI	Taille maximale d'une trame FP avec l'en-tête FP (en octets)	
			Sens montant	Sens descendant
UDD 64 kbit/s	42	8	341	339
UDD 144 kbit/s	42	18	761	759
UDD 384 kbit/s	42	48	2021	2019

Tableau 3.3 Taille maximale des trames FP-PDU

3.4 Modèle de trafic E-mail

Dans [29], l'auteur propose un modèle de source E-mail proche de celui d'une source WWW. Nous reprenons ce modèle mais nous modifions quelques paramètres en se basant sur des statistiques faites sur un nombre de messages électroniques. Nous supposons qu'un utilisateur envoie un certain nombre de messages électroniques pendant sa période d'activité. Une session E-mail correspond à l'envoie d'un groupe de message par un utilisateur. Le nombre de messages (E-mail) envoyés dans une session est une variable aléatoire géométriquement distribuée de moyenne 3. L'intervalle de temps entre deux messages successifs correspond au temps nécessaire à l'utilisateur pour rédiger son message. Cet intervalle est appelé "*writing-time*" et est une variable aléatoire exponentiellement distribuée de moyenne 90 secondes [29]. D'après une observation des tailles des E-mail sur une machine de laboratoire, nous avons remarqué que la fonction de répartition qui correspond mieux pour caractériser la taille d'un message électronique est la loi de Pareto bornée. La taille minimale d'un E-mail est de 500 octets (k=500). Une grande partie des messages électroniques ont une taille aux alentours de 1 koctet. Le sommet de la courbe de densité de probabilité doit alors être proche de cette valeur. En plus, la densité de probabilité des messages ayant une taille plus grande que 1 koctet ne doit pas être négligeable et par suite, la courbe de densité de probabilité ne doit pas présenter un sommet très élevé. Le facteur de Fano α agit sur ce sommet d'où le choix d'une valeur empirique de α =1,01. En pratique, la taille des E-mail est limitée par le serveur de messagerie, donc nous avons considéré que m=4000 000 octets. Cette valeur est fournie par l'opérateur FT R&D. D'après ces valeurs, nous trouvons que la taille moyenne d'un message électronique est de 4797,5 octets. Ce modèle est applicable à l'entrée de la couche RLC. Comme nous l'avons expliqué dans le paragraphe 3.3, le profil du trafic subit une déformation quand il traverse les couches RLC, MAC et FP. Le comportement du trafic à la sortie de la couche FP est quasi-périodique dont la période est le TTI du canal radio. Dans chaque TTI, un FP-PDU est envoyé dont la taille correspond au débit maximal autorisé par le mode UDD correspondant. Pour ce type de trafic, nous considérons que la taille d'un RLC-PDU est de 40 octets et que la valeur du TTI est de 40 ms. La couche RLC fonctionne en mode acquitté AM (Annexe B). Le débit maximal du support radio utilisé pour le transport du trafic E-mail est de 16 kbit/s (Annexe B) et par suite, le nombre maximal de TB dans un TTI est de 2. La taille d'un FP-PDU est alors de 87 dans le sens descendant et de 89 dans le sens montant. La figure 3.4 représente le profil du trafic E-mail à des différents niveaux.

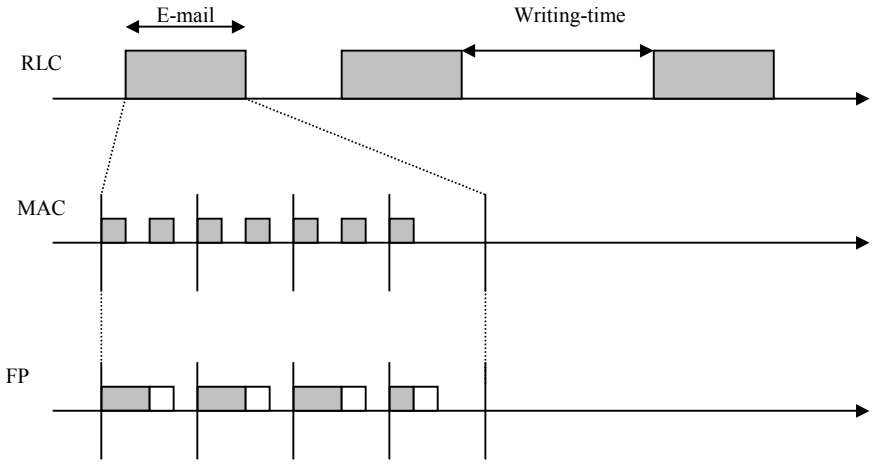


Figure 3.4 Profil du trafic E-mail

3.5 Modèle de trafic FTP

Une session FTP (*File Transfer Protocol*) est considérée dans [29] comme un transfert d'un seul fichier. La taille de ce fichier est une variable aléatoire qui suit une distribution de Pareto normale. Ce choix est inspiré du modèle de *web browsing* (WWW). La densité de probabilité et la fonction de répartition sont données par:

$$\begin{cases} f(x)=\dfrac{\alpha.k^{\alpha}}{x^{\alpha+1}};\forall x\geq k \\ F(x)=1-\left(\dfrac{k}{x}\right)^{\alpha};\forall x\geq k \\ \mu=\dfrac{k.\alpha}{\alpha-1};\alpha>1 \end{cases}$$

Équation 3.5 Loi de Pareto de la taille d'un fichier FTP

La taille minimale d'un fichier FTP est considérée comme k = 3000 octets. Le facteur de Fano est choisi de manière que la courbe de densité de probabilité ne présente pas un sommet très élevé. En effet, il y a une grande partie des fichiers FTP ayant une taille assez élevée (plus grande que 3000 octets). D'où le choix de α=1.06. La distribution de Pareto n'est pas bornée ce qui permet d'avoir des fichiers de taille très élevée. μ est la taille moyenne d'un fichier qui est égale à 53000 octets. Comme nous avons déjà expliqué dans le paragraphe 3.3, le trafic subit des déformations à cause des mécanismes des couches RLC, MAC et FP et il est caractérisé par un comportement quasi-périodique à la sortie de la couche FP. Le trafic FTP est transporté sur des canaux radio de 16 kbit/s (Annexe B). Le TTI est de 40 ms. La taille d'un RLC-PDU est de 40 octets (Annexe B). La couche RLC fonctionne en mode acquitté AM. Le nombre maximal de TB dans un TTI est de 2. La taille maximale d'une trame FP est alors de 87 octets dans le sens descendant et de 89 dans le sens montant. Nous considérons que le canal radio est utilisé à débit maximal, c'est à dire le nombre de TB envoyés dans un TTI est maximal sauf la dernière trame FP du fichier qui peut être d'une taille plus petite. Le modèle à la sortie de la couche FP est alors quasi-périodique avec une période de 40 ms.

3.6 Modèle de trafic SMS

Le trafic des messages courts SMS (*Short Message Service*) devient de plus en plus important dans les réseaux mobiles. Même si le volume d'un SMS est très faible, mais le nombre important des SMS risque d'occuper une partie non négligeable du trafic dans l'UTRAN. Nous n'avons pas trouvé un modèle d'une source SMS parce que ce type de sources a été négligé à cause du faible trafic qu'elles génèrent. Nous avons alors considéré qu'une session SMS est composée d'un certain nombre de messages. Ce nombre suit une loi géométrique de moyenne 2. En effet, le choix de la distribution géométrique est inspiré du modèle de *web browsing*. La valeur moyenne du nombre de SMS par session est choisie en tenant compte de l'activité pratique d'un utilisateur. La taille d'un message SMS est une variable aléatoire qui suit une loi uniforme de valeur minimale de 30 octets et de valeur maximale de 200 octets. En effet, la densité de probabilité de la taille d'un SMS n'est pas concentrée autour d'une valeur donnée parce qu'en pratique, un SMS peut être constitué de quelques mots (quelques dizaines d'octets) ou d'une longue phrase de taille limitée selon l'opérateur (en général 164 caractères). La taille d'un SMS peut prendre des valeurs différentes avec la même probabilité d'où le choix d'une loi uniforme. Les valeurs maximales et minimales ont été fournies par FT R&D. Une loi uniforme est définie comme suit:

$$\begin{cases} f(x)=\dfrac{1}{b-a};a<b \\ \mu=\dfrac{a+b}{2} \end{cases}$$

Équation 3.6 Loi uniforme

μ est la valeur moyenne. Elle est égale à 115 octets. a et b sont respectivement la valeur minimale et la valeur maximale de la variable aléatoire (dans notre cas [a;b]=[30;200]). L'intervalle de temps entre deux SMS successifs correspond au temps nécessaire pour la rédaction d'un message. Cet intervalle suit une loi exponentielle de moyenne 120 secondes. Ce choix est inspiré du modèle des sources E-mail.

La valeur du TTI pour le trafic SMS est de 40 ms. La taille d'un RLC-PDU est de 40 octets. La couche RLC fonctionne en mode AM pour le trafic SMS. Le débit du support radio est de 16 kbit/s (Annexe B). Le nombre maximal de TB dans un TTI est alors de 2. La taille maximale d'une trame FP est alors de 87 octets dans le sens descendant et de 89 octets dans le sens montant. Le trafic à la sortie de la couche FP est alors quasi-périodique dont la période est de 40 ms.

3.7 Modèle de trafic vidéo

Le système UMTS fournit des services audio-visuels telle que la visio-phonie et la vidéo *streaming*. Ce dernier service représente la diffusion des scènes vidéo [1]. Plusieurs modèles de trafic pour les sources vidéo sont analysés. Dans [40] par exemple, les auteurs proposent un modèle Markovien pour modéliser une source de trafic vidéo à débit variable. Un autre modèle est aussi proposé dans [54].

Or, les codeurs vidéo recommandés dans la spécification TS 26.234 [112] sont H.263 et MPEG-4. Des modèles de source MPEG sont proposés dans [1] et [33]. Dans cette thèse, nous supposons que le service vidéo utilisé est la vidéo *streaming* et que le codeur H.263 est utilisé pour encoder le signal vidéo. Un modèle de source H.263 est proposé dans [41]. Or, pour que le trafic généré dans nos simulations soit le plus proche possible de la réalité, le laboratoire de France Télécom R&D a accepté de nous fournir plusieurs traces de trafic à la sortie d'un codeur H.263. Le débit de sortie du codeur H.263 est de l'ordre de 130 kbit/s. Nous avons alors utilisé ces traces pour les appliquer à l'entrée des couches RLC, MAC et FP du simulateur de réseau que nous avons développé. La valeur du TTI est de 40 ms. La taille d'un RLC-PDU est de 40 octets. Le trafic vidéo est transporté sur des canaux en mode

circuit dont le débit du support radio est de 128 kbit/s. la couche RLC fonctionne en mode non acquitté UM parce que le trafic vidéo est un trafic temps-réel qui tolère les pertes de paquets et par suite, on n'a pas besoin des mécanismes de retransmission du mode AM de la couche RLC. Dans un TTI, 16 TB au maximum peuvent être envoyés (Annexe B). La taille maximale d'une trame FP est alors de 659 octets dans le sens descendant et de 661 dans le sens montant. Après observation du trafic à la sortie de la couche FP de notre simulateur, nous avons constaté que le débit est quasi-constant et que la taille des trames FP est maximale. Le trafic qu'on a obtenu à la sortie de la couche FP est alors réaliste et est représenté sur la figure 3.5.

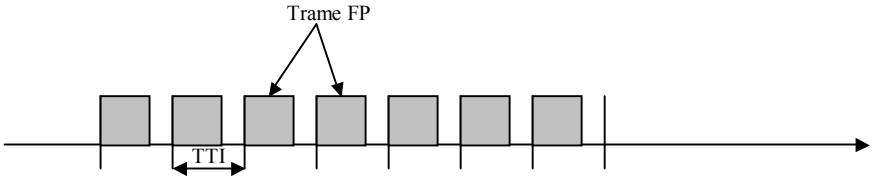


Figure 3.5 Profile du trafic vidéo

Ceci montre que le trafic est périodique avec une période de 40 ms. La différence avec les trafics WWW, FTP, E-mail et SMS est l'absence des périodes d'inactivité dans le cas du trafic vidéo. Le débit est constant pendant toute la durée de la communication.

3.8 Conclusion

Dans ce chapitre, nous avons présenté les modèles de trafic des principaux types de services de l'UMTS. Ces modèles comportent le trafic classique de voix, le trafic des données ainsi que le trafic multimédia telle que la vidéo. Ces modèles seront essentiels pour l'étude des performances des protocoles de transport du réseau TNL. Dans la suite de notre thèse, ces modèles seront implémentés dans un simulateur de réseau pour générer des flux semblables aux flux transportés dans l'UTRAN. Dans le cas de l'AAL2 (chapitre 6), seulement les services de voix et de *web browsing* WWW sont utilisés. Dans le cas de l'IP (chapitre 7), tous les modèles des services définis dans ce chapitre sont utilisés dans les simulations.

Chapitre 4

4 Le protocole AAL2/ATM et la QoS

4.1 Généralités sur les réseaux ATM

L'ATM (*Asynchronous Transfer Mode*) [70] est un mode de transfert qui fait appel à des techniques de multiplexage asynchrone par répartition dans le temps. Les flux d'informations sont structurés dans des petits blocs de taille fixe appelés cellules. L'UIT a défini un modèle de référence pour les réseaux ATM qui est représenté sur la figure 4.1.

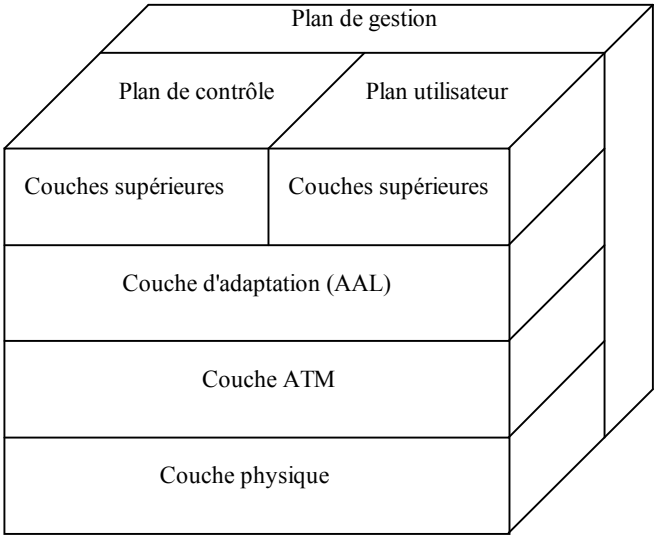


Figure 4.1 Modèle de référence de l'UIT pour les réseaux ATM

La couche physique transporte les données binaires sur le support physique. La couche ATM permet le transfert des cellules. La couche d'adaptation AAL (*ATM Adaptation Layer*) assure des fonctions des niveaux supérieures ainsi que la connexion avec des réseaux non-ATM. Quatre types de couches d'adaptation sont définis : AAL1, AAL2, AAL3/4 et AAL5. La couche AAL se divise en deux sous-couches : la sous-couche de convergence (*CS : Convergence Sublayer*) et la sous-couche de segmentation et de réassemblage (*SAR : Segmentation And Reassembly*). Le rôle essentiel de la sous-couche SAR est de segmenter les données des couches supérieures en des blocs de tailles correspondantes à la taille des cellules. A la réception, la sous-couche SAR rassemble les cellules pour restituer les données aux couches supérieures.

Une cellule ATM a une taille fixe de 53 octets dont un en-tête de 5 octets. Le champ d'information est alors de 48 octets. Le format de l'en-tête de la cellule ATM est présenté sur la figure 4.2.

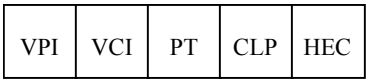


Figure 4.2 Format de l'en-tête d'une cellule ATM

Le champ VPI (*Virtual Path Identifier*) représente l'identificateur de conduit virtuel VP (*Virtual Path*). Le champ VCI (*Virtual Channel Identifier*) identifie le numéro du circuit virtuel VC (*Virtual Channel*). Le champ PT (*Payload Type*) indique si le champ d'information de la cellule est de type utilisateur ou signalisation. Le champ CLP (*Cell Loss Priority*) (1 bit) indique le niveau de priorité de la cellule en cas de congestion. Le champ HEC (*Header Error Control*) sert à détecter et à corriger les erreurs dans l'en-tête de la cellule.

Les réseaux ATM utilisent le mode connecté pour la transmission des cellules. Un circuit virtuel est ouvert avant le début de transfert des cellules. Plusieurs notions sont définies:

- Le circuit virtuel VC (*Virtual Channel*) : c'est une liaison qui transporte les cellules entre deux nœuds ATM. Le nœud qui contient la terminaison du circuit virtuel est appelé commutateur ATM.
- La connexion de circuit virtuel VCC (*Virtual Channel Connection*) : c'est la concaténation d'un ou de plusieurs VC.
- Le conduit virtuel VP (*Virtual Path*) : c'est un faisceau de VC. Tous les VC d'un même VP ont les mêmes nœuds d'extrémité. Le nœud qui contient la terminaison d'un VP est appelé brasseur.
- La connexion de conduit virtuel VPC (*Virtual Path Connection*) : elle est composée de la concaténation d'un ou de plusieurs VP.

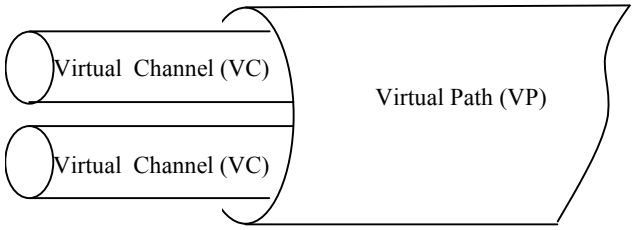


Figure 4.3 Structure de VP et VC

La couche ATM définit des paramètres de performance comme le délai de transfert des cellules (*CTD : Cell Transfer Delay*), la variation du temps de transfert des cellules (*CDV : Cell Delay Variation*), le taux de perte des cellules (*CLR : Cell Loss Ratio*), etc. [61]. Lors de l'établissement d'une connexion, un utilisateur peut négocier avec le réseau les bornes supérieures d'un ou de plusieurs paramètres de performance et le réseau garantira ces valeurs durant la vie de la connexion.

Plusieurs classes de services sont définies au niveau de la couche ATM. Une classe est définie comme un ensemble de valeurs prédéterminées des paramètres de performance. Une classe peut définir la valeur maximale autorisée d'un ou de plusieurs paramètres de performance et par suite, lors de l'établissement d'une connexion, l'utilisateur négocie avec le réseau la classe de service demandée sans préciser explicitement les valeurs des paramètres de performance. Les classes définies pour le moment sont quatre [61]:

- La classe 1 appelée aussi classe sévère (*Stringent Class*) qui définit des bornes supérieures pour le délai de transfert, la variation du délai et le taux de perte.

- La classe 2 appelée aussi classe tolérante (*Tolerant Class*) qui définit la valeur maximale du taux de perte et ne définit aucune valeur de délai ou de variation de délai.
- La classe 3 appelée aussi classe bi-niveaux (*Bi-level Class*) définie uniquement la valeur maximale du taux de perte des cellules de priorité élevée.
- La classe U (*Unspecified*) (non spécifié) ne définie aucune valeur des paramètres de performance.

Lors de la procédure d'établissement d'une connexion, l'utilisateur et le réseau négocient un contrat de trafic qui est constitué de la classe de service demandée et du descripteur de trafic. Le descripteur de trafic est un ensemble de paramètres de trafic qui caractérisent le flux des données tel que le débit crête des cellules (*PCR : Peak Cell Rate*) de l'émetteur, le débit moyen soutenu (*SCR : Sustainable Cell Rate*), la tolérance de la variation du délai (*CDV Tolerance*), la taille maximale d'une rafale de cellules (*IBT : Intrinsic Burst Tolerance*), etc. Après la négociation du contrat de trafic, le réseau garantit les valeurs des paramètres de performance à condition que l'utilisateur respecte le contrat de trafic. Si l'utilisateur viole le contrat de trafic, on dit que le trafic est non-conforme et par suite, les cellules non-conformes sont marquées et seront rejetées en premier si une congestion aura lieu dans un nœud du réseau.

Les standards de l'ATM définissent des capacités de transfert en mode ATM (*ATC : ATM Transfer Capability*) qui assurent le support des applications avec la qualité de service correspondante. Un ATC peut être associé à un VC ou un VCC. Une capacité ATC est vue par l'utilisateur comme un moyen pour garantir l'engagement de qualité de service et du contrat de trafic négocié. Elle est vue par le réseau comme un moyen d'offrir un gain statistique de multiplexage. Une ATC est définie par des paramètres comme le débit cellulaire crête, le débit soutenu, la taille maximale d'une rafale de cellules (*Burst*), etc. Si le trafic est conforme au contrat de trafic négocié, l'ATC choisi garantira les valeurs des paramètres de QoS négociées et sinon, la qualité de service du niveau ATM est garantie seulement pour le volume des cellules conformes aux tests de conformité. Les ATC définies par l'UIT-T sont : DBR, SBR, ABT/DT, ABT/IT et ABR. Le DBR est l'ATC par défaut.

- **L'ATC DBR (*Deterministic Bit Rate*):** Cette capacité de transfert définit un débit cellulaire crête PCR (*Peak Cell Rate*) et assure la garantie de ce débit durant toute la vie de la connexion. Ce type de capacité de transfert est une émulation d'un circuit virtuel. Elle est en général utilisée pour les applications temps-réel.
- **L'ATC SBR (*Statistical Bit Rate*):** Un débit moyen appelé SCR (*Sustainable Cell Rate*) est garanti durant la vie de la connexion. Durant une rafale de cellules de taille IBT (*Intrinsic Burst Tolrance*), le débit crête PCR est garanti.
- **L'ATC ABT (*ATM Block Transfer*):** Ce service fonctionne en mode de bloc. Quand un bloc de cellules est envoyé, le réseau lui offre un service de type DBR et il libère les ressources après l'émission du bloc entier. Ce service fournit une qualité de service comparable au service DBR mais il est plus souple et flexible. Il existe deux types de service ABT : Le premier est l'ABT-DT (*ABT – with Delayed Transmission)* dans lequel le réseau assure la réservation des ressources nécessaires avant d'envoyer le bloc de cellules. Le deuxième est l'ABT-IT (*ABT – with Immediate Transmission*) dans lequel le réseau ne fait pas de réservation des ressources et compte sur la capacité statistique du réseau pour supporter ce bloc. Ce deuxième mode est utilisé dans le cas de transport des flux temps-réel ou dans le cas d'un bloc de cellules de petite taille où le temps nécessaire pour son émission serait faible par rapport au temps nécessaire pour la procédure de réservation des ressources.
- **L'ATC ABR (*Available Bit Rate*):** Ce mode utilise la bande passante restante mais il garantit quand même un débit minimum pour assurer le transfert des cellules dans un temps acceptable.

L'ATM Forum a défini de sa part des ATC qui sont plus ou moins semblables aux ATC définis par l'UIT. Ces ATC sont le CBR (*Constant Bit Rate*) qui est semblable au DBR. Le VBR (*Variable Bit Rate*) qui est semblable au SBR. L'ABR qui est le même que celui de l'UIT. Le GFR (*Guaranteed Frame Rate*) dans lequel le contrôle des paquets s'effectue sur la base de la trame (si une cellule est perdue, toutes les autres cellules appartenant à la même trame seront éliminées) et un débit minimal est garanti pour assurer un taux de perte faible. Enfin, l'ATC UBR (*Unspecified Bit Rate*) correspond au service *Best Effort (BE)* où il n'y a aucune garantie de qualité de service. La figure 4.4 représente une illustration de la répartition de la bande passante entre les différentes ATC.

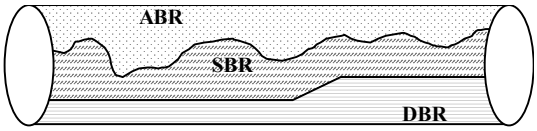


Figure 4.4 Répartition de la bande passante entre les ATC

Les ressources sont allouées aux classes DBR (ou CBR) et SBR (ou VBR) pour assurer une garantie totale de la qualité de service et le reste de la bande passante disponible sera récupéré par le service ABR.

4.2 La couche d'adaptation AAL2

L'idée d'une nouvelle couche d'adaptation AAL remonte au milieu des années 90 quand s'est posée la question du transport en ATM des applications à très bas débit, à contraintes de temps-réel et à débit variable, dont la parole compressée constitue le meilleur exemple. Les AAL existantes s'y prêtaient assez mal : l'AAL1 est spécifiée pour transporter les débits fixes alors que l'AAL5 est inadaptée au transport du temps-réel.

La définition de l'AAL2 (*ATM Adaptation Layer - type 2*) [63, 64] a été très rapide (environ deux ans) en regard des durées qui sont habituellement celles de l'UIT. Cette rapidité a une explication : l'AAL2 vise non seulement les applications mobiles, mais aussi les applications de *trunking*, deux créneaux porteurs pour l'ATM. Contrairement aux autres AAL, l'AAL2 a été définie par les constructeurs (et non par les opérateurs), avec une double paternité : les constructeurs orientés "mobiles" dont Ericsson est le chef de file, et les constructeurs orientés "*trunking*", pratiquement tous américains.

L'AAL2 a été définie pour contourner le problème du temps d'assemblage d'une cellule ATM qui devient critique pour les bas débits (à 16 kbit/s, sa valeur est de 24 ms). La solution retenue est simple : en multiplexant les flux de plusieurs communications dans un même canal ATM, le délai reste raisonnable pour une communication donnée.

Le multiplexage fait appel à une structuration des informations en mini-cellules : les blocs d'informations provenant d'une source donnée constituent la *payload* d'une mini-cellule. Les mini-cellules des différentes sources sont ensuite rangées consécutivement dans les cellules ATM. Le transport des paquets d'informations dans des mini-cellules constitue la partie basse du protocole nommée CPS (*Common Part Sublayer*).

A vrai dire, ce mode de transport en mini-cellules s'apparente plus à une nouvelle couche ATM-bis, superposée à la couche ATM elle-même, qu'à une véritable couche d'adaptation. En pratique, il s'agit aussi de s'affranchir de la longueur fixe de la cellule ATM en lui superposant une structure de transport à longueur variable jugée mieux appropriée aux informations à transporter.

4.2.1 Structuration de la couche AAL2

La couche AAL2 se divise en deux grandes sous-couches (figure 4.5) :

- **La sous-couche SSCS:** qui assure les fonctions d'adaptation indispensables.
- **La sous-couche CPS:** qui a pour rôle d'assurer les fonctions de multiplexage.

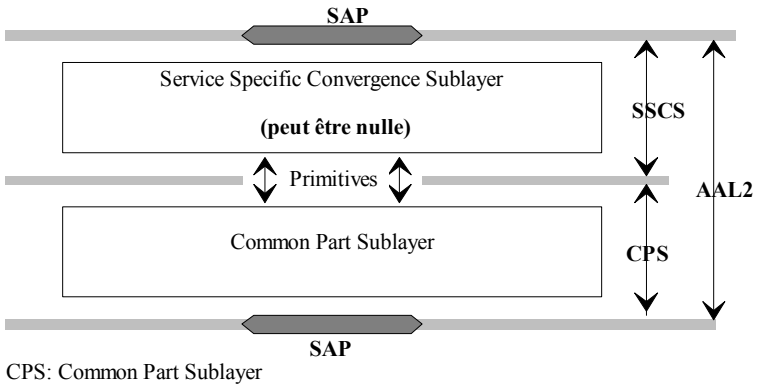


Figure 4.5 Structuration de la couche AAL2

Les deux sous-couches CPS et SSCS s'échangent les informations à travers des primitives. Par contre, la sous-couche SSCS interagit avec les couches supérieures à travers des points d'accès au service SAP (*Service Access Point*). La CPS interagit à son tour avec les couches inférieures à travers un SAP.

4.2.2 La sous-couche SSCS

Cette sous-couche SSCS de segmentation est appelée "*Segmentation And Reassembly Service Specific Convergence Sublayer*". Le service minimum offert par cette sous-couche est la fonction de segmentation et de réassemblage des unités de données des couches supérieures. Dès la définition de la nouvelle norme AAL2, il est apparu qu'il serait nécessaire de définir une SSCS de segmentation pour le transport des messages dont la longueur dépasse la longueur maximale permise pour les paquets de la couche CPS, les mini-cellules. Initialement, cette SSCS de segmentation n'était pas considérée comme un élément essentiel de l'AAL2 destinée avant tout au transport des paquets relativement courts et à contrainte de temps-réel. C'est pourtant celle qui a été définie la première [62]. La raison en est que cette SSCS positionnait l'AAL2 sur un créneau déjà couvert par l'AAL5 : la volonté a donc été très marquée de la rendre la plus compatible possible avec l'AAL5 avec laquelle on se doutait qu'elle aurait à interfonctionner. La normalisation a ainsi repris au maximum ce qui avait été défini pour l'AAL5.

La découpe fonctionnelle de cette SSCS (figure 4.6) fait déjà apparaître de fortes similitudes avec l'AAL5. Elle comprend en effet trois sous-couches empilées au-dessus de la CPS.

On notera que chaque sous-couche possède un point d'accès SAP à sa partie supérieure, ce qui permet d'accéder à la CPS via n'importe quelle sous-couche. Les trois sous-couches de la SSCS de segmentation ont pour rôle de fournir aux applications un transport de messages pouvant aller jusqu'au mode assuré.

La sous-couche de base SS-SAR (*Service Specific - Segmentation And Reassembly*) a une simple fonction de segmentation et de réassemblage des messages. Son rôle est de découper les messages (dont la longueur peut atteindre 65536 octets) en blocs de 45 octets, le dernier bloc du message ayant une longueur variable inférieure à 45 octets. La taille maximale d'un segment peut être de 64 octets si

cette valeur est indiquée lors de la procédure d'établissement de la connexion. Si aucune valeur n'est indiquée, la valeur par défaut et de 45 octets. Le processus de réassemblage en réception est symétrique.

La sous-couche intermédiaire SS-TED (*Service Specific - Transmission Error Detection*) a pour rôle de détecter toutes les erreurs qui peuvent affecter les messages : erreurs bit, cellules perdues. Ses fonctions sont en fait très proches de celles de la CPCS de l'AAL5, avec laquelle la compatibilité était recherchée. La SS-TED utilisera donc pour réaliser ses fonctions un *trailer* de 8 octets similaire à celle de l'AAL5. Seuls les messages corrects sont transmis à la sous-couche supérieure. En cas d'erreur détectée, la SS-TED écarte le message et transmet à la couche supérieure une indication d'erreur.

La sous-couche SS-ADT (*Service Specific - Assured Data Transfer*) réalise le mode assuré. Dans le cas où une primitive de détection d'erreurs aurait été transmise par SS-TED, un protocole de retransmission de messages est mis en œuvre.

La sous-couche SS-SAR est indispensable parce qu'elle assure les fonctions de base de la couche SSCS. Les sous-couches SS-TED et SS-ADT sont optionnelles. Sur les interfaces Iub et Iur de l'UTRAN, seulement la sous-couche SS-SAR est utilisée et par suite, le mode de transfert est non assuré.

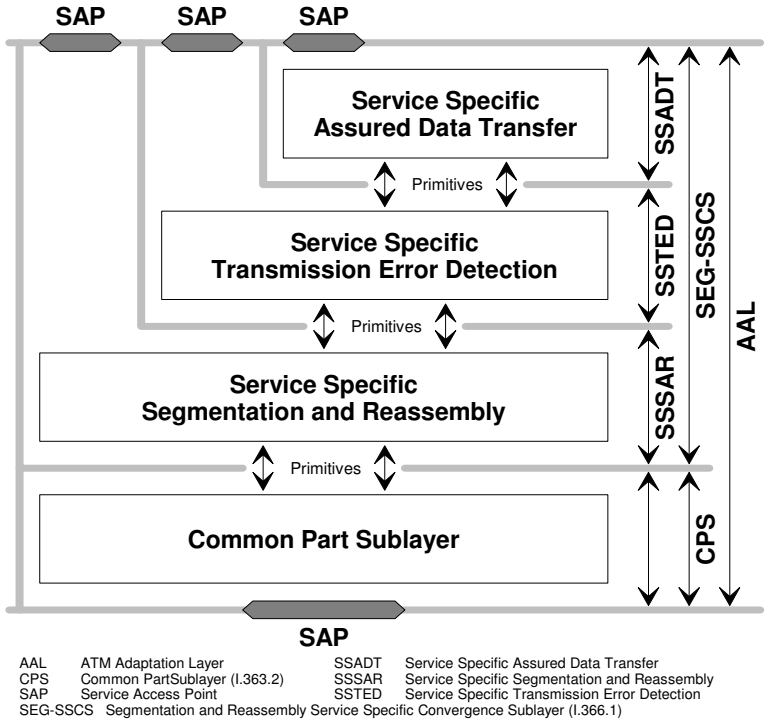


Figure 4.6 Structuration de la SSCS

4.2.3 La sous-couche CPS

La sous-couche de partie commune CPS (*Common Part Sublayer*) a pour rôle de multiplexer plusieurs canaux AAL2 correspondant à des communications différentes en rangeant leurs mini-cellules dans les *payloads* des cellules ATM d'un VPI/VCI donné. Ces canaux AAL2 peuvent transporter différents types de signaux, chaque canal AAL2 choisissant la SSCS adéquate. La figure 4.7 représente les différents niveaux de multiplexage : les canaux AAL2 sont multiplexés dans les VC qui sont à leur tour multiplexés dans les VP.

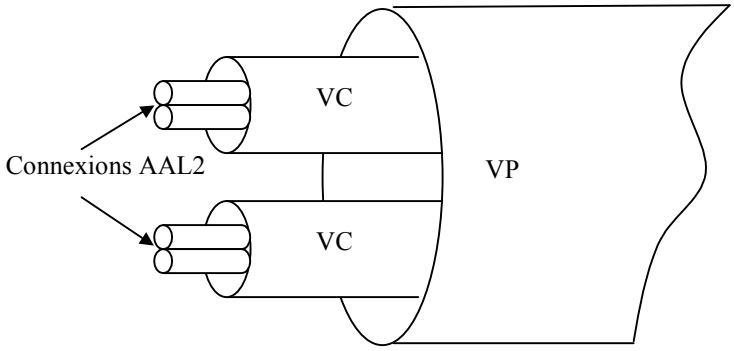


Figure 4.7 Les différents niveaux de multiplexage

Les entités des données transportées par la CPS sont nommées CPS-SDU ou mini-cellules. Elles ont un en-tête de trois octets et une *payload* de longueur variable qui contient l'information usager. Par défaut, cette longueur est au maximum de 45 octets. Elle peut cependant avoir la valeur de 64 octets dans le cas où les mini-cellules ne seraient pas commutées (il faut alors le préciser par signalisation). La figure 4.8 donne le format de la mini-cellule AAL2.

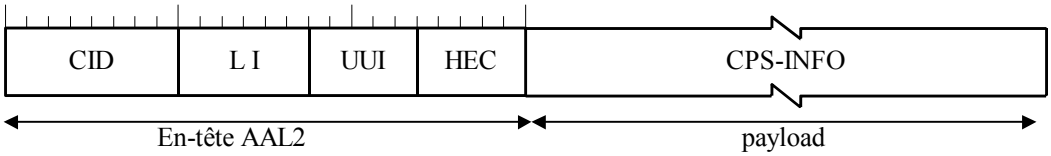


Figure 4.8 Format d'une mini-cellule AAL2

Les trois octets d'en-tête comportent les champs suivants :

- Le champ CID (*Channel IDentifier*) contient sur 8 bits le numéro du canal AAL2. Si l'on tient compte des valeurs réservées, une CPS pourra transporter jusqu'à 248 canaux AAL2, c'est à dire 248 communications différentes, sur un VPI/VCI.
- Le champ LI (*Length Indicator*) code sur 6 bits la longueur de la mini-cellule courante. La longueur maximale par défaut est de 45 octets, elle est extensible à 64 octets si la signalisation le demande.
- Le champ UUI (*User to User Indication*) de 5 bits, comporte 32 *codepoints* qui servent à transporter de bout en bout des informations relatives à la sous-couche supérieure (la SSCS).
- Le champ HEC (*Header Error Control*) de 5 bits, calqué sur celui de l'en-tête ATM sert à protéger l'en-tête de la mini-cellule.

La séquence des mini-cellules est garantie sur chaque canal AAL2, mais le service fourni par la CPS est de type non-assuré, c'est à dire que les mini-cellules manquantes (par suite de pertes de cellules ATM) ne sont pas remplacées par retransmission à ce niveau.

Le protocole au niveau CPS définit un véritable mode de transport. Les mini-cellules sont insérées dans les cellules ATM dont le premier octet contient systématiquement un pointeur de repérage. Les mini-cellules ayant une longueur quelconque, leur séquence n'est en général pas cadrée dans les cellules ATM. Pour améliorer le remplissage, l'*overlapping* a été retenu : une mini-cellule dont le début est rangé en fin de cellule n peut déborder sur la cellule n+1 (figure 4.9). Dans le cas où il n'y aurait plus de données à transmettre, la cellule ATM est remplie par des octets de bourrage (*Padding*).

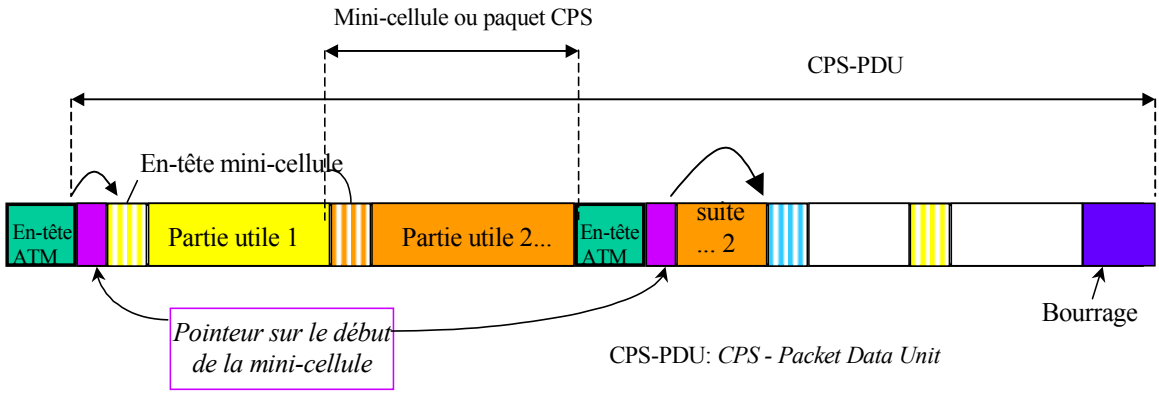


Figure 4.9 Principe de l'*Overlapping* et du *Padding* (bourrage)

La figure 4.10 représente le format d'une cellule ATM transportant des mini-cellules AAL2. Le STF (*STart Field*) est un champ d'un octet inséré au début de la *payload* de chaque cellule ATM juste après l'en-tête ATM. C'est l'en-tête de la CPS-PDU et il est composé de trois champs (figure 4.10).

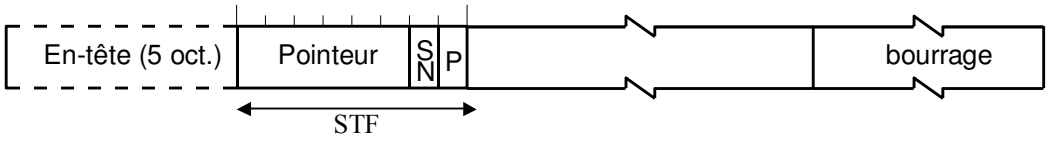


Figure 4.10 Structure des cellules ATM transportant des mini-cellules AAL2

Le pointeur sert à repérer le début de la première mini-cellule dans la *payload*. Son rôle est essentiel en cas de perte de cellule ATM : dans le cas général, les mini-cellules ne sont pas cadrées en début de *payload* ATM à cause de l'*overlapping*. Le pointeur permet de se recadrer dans chaque cellule ATM en donnant le nombre d'octets compris entre l'octet de pointage et le premier début de mini-cellule (si aucun début de mini-cellule n'est présent dans le champ d'information, la valeur 47 sera donnée au pointeur). Six bits sont suffisants pour le pointeur, les deux bits restants sont utilisés pour un numéro de séquence (*SN : Sequence Number*) servant à la détection des cellules perdues, et un bit de parité (*P : Parity*) pour détecter les erreurs dans le champ STF.

Rappelons les conventions de nomination des unités de données de la couche AAL2:

- **AAL2-SDU (*AAL2-Service Data Unit*):** C'est l'unité des données qui entre dans la couche AAL2, c'est à dire dans la sous-couche SSCS. Elle a une taille variable qui peut atteindre 65536 octets. Il est possible que la sous-couche SSCS ajoute un en-tête à cette unité de données pour former une unité appelée SSCS-PDU (le champ d'en-tête peut être nul). Cette dernière est ensuite segmentée en des petites entités (les CPS-SDU) avant d'être envoyée vers la sous-couche CPS.
- **CPS-SDU (*CPS-Service Data Unit*):** C'est la charge utile d'une mini-cellule. Elle a une taille variable limitée à 45 octets (ou 64 octets). Dans notre thèse, nous utilisons la valeur par défaut (45 octets).
- **Mini-cellule ou paquet CPS:** C'est l'unité des données échangée entre la sous-couche SSCS et la sous-couche CPS. Elle comporte l'en-tête AAL2 (ou l'en-tête CPS) de 3 octets suivi de la charge utile (les octets de données) "CPS-SDU".
- **CPS-PDU (*CPS-Packet Data Unit*):** C'est l'unité des données envoyée de la sous-couche CPS vers la couche ATM. Elle a une taille fixe de 48 octets pour qu'elle puisse remplir le champ d'information de la cellule ATM. Une CPS-PDU est formée d'un en-tête d'un octet qui est le champ STF (*STart Field*) et d'une charge utile de taille fixe (47 octets). La charge utile contient

une ou plusieurs mini-cellules complètes et/ou une partie d'une mini-cellule et/ou des octets de bourrage. Au niveau ATM, la CPS-PDU sera appelée ATM-SDU. On lui ajoute l'en-tête ATM pour former la cellule ATM.

Afin de limiter le délai dans certains cas, une cellule ATM peut être transmise même si elle n'est pas totalement remplie, elle est complétée dans ce cas par du bourrage après une temporisation nommée "Timer-CU" (*Timer – Combined Use*).

Le mécanisme d'insertion des mini-cellules AAL2 dans les cellules ATM est implémenté dans l'émetteur CPS et il est défini dans la recommandation ITU-T I.363.2. C'est un automate à quatre états représenté sur la figure 4.11.

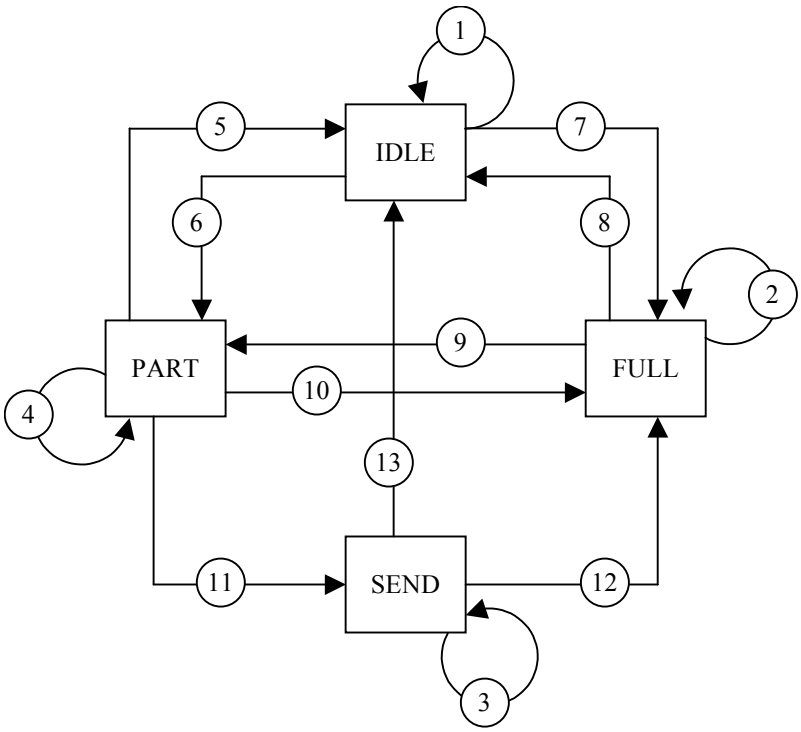


Figure 4.11 Automate de l'émetteur CPS

Si l'émetteur est dans l'état IDLE, il attend l'arrivée d'un nouveau paquet. Si un paquet arrive et ne remplit pas la CPS-PDU, l'émetteur passe à l'état PART et le Timer-CU est déclenché (Transition 6). Si la CPS-PDU est pleine et l'émetteur a déjà reçu l'autorisation d'émettre du plan de gestion des couches, il soumet la CPS-PDU à la couche ATM et reste à l'état IDLE (transition 1). Dans le cas où l'autorisation d'émettre n'est pas encore reçue, l'émetteur passe à l'état FULL (transition 7).

Dans l'état PART, si un nouveau paquet arrive et ne remplit pas la CPS-PDU, l'émetteur reste à l'état PART (transition 4). Sinon, l'automate passe à l'état FULL (transition 10).

Dans l'état PART si le Timer-CU expire, l'émetteur passe à l'état SEND s'il n'a pas encore reçu l'autorisation d'émettre (transition 11). Sinon, il soumet la CPS-PDU à la couche ATM et passe à l'état IDLE (transition 5).

Dans l'état SEND, si une nouvelle mini-cellule arrive dans la couche CPS, elle est insérée dans la CPS-PDU en cours si le mode de fonctionnement est sans blocage et l'automate passe à l'état FULL si la CPS-PDU est pleine (transition 12) ou reste dans l'état SEND dans le cas contraire (transition 3). Si on est dans le mode de blocage, c'est à dire, on bloque l'insertion des nouveaux paquets si l'émetteur est dans l'état SEND, alors l'automate reste dans le même état en attendant l'autorisation d'émettre.

Dans l'état SEND, lorsque l'émetteur reçoit l'autorisation d'émettre du plan de gestion des couches, il soumet la CPS-PDU à la couche ATM et passe à l'état IDLE (transition 13).

Dans l'état FULL, l'émetteur ne traite que les primitives venant du plan de gestion des couches. Si un nouveau paquet est arrivé, il ne sera pas traité qu'après la sortie de l'état FULL.

Si l'automate est dans l'état FULL et il reçoit l'autorisation d'émettre du plan de gestion des couches, il soumet le CPS-PDU en cours à la couche ATM et passe à l'état IDLE si le dernier paquet n'a pas chevauché sur la CPS-PDU suivante (transition 8). Sinon, et si la CPS-PDU suivante n'est pas pleine, il passe à l'état PART (transition 9). Si elle est pleine, il reste dans l'état FULL (transition 2).

Le récepteur CPS joue le rôle inverse de celui de l'émetteur. Il examine le STF, extrait les mini-cellules à partir des cellules ATM, regarde les en-têtes AAL2, puis envoie les informations des mini-cellules vers la sous-couche SS-SAR pour faire le ré-assemblage des paquets qui seront envoyés vers les couches supérieures. Si un paquet CPS chevauche sur deux cellules, le récepteur CPS s'occupe de la procédure de ré-assemblage de la mini-cellule.

4.3 La qualité de service au niveau AAL2

La figure 4.12 représente un schéma de correspondance entre les points d'accès au service (SAP) de la couche AAL2 et les SAP de la couche ATM. Dans la recommandation ITU-T I.363.2 [64], on signale que l'utilisateur peut choisir un SAP au niveau AAL2 (AAL-SAP) avec la qualité de service (QoS) associée pour transporter ses données. Pour fournir à l'utilisateur une certaine qualité de service, la couche AAL2 peut utiliser les services fournis par la couche ATM sous-jacente.

La nécessité de la définition de plusieurs classes de service au niveau AAL2 vient du fait que tous les flux sont multiplexés au niveau de la sous-couche CPS. Par conséquent, on ne peut pas différencier entre les différentes classes au niveau de la couche ATM. Une solution consiste à agréger les flux de chaque classe dans un VC séparé des autres classes. Dans ce cas, il faut établir un VC par classe et un multiplexeur CPS par VC. Dans le cas où la différenciation des services serait faite au niveau de la couche AAL2, on peut agréger des connexions AAL2 de différentes classes de services dans un même VC ATM. Les différentes classes AAL2 seront différenciées au niveau de la sous-couche CPS et pourront être toutes transportées par une même classe ATM.

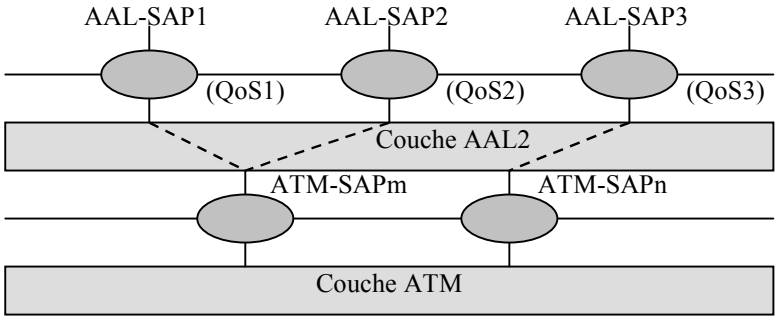


Figure 4.12 Relation entre les SAP de l'AAL2 et les SAP de l'ATM

Des études sont menées au sein de l'UIT-T pour l'introduction de la QoS au niveau de la couche AAL2 mais aucune spécification n'était encore achevée pendant la période de notre étude. Nous avons alors proposé un modèle complet pour la définition de la qualité de service au niveau de la couche AAL2 avec des paramètres de QoS, des capacités de transfert et des classes de service. Le réseau doit alors garantir la qualité de service négociée à condition que le trafic respecte l'algorithme de conformité du trafic. Cet algorithme est basé sur un profil de trafic négocié entre l'utilisateur et le

réseau lors de l'établissement de la connexion AAL2. Le profil du trafic est un ensemble de paramètres qui caractérisent le comportement du trafic. Dans la suite de ce chapitre, nous allons présenter notre contribution pour la définition de la qualité de service au niveau de la couche AAL2. Cette proposition est inspirée des travaux faits dans le cadre de la définition de la QoS au niveau ATM.

Nous présentons dans la figure 4.13 le modèle fonctionnel d'un émetteur AAL2 où nous réalisons la différenciation des services au niveau de la sous-couche CPS. En effet, un utilisateur AAL2 accède à la couche AAL2 à travers un point de terminaison de connexion AAL2-CEP (AAL2-*Connexion End Point*). Un groupe de AAL2-CEP forme un point d'accès au service AAL2-SAP. Tous les utilisateurs qui utilisent le même SAP appartiennent à la même classe de service. Une sous-couche SSCS correspond à chaque utilisateur qui sera identifié au niveau de la sous-couche CPS par son identificateur CID (*Channel IDentifier*). Au niveau de la sous-couche de multiplexage CPS, les unités sont les mini-cellules de tailles variables qui seront stockées dans des files d'attente avant d'être insérées dans les unités CPS-PDU. Les mécanismes de différenciation des services sont mis en place à l'aide d'un mécanisme d'ordonnancement qui sélectionne la file d'attente qui doit être servie. Le délai d'attente maximale tolérable dans chaque file d'attente dépend de la classe de service de cette dernière. Ce délai constitue la composante la plus importante du délai de transfert.

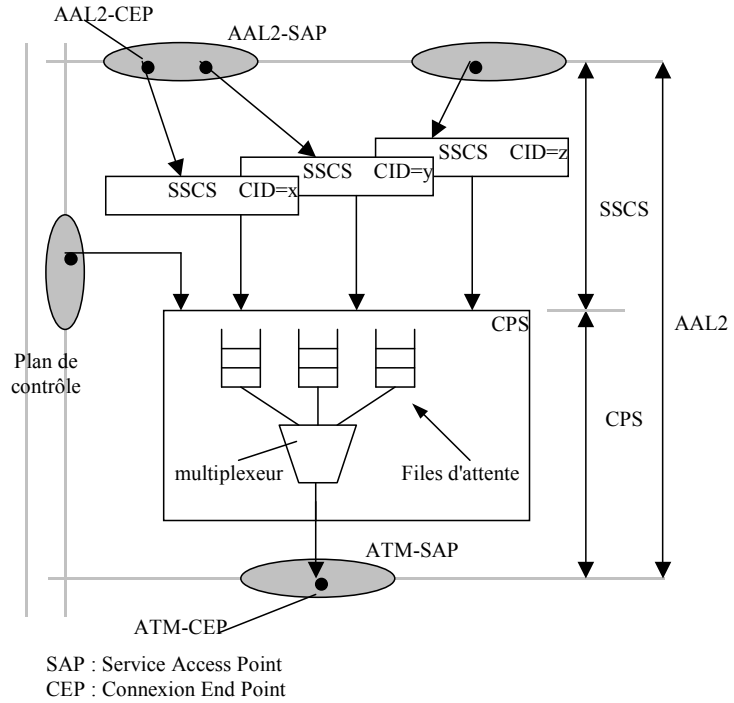


Figure 4.13 Modèle fonctionnel de l'émetteur AAL2 avec différenciation des services au niveau CPS

4.3.1 Paramètres de qualité de service au niveau AAL2

Pour pouvoir juger le bon fonctionnement d'une connexion AAL2, il faut définir des critères de performance qui sont traduits par des paramètres de QoS et qui sont aussi utilisés pour définir des classes de services au niveau AAL2. Le réseau implémente des mécanismes pour mesurer les valeurs actuelles de ces paramètres à travers l'observation du trafic écoulé. Lors de l'établissement d'une nouvelle connexion AAL2, le réseau communique à l'utilisateur les bornes maximales des paramètres de performance. Si l'utilisateur accepte ces valeurs, le réseau doit alors garantir ces bornes durant toute la vie de la connexion. Dans la normalisation de l'AAL2, aucune spécification des paramètres de la qualité de service n'est définie. Dans ce paragraphe, nous allons définir les paramètres de qualité de service que nous jugeons indispensable pour garantir une certaine qualité de service au niveau de la couche AAL2.

Pour cela, nous distinguons deux niveaux pour la définition des paramètres de qualité de service. En effet, les unités des données qui entrent dans la couche AAL2 sont tout d'abord segmentées dans la sous-couche SSCS en mini-cellules qui seront à leur tour transportées au niveau de la sous-couche CPS. Puisque la couche CPS constitue un véritable protocole de transport, grâce au champs CID qui assure la séparation entre les différentes connexions, la définition des paramètres de performance à ce niveau est nécessaire pour évaluer le fonctionnement du transport des mini-cellules (paquets CPS). Des paramètres de performance pour le transport des mini-cellules sont alors définis.

D'ailleurs, les deux sous-couches SSCS et CPS forment la couche AAL2 et interagissent ensemble pour assurer le transport des unités des données du niveau supérieur, les AAL2-SDU. Les mécanismes de segmentation et de réassemblage de la sous-couche SSCS influent sur le délai de transfert des paquets AAL2-SDU. Des critères de performance à ce niveau sont alors définis pour évaluer le fonctionnement de la couche AAL2 toute entière. Notons que ces paramètres n'ont pas de sens qu'entre les nœuds du réseau qui contiennent les terminaisons des sous-couches SSCS, c'est à dire entre les SAP de la couche AAL2.

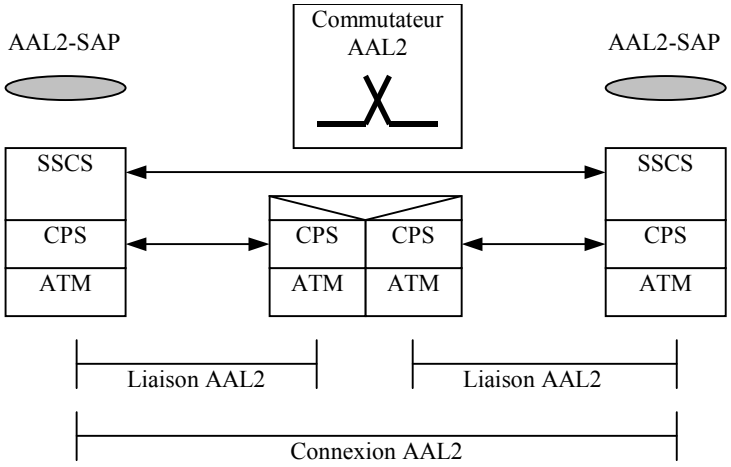


Figure 4.14 Les liaisons et les connexions AAL2

La figure 4.14 représente les terminaisons des sous-couches CPS et SSCS. Les terminaisons d'une liaison AAL2 (*AAL2 link*) sont les sous-couches CPS. Une liaison AAL2 peut être commutée dans un commutateur AAL2 où on remonte jusqu'à la sous-couche CPS et les mini-cellules sont commutées en utilisant leurs identificateurs CID. Dans ce cas, on n'a pas besoin de remonter jusqu'à la sous-couche SSCS pour reconstruire les AAL2-SDU. Une connexion AAL2 (*AAL2 connection*) est une concaténation d'une ou de plusieurs liaisons AAL2. Elle relie deux points d'accès AAL2 (*AAL2-SAP*). Les extrémités d'une connexion AAL2 contiennent les terminaisons des sous-couches SSCS.

Pour garantir une certaine qualité de service pour une connexion AAL2, nous utilisons une approche point-à-point dans laquelle nous définissons des critères de QoS sur une liaison AAL2. Les critères de QoS d'une connexion AAL2 sont alors le résultat de la concaténation des différents critères de QoS des liaisons qui constituent cette connexion.

4.3.1.1 Paramètres de QoS au niveau CPS

Nous définissons ces paramètres pour évaluer le bon fonctionnement d'une liaison AAL2. Ils concernent les unités des données de la sous-couche CPS (les mini-cellules).

4.3.1.1.1 Le délai de transfert d'une mini-cellule (MTD : Minicell Transfer Delay)

Nous le définissons comme la différence entre l'instant d'entrée d'une mini-cellule (paquet CPS) dans la sous-couche CPS du coté émetteur et l'instant de sortie de la sous-couche CPS du coté récepteur. Ce délai inclut le délai d'attente dans la file d'attente du multiplexeur CPS, le délai d'assemblage des unités CPS-PDU, le délai de transmission des cellules ATM et le délai de réassemblage des mini-cellules qui chevauchent sur deux cellules consécutives. Il est mesuré entre le temps d'entrée du premier bit de la mini-cellule dans l'émetteur et le temps de sortie du dernier bit du coté récepteur.

Le délai moyen de transfert des mini-cellules est égal à la moyenne arithmétique du temps de transfert des mini-cellules pour une population donnée. Il est donné par la formule :

$$\overline{X}=\frac{1}{N}\sum_{i=1}^{N}X_i \qquad \textbf{Équation 4.1}$$

où N est la grandeur de la population et X_i est le délai de la mini-cellule "i".

4.3.1.1.1.1 Le X^{ème}-percentile du délai

Le MTD peut être mesuré en terme de percentile du délai au lieu du délai moyen. Un X^{ème}-*percentile* du délai est le plus petit délai supérieur ou égal à X% des délais de la population (en général, X≥95). En d'autre terme, si "Dx" est le *X^{ème}-percentile* du délai d'une population, alors la probabilité pour qu'un paquet "i" ait un délai "Di" inférieur ou égal à Dx est de X%. Dans cette thèse, nous utilisons le 95ème-*percentile* et le 99,9^{ème}-*percentile* du délai. Ces deux valeurs donnent le délai maximal de la majorité des paquets de la population étudiée.

4.3.1.1.2 La variation du délai des mini-cellules (MDV : Minicell Delay Variation)

Ce paramètre représente la variation du délai de transfert des mini-cellules entre deux points du réseau. Il a une importance dans le cas du transport des applications temps-réel où une grande variation du délai serait percevable par l'utilisateur et provoque un effet gênant surtout pour le transport de la voix. Ce type d'application tolère une certaine variation du délai qui ne sera pas percevable par l'oreille de l'utilisateur. Ce paramètre peut être représenté de deux manières :

- Par la différence entre la borne maximale et la valeur minimale du délai de transfert des mini-cellules $\Delta T=T_{\max}-T_{\min}$. La valeur de ΔT doit être inférieure à une certaine valeur.
- Par la variance ou l'écart-type (*Standard Deviation : StdDev*) des valeurs des délais de transfert pour la population étudiée. L'écart-type représente la variation du délai par rapport à la valeur moyenne du délai.

  La variance est définie par : $VAR=\frac{1}{N-1}\sum_{i=1}^{N}\left(X_i-\overline{X}\right)^2$ **Équation 4.2**

  L'Ecart-Type est défini par : StdDev=$\sigma=\sqrt{VAR}$ **Équation 4.3**

  Où N est la grandeur de la population, X_i est le délai de la mini-cellule "i" et $\overline{X}$ est la moyenne du délai des mini-cellules de la population.

4.3.1.1.3 Le taux de perte des mini-cellules (MLR : Minicell Loss ratio)

Nous le définissons comme le rapport entre le nombre des mini-cellules perdues et le nombre des mini-cellules envoyées par la sous-couche SSCS vers la sous-couche CPS de l'émetteur pour une

population donnée. Une mini-cellule peut être perdue soit dans la file d'attente du multiplexeur CPS à cause d'un débordement de tampon (*buffer overflow*), soit à cause de la perte de la cellule ATM qui contient cette mini-cellule, soit à cause de la perte d'une partie de la mini-cellule qui chevauchent sur deux cellules ATM, soit à cause d'une erreur dans l'en-tête de la mini-cellule qui est détectée par le champ HEC de cet en-tête.

4.3.1.2 Paramètres de QoS au niveau SSCS

Nous définissons ces paramètres pour les paquets AAL2-SDU qui entrent dans le SAP de la couche AAL2, c'est à dire dans la sous-couche SSCS. Ce sont des paramètres de QoS de bout en bout, c'est à dire entre les deux extrémités d'une connexion AAL2. Notons que dans le cas où les paquets AAL2-SDU auraient une taille plus petite que la taille maximale d'un CPS-SDU (45 octets), les fonctions de segmentation de la sous-couche SSCS ne sont pas appliquées et par suite, les paramètres de performance au niveau SSCS (pour les AAL2-SDU) et ceux du niveau CPS (pour les mini-cellules) sont les mêmes. On peut rencontrer un tel cas pour les applications de voix compressée par exemple où la taille d'un paquet ne dépasse pas la taille maximale de la charge d'une mini-cellule.

4.3.1.2.1 Le délai de transfert des AAL2-SDU (ASTD : AAL2-SDU Transfert Delay)

Nous le définissons comme le délai de transfert d'un AAL2-SDU entre deux SAP de la couche AAL2 dans deux nœuds du réseau. Il correspond au délai de transfert MTD de la dernière mini-cellule qui compose l'AAL2-SDU. Ce délai est formé de plusieurs composantes:

- Le délai de segmentation des AAL2-SDU dans la sous-couche SSCS.
- Le délai de transfert entre les sous-couches CPS de l'émetteur et du récepteur. Il correspond au délai de transfert MTD de la dernière mini-cellule qui compose l'AAL2-SDU.
- Le délai de réassemblage dans le récepteur pour restituer les AAL2-SDU aux couches supérieures.

Ce délai tient compte de tous les mécanismes de la couche AAL2. Le délai de transfert moyen des AAL2-SDU est défini comme la moyenne arithmétique des délais de transfert d'une population donnée. Ce délai peut être exprimé soit en terme de valeur moyenne, soit en terme de percentile de la distribution des délais ASTD.

4.3.1.2.2 La variation du délai des AAL2-SDU (ASDV : AAL2-SDU Delay Variation)

Nous le définissons d'une manière similaire à la MDV mais au niveau des AAL2-SDU. Ce paramètre a une importance dans le cas de transport des données temps-réel comme la voix. Cette variation du délai tient compte du retard d'envoie des mini-cellules à cause des mécanismes d'ordonnancement au niveau du multiplexeur CPS surtout lorsqu'il y a des paquets AAL2-SDU segmentés en plusieurs mini-cellules.

4.3.1.2.3 Le taux de perte des AAL2-SDU (ASLR : AAL2-SDU Loss Ratio)

Nous le définissons comme le rapport entre le nombre des AAL2-SDU perdus et celui des AAL2-SDU soumis à la couche AAL2. Une unité de données AAL2-SDU peut être perdue lors de la perte de l'une ou de plusieurs des mini-cellules qui la constituent. Ce rapport est en corrélation avec le taux de perte des mini-cellules MLR. Soit n le nombre des mini-cellules générées lors de la segmentation d'une AAL2-SDU. Le taux de perte ASLR est donné alors en fonction du MLR par la relation suivante:

$$ASLR = \sum_{i=1}^{n} C_n^i MLR^i (1 - MLR)^{n-i} \qquad \textbf{Équation 4.4}$$

Si chaque AAL2-SDU est constituée d'une seule mini-cellule alors ASLR=MLR.

La figure 4.15 représente le taux ASLR en fonction du MLR. Il est très clair que lorsque le nombre des mini-cellules par AAL2-SDU augmente, le taux de perte des AAL2-SDU augmente rapidement en fonction du MLR.

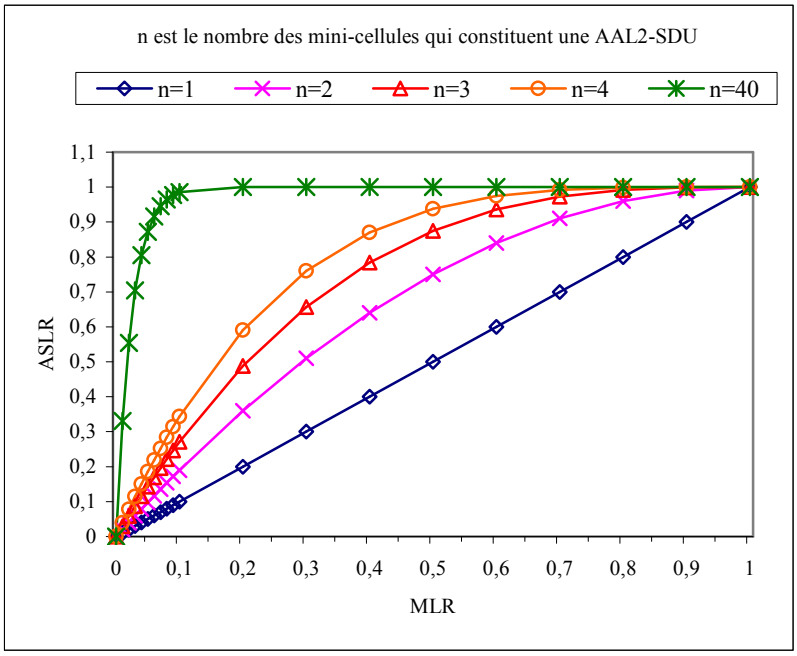


Figure 4.15 ASLR en fonction de MLR

4.3.2 Paramètres de trafic

Lors de l'établissement d'une connexion AAL2, l'utilisateur communique au réseau les valeurs des paramètres de QoS demandées ainsi que le profil de son trafic (débit maximal, taille des rafales, etc.) sous la forme d'un contrat de trafic. L'ensemble des paramètres de trafic forme le descripteur de trafic. Si le réseau accepte cette connexion, il devra alors garantir les bornes supérieures des paramètres de QoS tant que l'utilisateur respecte le contrat de trafic négocié, c'est à dire les critères de conformité. Dans ce paragraphe, nous allons définir des paramètres de trafic que nous jugeons indispensable en s'inspirant des travaux faits dans le cadre de la qualité de service dans les réseaux réel

Durant la procédure d'établissement d'une connexion AAL2, les liaisons AAL2 sont établies une après l'autre pour former une connexion AAL2 de bout en bout. Sur chaque liaison AAL2, ce sont les mini-cellules qui seront transportées et par suite, la négociation du contrat de trafic pour l'établissement d'une liaison AAL2 doit se baser sur le profil du trafic des mini-cellules. Pour cela, nous définissons des paramètres qui caractérisent le comportement du flux des mini-cellules au niveau de la sous-couche réel

Or, le profil du trafic des mini-cellules est en relation étroite avec celui du trafic des AAL2-SDU. Une longue AAL2-SDU génère une arrivée groupée de mini-cellules (*MG : Minicell Group*) au niveau réel Une définition des paramètres de trafic au niveau de la sous-couche SSCS est donc nécessaire. Quand un utilisateur demande l'établissement d'une connexion AAL2, il communique au réseau les paramètres du profil de son trafic ainsi que les valeurs des paramètres de QoS demandées. Ensuite, les liaisons AAL2 seront établies successivement en échangeant les paramètres du niveau réel Si une

partie du réseau n'est pas capable d'établir une liaison AAL2 entre deux points donnés avec la qualité de service requise, la procédure d'établissement de la connexion AAL2 sera alors échouée.

4.3.2.1 Paramètres de trafic et algorithmes de conformité au niveau CPS

Nous définissons ces paramètres entre deux entités CPS sur une liaison AAL2.

4.3.2.1.1 Débit crête PMBR et taille maximale de groupe MMGS

Nous définissons le débit crête des mini-cellules **PMBR** (***Peak Minicell Byte Rate***) comme la valeur maximale en octet/s du débit instantané des mini-cellules qui arrivent dans la sous-couche CPS pour une liaison AAL2 donnée. Ce débit prend en compte les 3 octets de l'en-tête de la mini-cellule. L'utilisateur envoie la valeur du PMBR de son flux dans le contrat de trafic avant de commencer à envoyer les mini-cellules. Les fonctions du test de conformité contrôlent les mini-cellules envoyées pour vérifier si l'utilisateur respecte le débit maximal négocié.

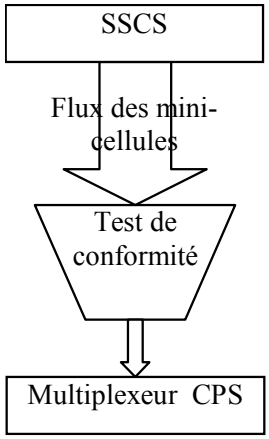


Figure 4.16 Test de conformité

La figure 4.16 représente la position des fonctions correspondantes au test de conformité du débit des mini-cellules. Quand une mini-cellule arrive, le test de conformité est déclenché. Si la mini-cellule est conforme au descripteur de trafic négocié, elle est acceptée pour entrer dans la file d'attente du multiplexeur, sinon elle peut être rejetée si la file d'attente est pleine ou même acceptée si le trafic est faible et le multiplexeur CPS n'a pas envoyé un message de congestion. La décision de conformité sera prise suivant l'algorithme du sceau à jetons (*Token Bucket*) représenté sur la figure 4.17.

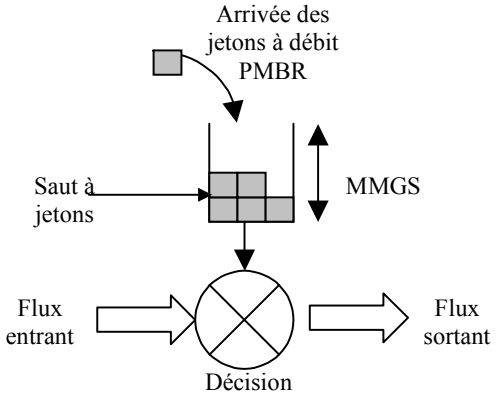


Figure 4.17 Sceau à jetons (Token Bucket)

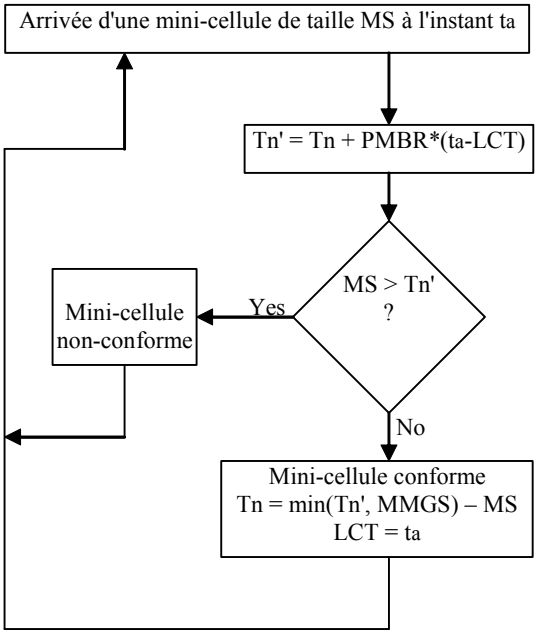
MMGS (***Maximum Minicell Group Size***) est la taille maximale tolérée d'un groupe de mini-cellules arrivant dans la sous-couche réel Cette valeur prend en compte les 3 octets d'en-tête AAL2. Une arrivée groupée de mini-cellules peut résulter par exemple de la segmentation d'un paquet AAL2-SDU. MMGS est toujours liée à PMBR et $MMGS \geq CPS_{max}+3$ où CPSmax est la taille maximale de la charge utile d'une mini-cellule (généralement 45 octets), "3" est la taille en octets de l'en-tête AAL2. Le sceau est rempli par les jetons avec un débit égal au débit crête PMBR. On considère qu'un jeton correspond à un octet. Si le sceau est plein, c'est à dire il y a MMGS jetons dans le sceau, alors les nouveaux jetons sont rejetés. Soit Tn (*Token number*) le nombre actuel des jetons dans le sceau. Tn' est une variable auxiliaire. Soit LCT (*Last Conformance Time*) l'instant d'arrivée de la dernière mini-cellule conforme. Soit "ta" l'instant d'arrivée d'une mini-cellule. Initialement, lors de l'arrivée de la première mini-cellule: *LCT=ta* et *Tn=MMGS.* Lors de l'arrivée d'une mini-cellule de taille MS (*Minicell Size*):

Tn' = Tn + PMBR*(ta-LCT)

Si MS>Tn' alors la mini-cellule n'est pas conforme; aucune modification des variables de l'algorithme.

Sinon la mini-cellule est conforme; Tn = min(Tn', MMGS)–MS et LCT = ta.

L'organigramme de la figure 4.18 représente l'algorithme de conformité du débit crête des mini-cellules. Cet algorithme tolère des petits piques de débit au-delà du débit crête mais pour des intervalles infinitésimaux.



Conditions initiales : LCT = ta; Tn = MMGS

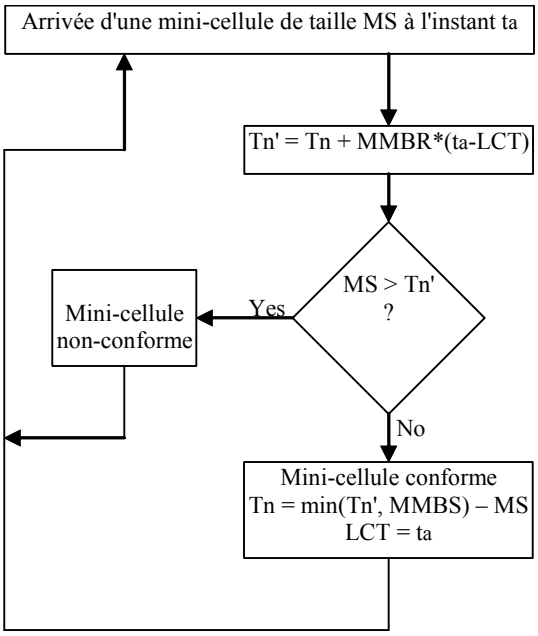
Figure 4.18 Algorithme de conformité du débit crête PMBR

4.3.2.1.2 Débit moyen MMBR et taille maximale de rafale MMBS

Le débit moyen des mini-cellules **MMBR** (***Mean Minicell Byte Rate***) est la valeur moyenne à long terme du débit des mini-cellules, c'est à dire sur une longue période (en octet/s). Généralement, le débit moyen MMBR est associé à la taille maximale d'une rafale de mini-cellules **MMBS** (***Maximum Minicell Burst Size***). Si pendant une certaine durée de temps, aucune mini-cellule n'est arrivée dans la

sous-couche CPS et après cette période, une rafale de mini-cellules arrive, ces mini-cellules seront considérées comme conformes si la taille totale de la rafale ne dépasse pas MMBS. Ceci est valable même si les mini-cellules de la rafale arrivent avec un débit plus élevé que MMBR. Sur une période très longue, le débit moyen doit rester inférieur ou égal à MMBR pour qu'il soit considéré comme conforme. L'algorithme de conformité est considéré comme un sceau à jetons déjà décrit dans le paragraphe §4.3.2.1.1. Mais dans ce cas, les jetons arrivent dans le sceau avec un débit MMBR au lieu de PMBR et la taille du sceau est égale à la taille maximale d'une rafale de mini-cellules MMBS au lieu de MMGS. L'organigramme de la figure 4.19 représente le fonctionnement de l'algorithme de conformité du débit moyen.

Généralement, la taille d'une rafale est plus grande que la taille maximale tolérée d'un groupe de mini-cellules (MMBS > MMGS). Sur une courte période égale à la longueur de la rafale, le débit peut être plus élevé que le débit moyen.



Conditions initiales : LCT = ta; Tn = MMBS

Figure 4.19 Algorithme de conformité du débit moyen MMBR

4.3.2.2 Paramètres de trafic et algorithmes de conformité au niveau SSCS

Nous définissons les paramètres de trafic au niveau SSCS pour permettre d'implémenter des mécanismes de contrôle de flux à l'entrée d'un sous-réseau AAL2. En effet, ces paramètres sont définis de bout en bout entre deux entités SSCS et sont étroitement liés aux paramètres de trafic du niveau réel

Le schéma de la figure 4.20 représente un sous-réseau AAL2 qui contient des commutateurs AAL2 où les flux remontent jusqu'à la sous-couche CPS seulement. Dans le nœud d'entrée de ce sous-réseau, on peut implémenter des mécanismes de contrôle de flux à l'entrée de la sous-couche SSCS. De cette manière, on peut contrôler le trafic entrant dans un sous-réseau pour éviter les problèmes de congestion et pour pouvoir garantir une certaine qualité de service. Dans les nœuds intermédiaires du sous-réseau, les mécanismes de contrôle de flux sont réalisés au niveau de la sous-couche réel

Le rôle de ces paramètres est de pouvoir contrôler en amont le trafic entrant dans la sous-couche SSCS avant qu'il entre dans la sous-couche CPS et pour qu'il soit conforme aux paramètres de trafic du niveau réel Ce contrôle de flux au niveau SSCS est important pour éviter la circulation des mini-cellules inutiles dans le sous-réseau. En effet, si un paquet AAL2-SDU perd une de ses mini-cellules,

les autres mini-cellules qui composent ce paquet seront inutiles. Si ces mini-cellules ne sont pas éliminées, alors une partie de la bande passante sera utilisée pour un flux de mini-cellules inutiles. En définissant des paramètres de trafic et des algorithmes de conformité au niveau SSCS, on peut éliminer les paquets AAL2-SDU non-conformes à l'entrée du réseau et par conséquent on évite l'envoie des mini-cellules inutiles même si une partie de ces mini-cellules pourra être conforme aux algorithmes de conformité du niveau réel

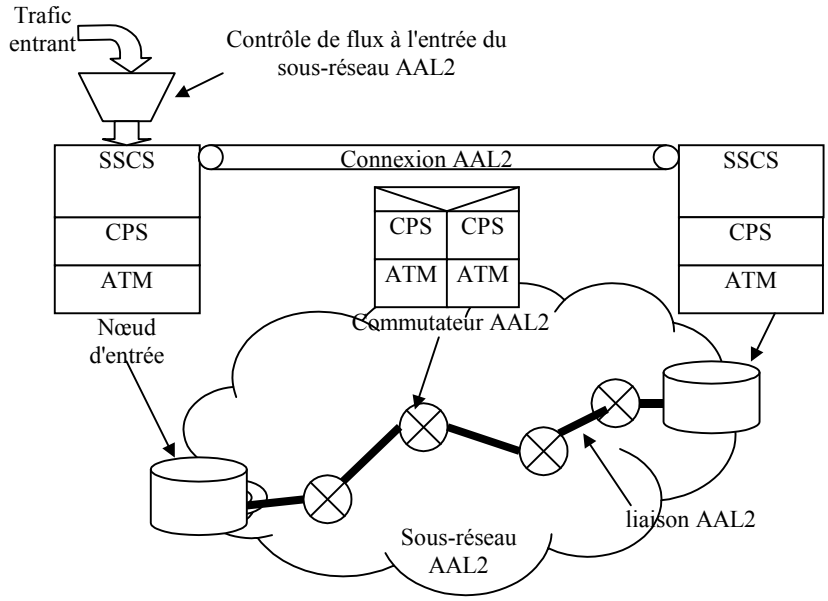


Figure 4.20 Contrôle de flux à l'entrée d'un sous-réseau AAL2

Puisque les paramètres de trafic du niveau SSCS sont liés à ceux du niveau CPS, alors lors de l'établissement d'une connexion AAL2, il suffit de communiquer les paramètres de trafic du niveau CPS et on peut en déduire ceux du niveau SSCS. Notons que les fonctions de contrôle de trafic au niveau SSCS peuvent être nulles et par suite, les mécanismes de contrôle de flux seront assurés au niveau CPS uniquement.

Dans la suite de ce paragraphe, nous allons définir les paramètres de trafic du niveau SSCS que nous jugeons indispensables pour contrôler les flux à l'entrée d'un sous réseau AAL2.

4.3.2.2.1 Taille maximale d'un paquet AAL2-SDU (MASS : Maximum AAL2-SDU Size)

C'est la taille maximale en octets d'un paquet AAL2-SDU qui sera décomposé en des mini-cellules. Les en-têtes des mini-cellules ne sont pas pris en compte. Ce paramètre est en relation avec la taille maximale tolérée d'un groupe de mini-cellules MMGS. En effet, soit "*AAL2_SDU_Size*" la longueur d'un paquet AAL2-SDU. Soit "*Group_Size*" la taille du groupe des mini-cellules générées par la segmentation de ce paquet. Alors ces deux variables sont liées par la relation suivante:

$$Group\_Size = 3 \times \left\lceil \frac{AAL2\_SDU\_Size}{CPS\max} \right\rceil + AAL2\_SDU\_Size \quad \textbf{Équation 4.5}$$

"*CPSmax*" est la taille maximale de la charge utile d'une mini-cellule (généralement 45 octets). Le chiffre "3" désigne la taille en octets de l'en-tête AAL2.

$\lceil x \rceil$ est le plus petit nombre entier plus grand ou égal à x.

Le terme $\left\lceil \frac{AAL2\_SDU\_Size}{CPS\max} \right\rceil$ désigne le nombre des mini-cellules générées par la segmentation de la AAL2-SDU.

Ces mini-cellules entrent dans la sous-couche CPS sous forme d'un groupe (*Minicell Group*). La taille de ce groupe ne doit pas dépasser la valeur négociée du MMGS ce qui impose une limite sur la taille d'une AAL2-SDU. La taille maximale d'une AAL2-SDU (MASS) est alors liée à MMGS par la formule suivante:

$$3\times\left\lceil \frac{MASS}{CPS\ \max} \right\rceil + MASS \leq MMGS$$

Équation 4.6

L'algorithme de conformité au niveau SSCS pour la taille d'un paquet AAL2-SDU est très simple : si la taille d'un paquet AAL2-SDU arrivant dans la sous-couche SSCS est inférieure à MASS, alors ce paquet est conforme, sinon il n'est pas conforme. Deux solutions sont possibles pour un paquet non-conforme:

- soit il traverse la sous-couche SSCS après segmentation mais il n'y aura aucune garantie pour les mini-cellules générées qui traversent le réseau.
- soit il sera rejeté si le réseau est congestionné.

En effet, si un paquet AAL2-SDU non-conforme passe à travers la SSCS, les mini-cellules non-conformes générées seront rejetées par les mécanismes de contrôle de flux au niveau CPS lors d'une congestion. Sachant que la perte d'une mini-cellule appartenant à un paquet AAL2-SDU entraîne la perte de tout le paquet, les mini-cellules conformes seront envoyées dans le réseau mais rejetées à l'arrivée, c'est à dire lors du réassemblage dans la SSCS de réception. Par conséquent une partie de la bande passante sera utilisée pour un trafic de mini-cellules inutiles. C'est pourquoi la deuxième solution semble importante dans certains cas.

Etant donnée la valeur du MMGS qui est échangée lors de l'établissement de la connexion, la valeur du paramètre MASS peut être calculée à partir de la valeur du MMGS par la méthode suivante:

Soit U_i une suite numérique de nombres entiers. Elle est définie d'après l'équation 4.6 comme suit:

$$U_{i+1}=MMGS-3\times\left\lceil \frac{U_i}{CPS_{\max}} \right\rceil \text{ avec } U_0=1 \quad \textbf{Équation 4.7}$$

On suppose que U_i est une suite convergente ayant pour valeur limite U_L, alors MASS = U_L. Pour calculer cette valeur limite, il suffit de procéder par itération jusqu'à ce que $U_{i+1}=U_i=U_L$.

4.3.2.2.2 Débit crête des AAL2-SDU (PASBR : Peak AAL2-SDU Byte Rate)

C'est la valeur maximale en octet/s du débit des paquets AAL2-SDU. Ce débit à l'entrée de la sous-couche SSCS va générer un débit plus élevé au niveau de la sous-couche CPS à cause des en-têtes AAL2. Le débit généré au niveau CPS doit être inférieur ou égal au débit PMBR négocié. Il est difficile de déterminer la valeur de PASBR à partir de la valeur de PMBR à cause de la variabilité de la taille des paquets. En effet, prenons un exemple simple où le débit crête PMBR est fixé à 10 koctet/s. Ce débit peut être généré par plusieurs façons. Prenons deux possibilités : dans la première, une mini-cellule de 10 octets arrive chaque 1 ms, dans la deuxième une mini-cellule de 40 octets arrive toutes les 4 ms. Dans les deux cas, le débit au niveau CPS est de 10 koctet/s, mais au niveau SSCS le débit est différent. Dans le premier cas, on a un paquet AAL2-SDU de 7 octets (on a enlevé l'en-tête

AAL2) qui arrive chaque 1 ms d'où un débit de 7 koctet/s. Dans le deuxième cas, un paquet AAL2-SDU de 37 octets arrive toutes les 4 ms d'où un débit de 9,25 koctet/s. Il est clair que pour un même débit PMBR négocié, le PASBR peut être plus élevé pour des paquets AAL2-SDU de plus grande taille. La figure 4.21 représente le rapport entre le débit CPS et celui du niveau SSCS en fonction de la taille des paquets AAL2-SDU. Ce rapport varie entre 1,05 et 4 d'où la difficulté de trouver une relation exacte entre PASBR et PMBR dans le cas où les paquets auraient des tailles variables.

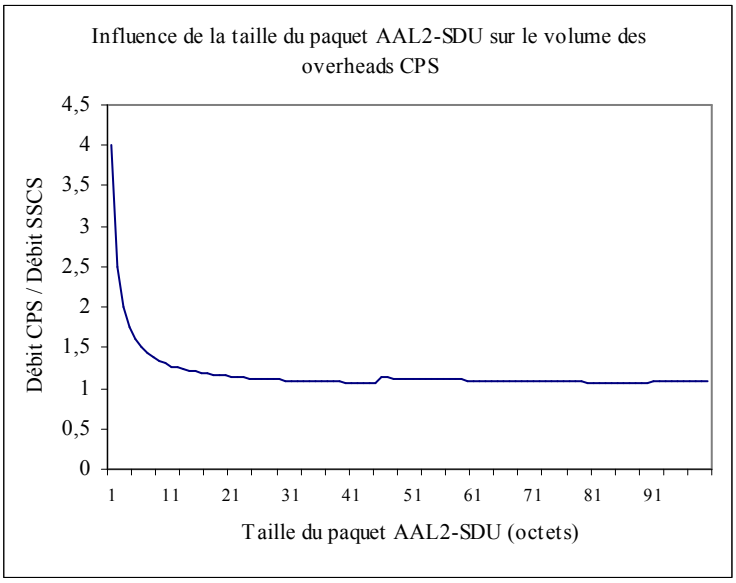


Figure 4.21 Débit CPS/débit SSCS et taille des paquets

Dans le cas où tous les paquets AAL2-SDU d'un flux donné auraient une taille fixe, notée p, on peut trouver la relation suivante entre PASBR et PMBR:

$$PASBR = PMBR \times \frac{p}{3 \times \left\lceil \frac{p}{CPS_{\max}} \right\rceil + p}$$ **Équation 4.8**

Dans ce cas, l'algorithme de conformité du débit crête des AAL2-SDU (PASBR) est un sceau à jetons dont le débit d'arrivée des jetons est PASBR et la taille du sceau est de **MASGS** (***Maximum AAL2-SDU Group Size***). MASGS représente la taille maximale en octets d'un groupe de paquets AAL2-SDU qui peuvent arriver à la couche SSCS sans violer les critères de conformité. La valeur de MASGS doit être choisie de manière que le groupe de mini-cellules générées par ce groupe de AAL2-SDU ait une taille inférieure ou égale à la taille maximale tolérée d'un groupe de mini-cellules MMGS. Cela se traduit par la relation suivante:

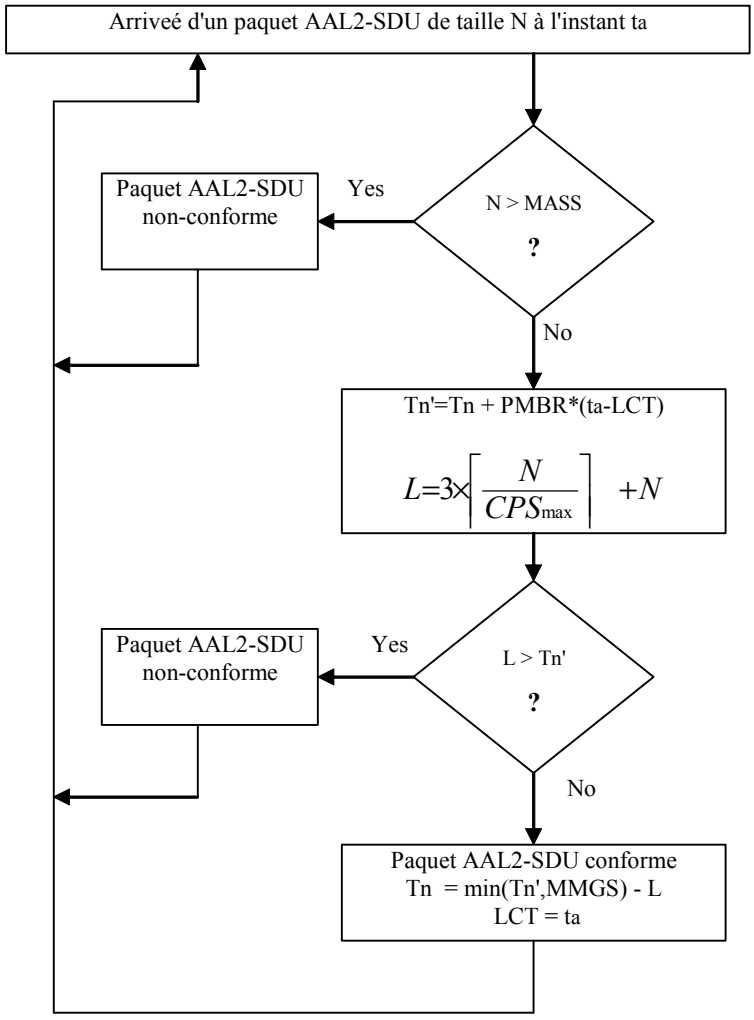
$$MASGS \leq MMGS \times \frac{p}{3 \times \left\lceil \frac{p}{CPS_{\max}} \right\rceil + p}$$ **Équation 4.9**

Dans le cas où la taille des paquets AAL2-SDU serait variable, on ne peut pas trouver la valeur exacte du PASBR, mais on peut toujours établir un algorithme de conformité. Cet algorithme est important dans le cas où on voudrait contrôler le trafic à l'entrée d'un sous-réseau AAL2. Cet

algorithme est général et peut être appliqué quelle que soit la taille des paquets AAL2-SDU. Il est décrit dans l'organigramme de la figure 4.22.

Le mécanisme est aussi un sceau à jetons mais le débit d'arrivée des jetons est le PMBR. La taille du sceau est MMGS. Quand un paquet AAL2-SDU de taille N arrive à l'instant t_a, l'algorithme vérifie la conformité de la taille de ce paquet, elle doit être inférieure à MASS. Si ce n'est pas le cas, alors le paquet est non conforme, sinon l'algorithme continue la vérification de conformité:

Tn (*Tocken number*) est le nombre des jetons dans le sceau (1 jeton correspond à 1 octet). Tn' est une variable auxiliaire. LCT (*Last Conformance Time*) est l'instant d'arrivée du dernier paquet conforme. "L" est une variable auxiliaire, elle désigne le nombre d'octets qui vont être générés au niveau CPS par le paquet AAL2-SDU (y compris les en-têtes AAL2). Initialement, LCT est égal à l'instant d'arrivée du premier paquet et Tn = MMGS. L'algorithme calcule les valeurs de L et de Tn' (voir organigramme). Si L > Tn' alors le paquet est non-conforme, sinon il est conforme et les variables sont mises à jour. Aucune mise à jour n'est effectuée lors de l'arrivée d'un paquet non-conforme.



Conditions initiales: LCT = ta, Tn = MMGS

Figure 4.22 Algorithme de conformité du PASBR

4.3.2.2.3 Débit moyen des AAL2-SDU (MASBR : Mean AAL2-SDU Byte Rate)

C'est la valeur moyenne à long terme en octet/s du débit des paquets AAL2-SDU. De même que pour PASBR, la détermination du MASBR en fonction du débit moyen des mini-cellules MMBR est difficile à cause de la taille variable des paquets AAL2-SDU. Toutefois, nous pouvons implémenter un algorithme de conformité complexe pour vérifier si les paquets AAL2-SDU entrant dans le sous-réseau AAL2 sont conformes aux paramètres de trafic négociés. L'algorithme de conformité est un sceau à jetons dont le débit d'arrivée est de MMBR et la taille du sceau est de MMBS. Cet algorithme est représenté sur le schéma de la figure 4.23. Il est semblable à l'algorithme de conformité de PASBR mais avec changement de quelques paramètres.

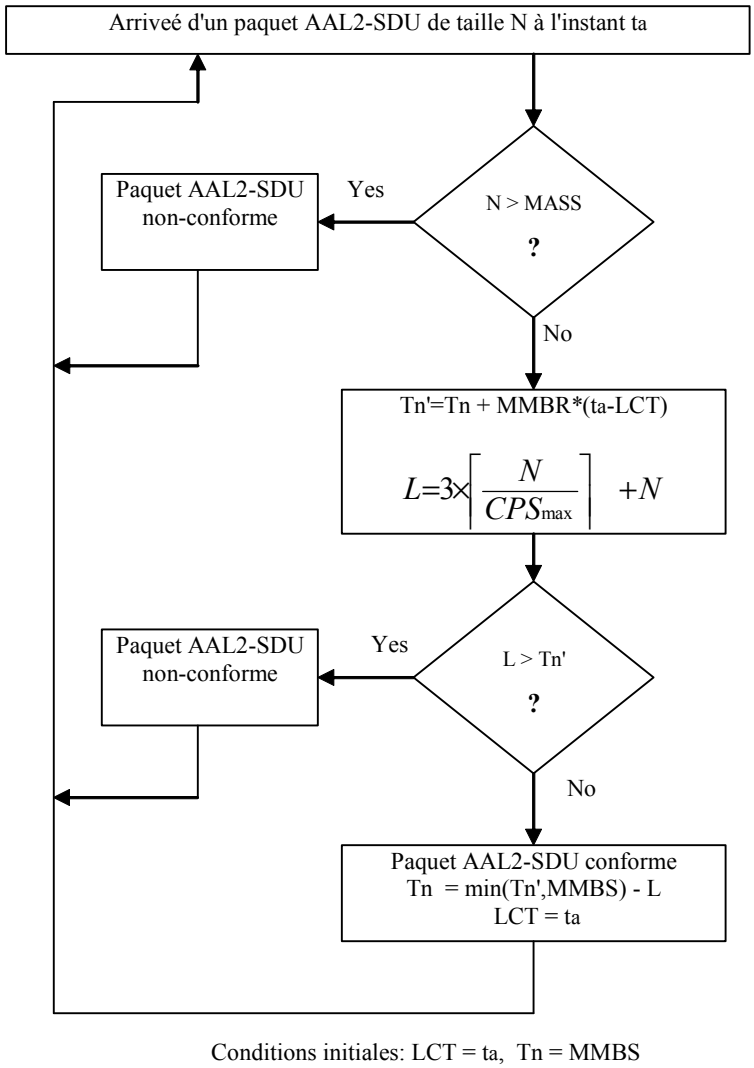


Figure 4.23 Algorithme de conformité du MASBR

4.3.3 Les classes de qualité de service de l'AAL2 (*AAL2 QoS Classes*)

Une classe de QoS est une combinaison spécifique de limites des valeurs des paramètres de QoS définies comme objectif de qualité de service pour le type d'application transportée. Nous définissons ces classes pour faciliter la procédure de négociation entre l'utilisateur et le réseau. Lors de l'établissement de la connexion, l'utilisateur communique au réseau la classe de service souhaitée pour acheminer ses paquets. Cette négociation comporte implicitement les limites des valeurs des paramètres de QoS. Si le réseau est capable d'assurer ces valeurs limites, il accepte la nouvelle

connexion, sinon il envoie explicitement à l'utilisateur les valeurs limites qu'il pourra garantir. L'utilisateur peut alors accepter ou refuser selon ces valeurs et selon ses besoins de qualité de QoS. Les valeurs limites des paramètres de QoS dans le réseau peuvent être estimées après des mesures faites sur le trafic actuel du réseau. Des mini-cellules de supervision peuvent être aussi utilisées pour réaliser cette tache (voir §4.3.5).

Il est toujours possible que l'utilisateur n'utilise pas des classes de QoS prédéfinies et qu'il communique explicitement une ou plusieurs valeurs limites des paramètres de QoS selon ses besoins. Dans ce cas, la classe de QoS ne serait pas spécifiée lors de l'établissement de la connexion. Dans cette thèse, nous définissions quatre classes de QoS au niveau AAL2. Il est toujours possible de définir des nouvelles classes si on trouve un besoin indispensable.

4.3.3.1 Classe temps-réel sévère SRT (Stringent Real Time class)

Nous définissons cette classe pour le transport des flux temps-réel très exigeants en terme de délai de transfert et de variation de délai. Les valeurs limites du délai de transfert, de la variation du délai et du taux de perte sont définies en amont lors de la configuration du réseau. Les paramètres de QoS peuvent être définis au niveau CPS ou au niveau SSCS ou même simultanément aux deux niveaux. De toute façon, les paramètres des deux niveaux ne sont pas indépendants et on peut toujours trouver une relation entre eux. Dans cette étude, nous allons définir des valeurs limites des paramètres du niveau SSCS, c'est à dire sur une connexion AAL2. En effet, nous nous intéressons dans cette thèse à l'étude de l'AAL2 dans le contexte de l'UTRAN. Les unités des données transportées dans l'UTRAN sont les trames radio encapsulées dans les FP-PDU. Les contraintes de l'UTRAN s'appliquent sur ces unités des données. C'est pourquoi nous nous intéressons aux paramètres de QoS du niveau SSCS car chaque FP-PDU constitue un paquet SSCS (AAL2-SDU). En tenant compte des contraintes de l'UTRAN [91], nous définissons les valeurs limites des paramètres de cette classe dans le tableau 4.1. Ces valeurs sont applicables dans le contexte de l'UTRAN uniquement.

ASTD	ASDV	ASLR
5 ms	1 ms	10^{-2}

Tableau 4.1 Valeurs limites des paramètres de QoS de la classe SRT

4.3.3.2 Classe temps-réel tolérante TRT (Tolerant Real Time class)

Cette classe est utilisée pour les applications dont les contraintes temporelles ne sont pas très strictes mais elles sont sensibles au taux de perte. La variation du délai n'est pas spécifiée pour cette classe. Le tableau 4.2 donne les valeurs limites des paramètres de QoS spécifiées pour cette classe.

ASTD	ASLR
50 ms	10^{-4}

Tableau 4.2 Valeurs limites des paramètres de QoS de la classe TRT

4.3.3.3 Classe non temps-réel NRT (Non Real Time class)

Cette classe n'a pas de contraintes temporelles mais elle est sensible aux pertes. La variation de délai n'est pas spécifiée pour cette classe mais le délai doit être inférieur à un certain seuil pour que les

paquets arrivent à leurs destinations dans une durée de temps acceptable. Les limites des paramètres de QoS sont données dans le tableau 4.3.

ASTD	ASLR
1 sec	10^{-6}

Tableau 4.3 Valeurs limites des paramètres de QoS de la classe NRT

4.3.3.4 Classe du meilleur effort BE (Best Effort)

Dans cette classe, il n'y a aucune spécifications des valeurs limites des paramètres de QoS. Aucune garantie n'est fournie par le réseau pour les flux de cette classe, ni en terme de délai ni de taux de perte. Cette classe est semblable à la classe *Best Effort* dans les réseaux Internet actuels.

4.3.4 Les capacités de transfert de l'AAL2 (*AAL2-TC : AAL2-Transfer Capabilities*)

La qualité de service sera fournie sous la forme de capacités de transfert. Une capacité de transfert au niveau AAL2 (AAL2-TC) est un ensemble de paramètres relatifs au trafic transporté. Lors de l'établissement d'une connexion AAL2, l'utilisateur demande au réseau l'AAL2-TC sur laquelle il veut acheminer son trafic et lui communique les valeurs des paramètres de trafic relatifs à son profil (le descripteur de trafic) ainsi que la classe de QoS souhaitée. Comme on l'a déjà dit dans le paragraphe précédant, les bornes supérieures des paramètres de QoS (délai, taux de perte, etc.) peuvent être communiquées explicitement sans utiliser les classes de QoS. Le réseau doit alors réserver les ressources nécessaires pour garantir la qualité de service demandée. L'allocation des ressources sera effectuée sur la base des paramètres de trafic négociés (débit crête, débit moyen, etc.). Si les ressources disponibles dans le réseau sont suffisantes pour établir cette nouvelle connexion, alors elle est acceptée, sinon la demande de connexion est refusée. Une fois que le réseau a accepté la nouvelle connexion, il doit alors garantir les bornes supérieures des paramètres de QoS à condition que l'utilisateur respecte le contrat de trafic et que le flux envoyé ne dépasse pas les ressources réservées.

Dans cette thèse, nous définissions trois capacités de transfert au niveau AAL2. Une AAL2-TC pour les flux à débits constants, une AAL2-TC pour les flux à débits variables et enfin une AAL2-TC pour les flux qui n'ont pas de contraintes de qualité de service.

Il n'y a pas une association particulière entre les classes de QoS et les AAL2-TC. Une capacité de transfert peut supporter les flux des différentes classes de QoS selon les besoins de l'utilisateur et les capacités du réseau. Donc lors de l'établissement de la connexion, l'utilisateur doit indiquer la classe de QoS souhaitée ou communiquer explicitement ses limites de paramètres de QoS. D'ailleurs, il y a des AAL2-TC qui ne sont pas capables de supporter certaines classes. Nous allons voir cela dans les paragraphes concernant chaque capacité de transfert.

4.3.4.1 AAL2-CBR (AAL2-Constant Bit Rate)

C'est la capacité de transfert des flux à débits constants. L'allocation des ressources se fait sur la base du débit crête de l'utilisateur qui est échangé lors de l'établissement de la connexion AAL2. La bande passante allouée pour un ensemble de sources est égale à la somme des débits crêtes des différentes sources. L'utilisateur doit respecter l'algorithme de conformité relatif au débit crête et le réseau doit alors garantir les paramètres de QoS négociés. Les flux transportés par cette classe sont généralement de type temps-réel mais on peut transporter sur cette AAL2-TC tous types de classe de QoS.

Cette AAL2-TC est utilisée pour les flux à débits constants. Si l'utilisateur émet toujours à débit crête, alors la bande passante réservée est bien utilisée. Dans le cas où l'utilisateur n'émettrait pas toujours à débit crête, la bande passante réservée ne peut pas être utilisée par les autres classes d'où une perte de bande passante. Pour remédier à ce problème, une AAL2-TC est définie dans le paragraphe suivant pour les flux à débits variables.

4.3.4.2 AAL2-VBR (AAL2-Variable Bit Rate)

Nous définissons la capacité de transfert à débit variable pour le transport des applications dont le débit instantané change avec le temps et surtout pour les trafics qui présentent un comportement sporadique "*Bursty Traffic*". Cette AAL2-TC est spécifiée par un débit crête, un débit moyen et une longueur maximale de rafale. L'utilisateur peut envoyer un trafic à débit crête pendant une période équivalente à la durée de rafale tolérée. A long terme, le débit de l'utilisateur ne doit pas dépasser le débit moyen négocié. Si le trafic de l'utilisateur est conforme au contrat de trafic négocié, alors le réseau doit garantir les bornes supérieures des paramètres de QoS demandées par l'utilisateur. Il y a deux conditions de conformité pour cette AAL2-TC : tout d'abord, il faut que le trafic soit conforme au débit crête négocié puis il faut qu'il soit conforme au débit moyen avec la taille maximale de rafale tolérée. L'allocation des ressources se fait sur la base du débit moyen, de la longueur maximale de rafale et du débit crête. Cette manière d'allocation des ressources permet de récupérer la bande passante non utilisée par une source pour l'utiliser par d'autres flux. En effet, la bande passante réservée pour une source est égale au débit moyen et non pas au débit crête et par conséquent, le réseau peut profiter du gain statistique pour accepter un nombre plus grand de connexions. Dans ce cas, il y aura une sur-allocation des ressources en espérant que toutes les sources n'émettent pas à débit crête en même temps. Toutes les classes de QoS peuvent être transportées par cette capacité de transfert.

4.3.4.3 AAL2-ABR (AAL2-Available Bit Rate)

C'est la capacité de transfert qui n'exige pas la négociation de paramètres de trafic. Cette capacité de transfert n'offre aucune garantie pour les flux transportés, ni en terme de délai ni en terme de taux de perte. Le réseau ne fait pas de contrôle d'admission pour les nouvelles connexions et par suite aucun paramètre de trafic n'est échangé et aucune allocation des ressources n'est effectuée. Il n'y a pas de garantie de débit pour les flux AAL2-ABR qui partagent entre eux la bande passante restante après l'allocation des ressources des autres AAL2-TC. En effet, les utilisateurs AAL2-VBR n'envoient pas toujours des flux à débit crête et par suite, une partie de la bande passante réservée n'est pas utilisée. Dans ce cas, les flux AAL2-ABR viennent pour utiliser cette bande passante résiduelle et remplir le tuyau.

Toutefois, il est possible, lors de la configuration du réseau, de réserver une partie de la bande passante totale pour les flux AAL2-ABR et cette partie sera partagée par tous ces flux. De cette manière, on évite que les autres AAL2-TC "privilégiées" consomment toute la bande passante disponible et empêchent totalement le passage des flux AAL2-ABR.

Cette capacité de transfert ne peut pas transporter des flux appartenant aux classes SRT, TRT et NRT. En effet, ces trois classes exigent le respect des valeurs limites des paramètres de QoS et cette AAL2-TC ne peut pas les satisfaire parce qu'elle utilise les ressources résiduelles du réseau et ne peut pas fournir aucune garantie. Pour l'instant, la seule classe qui peut être transportée par cette capacité de transfert est la classe BE.

4.3.5 Trafic de supervision

Pour que le réseau puisse négocier un contrat de trafic avec un nouvel utilisateur, il faut qu'il connaisse l'état actuel du trafic circulant en terme de paramètres de QoS. Des mini-cellules de supervision (*SM : Supervision Minicell*) circulent alors dans le réseau et des statistiques sont

effectuées sur ces mini-cellules pour mesurer les valeurs actuelles des paramètres de QoS (délai, taux de perte, etc.). A partir de ces valeurs mesurées, le réseau peut accepter ou refuser une nouvelle connexion en fonction des bornes supérieures des paramètres de QoS demandées par l'utilisateur. Si ses bornes sont inférieures aux valeurs actuelles mesurées, alors le réseau pourra garantir la qualité de service demandée par l'utilisateur et par suite il accepte la nouvelle connexion après avoir fait la réservation des ressources nécessaires, sinon la nouvelle connexion est refusée.

Les mini-cellules de supervision peuvent être envoyées sur des connexions AAL2 spécifiques ou sur des connexions déjà établies pour le trafic utilisateur. Dans le premier cas, des CID doivent alors être réservés pour les connexions de supervision et dans ce cas, on peut faire des mesures sur les paramètres de QoS du niveau SSCS ainsi que du CPS, c'est à dire sur une liaison AAL2, puisqu'on peut identifier les mini-cellules de supervision au niveau CPS par leur CID. Dans le deuxième cas, les mini-cellules de supervision sont intercalées entre les mini-cellules du trafic utilisateur et ont le même CID, donc au niveau CPS, il n'y a pas de moyen pour identifier ces mini-cellules SM. Dans ce cas, l'identification des mini-cellules de supervision peut se faire au niveau SSCS. En effet, le champ UUI (*User to User Indication*) de la mini-cellule transporte des informations entre les sous-couches SSCS des deux extrémités. Il est transparent par rapport à la sous-couche réel On peut alors utiliser des valeurs réservées de ce champ pour identifier les mini-cellules de supervision au niveau SSCS. De cette manière, on peut mesurer les paramètres de QoS du niveau SSCS, c'est à dire de bout en bout sur une connexion AAL2.

Les mini-cellules de supervision contiennent dans leurs charges utiles (*payload*) des informations nécessaires pour mesurer les paramètres de QoS. Par exemple, pour mesurer le délai de transfert, on a besoin de savoir l'instant d'émission de la mini-cellule ainsi que l'instant d'arrivée au point de mesure. L'instant d'émission peut être alors codé et envoyé dans la charge de la mini-cellule de supervision. Cette charge utile est de 47 ce qui est largement suffisant pour avoir une haute précision de mesure. A l'arrivée de la mini-cellule, l'instant d'émission sera utilisé avec l'heure actuelle dans le réseau pour mesurer le délai de transfert. Cette méthode nécessite une synchronisation entre les horloges des différents nœuds du réseau.

4.4 Efficacité du transport en AAL2/ATM

Les paragraphes précédents traitent les aspects liés à la qualité de service des applications transportées en AAL2/ATM. D'ailleurs, l'efficacité du transport en AAL2 doit être évaluée surtout que le protocole AAL2 ajoute des informations nécessaires au transport. Ces informations (les *overhead*) peuvent dégrader l'efficacité du transport en AAL2. En plus, les mécanismes de multiplexage de la sous-couche CPS peuvent introduire des octets de bourrage, par exemple quand la valeur du Timer-CU est faible, ce qui entraîne une perte de la bande passante disponible. Pour évaluer l'efficacité du transport des données en AAL2/ATM, nous définissions trois critères d'efficacité.

4.4.1 Taux d'efficacité de la bande passante (ER : *Efficiency Ratio*)

Le protocole AAL2 introduit des octets supplémentaires à cause des en-têtes de la sous-couche CPS, le STF et le bourrage. Si l'unité des données a une taille faible par rapport aux "*overheads*", la bande passante sera mal utilisée. En outre, le protocole ATM introduit 5 octets d'en-tête pour chaque cellule. Le taux d'efficacité de la bande passante est défini comme le rapport entre le nombre total des octets envoyés par la sous-couche SSCS (les octets du AAL2-SDU) et la taille de l'ensemble des cellules ATM (53 octets) utilisées pour transporter ces AAL2-SDU. Les mesures sont faites sur des intervalles de temps assez larges pour que le nombre des échantillons soit significatif. Le taux d'efficacité maximal est obtenu lorsque toutes les mini-cellules ont la taille maximale de 45 octets et les cellules ATM sont pleines (pas de bourrage).

4.4.2 Taux de remplissage (FR : *Filling Ratio*)

Ce paramètre représente la bande passante perdue en terme d'octets de bourrage. Dans la *payload* de la cellule ATM, un octet est utilisé systématiquement pour le STF, les 47 octets restants sont utilisés par les mini-cellules ou sont remplis du bourrage. Le FR est défini comme le rapport: **FR = I / 47**, où I est le nombre d'octets utiles (informations). Ce paramètre reflète l'impact de la valeur du Timer-CU.

4.4.3 Taux d'utilisation de la bande passante (UR : *Utilisation Ratio*)

C'est le rapport entre le débit moyen au niveau ATM et la capacité maximale du lien. Dans le cas où le PCR (*Peak Cell Rate*) serait défini, le taux d'utilisation est défini comme:

UR = mean_BR / PCR, où mean_BR est le débit cellulaire moyen.

Ce paramètre reflète le seuil d'utilisation d'un tuyau tout en respectant un délai acceptable.

4.5 Conclusion

Dans ce chapitre, nous avons proposé une architecture de qualité de service au niveau AAL2. Cette architecture est composée de deux niveaux : SSCS et CPS. Les mécanismes liés au niveau CPS garantissent la qualité de service au sein d'un sous-réseau AAL2, c'est à dire entre les commutateurs AAL2. Les mécanismes du niveau SSCS contrôlent le trafic à l'entrée d'un sous- réseau AAL2. La notion de la qualité de service au niveau AAL2 est intéressante dans notre étude de la QoS dans l'UTRAN parce que les canaux radio, transportés sur les canaux AAL2 sur les interfaces Iub et Iur, ont des contraintes temporelles strictes. Le protocole AAL2 doit alors fournir des services capables de garantir ces contraintes. Notons que les classes de QoS de l'AAL2 définies dans ce chapitre ne sont pas toutes utilisées dans le cas de l'UTRAN. Par exemple, les classes NRT et BE ne sont pas adaptées au transport des canaux radio parce qu'elles ne peuvent pas satisfaire les contraintes de l'UTRAN. De même pour les capacités de transfert AAL2, l'AAL2-ABR ne peut pas être utilisée dans le contexte de l'UTRAN. Elles sont définies dans un cas général et peuvent être utilisées dans un contexte différent de l'UTRAN, par exemple pour le *truncking*.

Cette architecture de qualité de service au niveau AAL2 vient pour compléter celle du niveau ATM. Elle fournit un modèle complet de la qualité de service pour le transport des applications multimédia sur le protocole AAL2/ATM.

Chapitre 5

5 La QoS et la gestion des flux dans l'UTRAN

5.1 Introduction

Les systèmes des télécommunications mobiles de troisième génération vont fournir des nouveaux services pour l'utilisateur allant des services de voix classiques jusqu'aux services multimédia transportant des images et de la vidéo. Pour que les nouveaux services de l'UMTS soient concurrents avec les services déjà existants dans les systèmes GSM et GPRS, la qualité des nouveaux services doit être satisfaisante. La maîtrise de la qualité de service des flux transportés ainsi que la gestion de ces flux garantiront un bon fonctionnement du système, une qualité satisfaisante des applications transportées et une utilisation efficace des ressources de l'infrastructure pour que la tarification des services fournis soit concurrente.

La qualité de service dans les réseaux UMTS doit être maîtrisée de bout en bout, c'est à dire entre les terminaux des deux extrémités de la communication. Pour pouvoir gérer la qualité de service de bout en bout, on essaye en général de partager le chemin en plusieurs parties et de garantir la qualité de service sur chacune des parties pour aboutir à une garantie globale. Dans le cas de l'UMTS, on peut décomposer la notion de la qualité de service en trois grandes parties : la qualité de service dans le réseau d'accès de départ, la qualité de service dans le réseau cœur (*Core Network*) et la qualité de service dans le réseau d'accès d'arrivée.

5.2 La qualité de service dans les réseaux UMTS

5.2.1 Architecture générale de la QoS

Les services du réseau UMTS sont des services de bout en bout, c'est à dire d'un équipement terminal (*TE : Terminal Equipement*) à un autre TE. Un service de bout en bout doit avoir une certaine qualité de service fournie à l'utilisateur pour satisfaire sa demande. Pour pourvoir garantir une certaine qualité de service, des services supports (*BS: Bearer Services*) sont définis entre la source et la destination d'un service. Un service support comporte tous les aspects nécessaires pour garantir la qualité de service demandée. La figure 5.1 représente l'architecture des services supports de l'UMTS qui est décomposée en plusieurs niveaux. Chaque service support d'un niveau donné utilise les services offerts par le niveau sous-jacent pour offrir à son tour ses propres services [88, 89, 99].

- Le service de bout en bout (*End-to-End Service*) au niveau application utilise les services supports (BS) du réseau sous-jacent. Il s'étend entre l'équipement terminal (*TE : Terminal Equipement*) de départ et le TE d'arrivée. Ce service peut être transporté sur plusieurs réseaux qui peuvent être des réseaux non UMTS. Ce service utilise le service support local TE/MT, le service support UMTS et le service support externe.

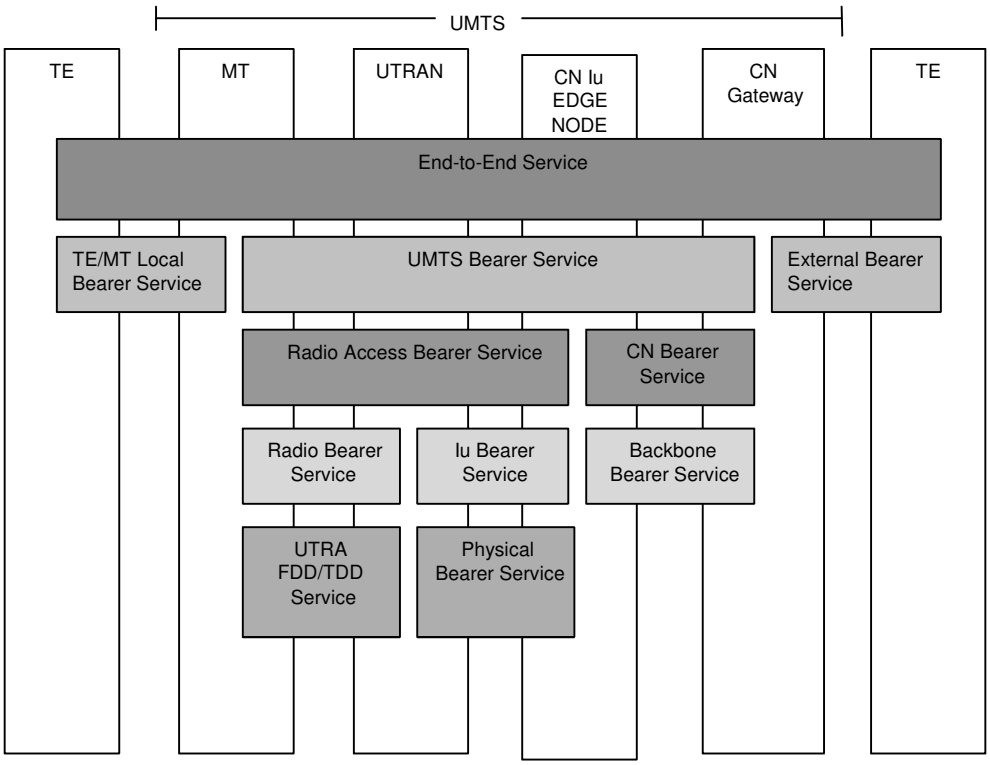


Figure 5.1 Architecture de la qualité de service dans l'UMTS

- L'équipement terminal est lié au réseau UMTS à travers le terminal mobile (*MT: Mobile Terminal*). Le service support local TE/MT (*TE/MT Local Bearer Service*) assure la liaison entre le TE et le MT.
- Le service support de l'UMTS (*UMTS Bearer Service*). C'est le service qui offre la qualité de service de l'UMTS. Ce service utilise le service support d'accès radio et le service support du réseau cœur CN (*Core Network*).
- Le service support externe (*External Bearer Service*). C'est un service offert par des réseaux externes qui peuvent être des réseaux UMTS ou autre.
- Le service support d'accès radio (*Radio Access Bearer Service*). Il assure le transport des données entre le MT et le nœud de bordure (*CN Iu Edge Node*). C'est le nœud de la frontière entre le CN et le réseau d'accès. Lors de l'établissement d'un service support d'accès radio, des paramètres sont communiqués au réseau d'accès UTRAN comme la taille et le format des paquets transportés. Ce service utilise le service support radio et le service support de l'interface Iu.
- Le service support du réseau cœur (*Core Network Bearer Service*). Il assure l'interconnexion entre le nœud de bordure (*CN Iu Edge Node*) et la passerelle entre le réseau cœur et les réseaux extérieurs (*CN Gateway*). Le rôle de ce service est d'utiliser le réseau dorsal (*Backbone*) pour fournir la qualité de service demandée.
- Le service support radio (*Radio Bearer Service*). Le rôle de ce service est de gérer tous les aspects liés au transport sur l'interface radio comme les fonctions de segmentation et de réassemblage. Il utilise les services fournis par l'UTRA FDD/TDD (*UMTS Terrestrial Radio Access*) en mode FDD (*Frequency Division Duplex*) et en mode TDD (*Time Division Duplex*). Ce service ne fait pas partie de l'étude de notre thèse.

- Le service support de l'interface Iu (*Iu-Bearer Service*). Il agit avec le service support physique (*Physical Bearer Service*) pour assurer le transport entre le réseau d'accès UTRAN et le réseau cœur CN.
- Le service support du réseau dorsal (*Backbone Bearer Service*). Il gère les fonctionnalités des couches 1 et 2 pour assurer les besoins de la qualité de service du réseau cœur CN. Ce service n'est pas spécifique pour l'UMTS, mais il peut réutiliser les spécifications existantes.

5.2.2 Les classes de QoS de l'UMTS

Les spécifications du 3GPP définissent quatre classes de qualité de service (QoS) pour le transport des applications multimédia dans l'UMTS. La différence majeure entre ces classes de QoS est leur sensibilité au délai de transfert. Ces quatre classes sont :

5.2.2.1 La classe "Conversational"

Le meilleur exemple d'utilisation de cette classe est la téléphonie. Elle peut être aussi utilisée pour les nouvelles applications Internet à aspect conversationnel et temps-réel comme la voix sur IP (VoIP). Cette classe exige des contraintes strictes sur le délai de transfert des paquets ainsi que sur la variation du délai de transfert (la gigue).

5.2.2.2 La classe "Streaming"

Cette classe est utilisée pour les flux unidirectionnels comme les applications de diffusion vidéo ou audio. Il n'existe pas des contraintes strictes sur le délai de transfert pour les applications *streaming*. Par contre, la variation du délai est un paramètre important parce que cette variation est perceptible par l'utilisateur. Toutefois, cette contrainte sur la variation du délai reste tolérante grâce aux tampons du récepteur qui peuvent amortir les variations du délai si elles sont toujours inférieures à une limite donnée.

5.2.2.3 La classe "Interactive"

Cette classe est utilisée pour les applications qui nécessitent une interaction entre les deux extrémités de la communication. Un exemple d'application de cette classe est le *web browsing*. Cette classe est de type transactionnel. Elle nécessite une certaine contrainte sur le délai de transfert des paquets parce que l'utilisateur attend une réponse dans une certaine limite de temps. Cette contrainte n'est pas stricte puisque ce sont des applications non temps-réel. En revanche, cette classe doit assurer un taux de perte des paquets assez faible parce que les applications transportées par cette classe sont très sensibles aux pertes.

5.2.2.4 La classe "Background"

C'est la classe la moins exigeante en terme de délai de transfert. Les applications transportées par cette classe sont des applications dont l'utilisateur n'attend pas les paquets dans une certaine limite de temps. La contrainte la plus importante est le taux de perte. Cette classe est très sensible aux pertes des paquets. Les applications E-mails et SMS constituent des exemples de la classe *background*.

5.2.3 Le service de transport de l'UTRAN

Comme on peut le remarquer dans le paragraphe 5.2.1, l'architecture globale de la qualité de service dans l'UMTS est décomposée en couches indépendantes. Chaque couche a pour rôle d'assurer des services pour les fournir à la couche supérieure. Le transport dans le réseau UTRAN doit offrir la qualité de service requise par les services supports des niveaux supérieurs. Cette qualité de service est

assurée par le service de transport de l'UTRAN (*UTRAN Transport Service*) qui est représenté dans l'architecture générale de la QoS comme le montre la figure 5.2. [101]

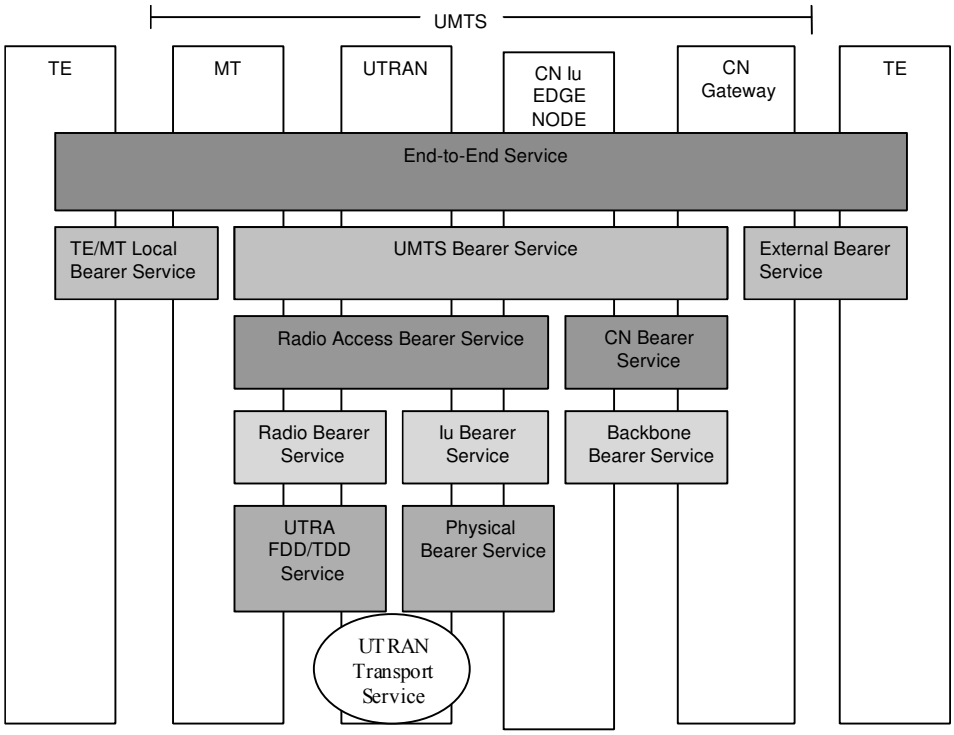


Figure 5.2 La position du service de transport de l'UTRAN dans l'architecture générale de la QOS

Notre étude se focalise sur le transport des flux dans l'UTRAN et spécialement sur les interfaces Iub et Iur. Le protocole de transport déployé au sein de l'UTRAN, que ce soit l'AAL2/ATM (Release 99) ou l'IP (Release 5), doit garantir une certaine qualité de service pour les flux transportés. Ce niveau de transport doit aussi fournir des classes de service de transport qui seront utilisées par les classes de service de l'UMTS présentées dans le paragraphe 5.2.2.

5.3 La qualité de service sur les interfaces Iub et Iur

La qualité de service sur les interfaces Iub et Iur de l'UTRAN sera fournie par les protocoles de transport déployés sur ces interfaces. Dans ce chapitre et celui d'après, nous allons étudier le protocole AAL2 et la qualité de service qu'il pourra fournir pour le transport des informations ainsi que les performances de ce protocole dans le contexte de l'UTRAN. Le transport en IP sera étudié dans le chapitre 7.

Le schéma de la figure 5.3 représente l'interface Iub entre un Node B et un RNC. Cette interface est constituée d'un sous-réseau de transport basé sur le protocole AAL2/ATM. Le *Last Mile Link* est le lien physique qui relie le Node B au nœud le plus proche du réseau. Les canaux radio, étendus entre le Node B et le RNC, sont transportés par le protocole AAL2/ATM dans le sous-réseau de transport.

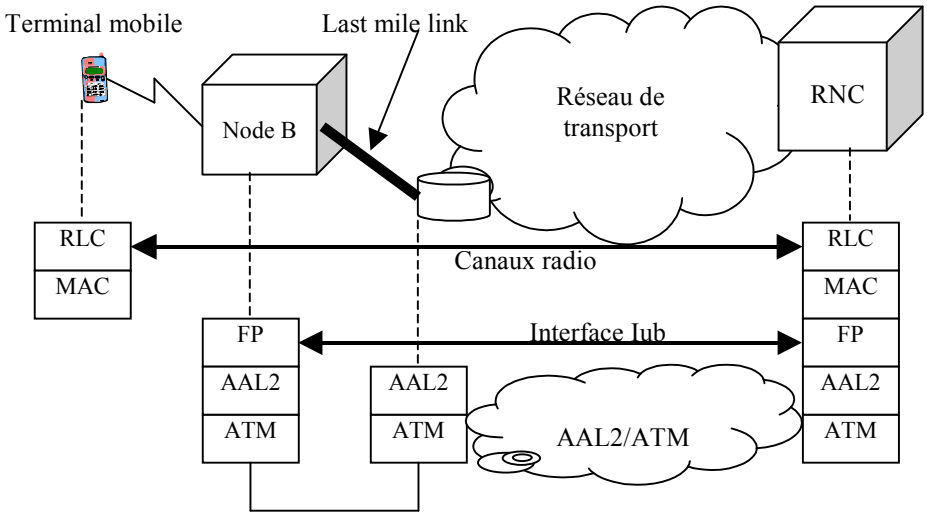


Figure 5.3 Le réseau de transport de l'interface Iub

Les flux transportés dans l'UTRAN ont des contraintes temporelles strictes à cause des mécanismes de synchronisation des couches radio [107]. D'ailleurs, on peut toujours faire une distinction entre les flux temps-réel et les flux non temps-réel malgré l'aspect plus ou moins temps-réel de tout les flux transportés. En effet, quand les couches supérieures décident d'envoyer un paquet radio sur les canaux de transport, une date d'émission ou un numéro de trame est affecté à ce paquet. Dans le sens descendant par exemple, le paquet doit alors arriver au Node B et être envoyé sur l'interface radio dans la trame correspondante, sinon il sera considéré comme perdu. Nous pouvons utiliser cette caractéristique pour permettre de retarder les paquets non temps-réel et par suite obtenir une souplesse dans la gestion des flux au niveau des protocoles de transport. Le paquet peut être envoyé sur l'interface Iub (ou Iur) à un instant donné mais l'instant de son émission sur l'interface radio peut être retardé ce qui permet à ce paquet de passer plus de temps dans le réseau de transport.

Un autre point de distinction entre les flux est possible sur la base du taux de perte. En effet, les flux temps-réel sont exigeants en terme de délai mais peuvent tolérer un certain taux de perte. Par exemple, la perte de quelques paquets d'un flux vidéo ne dégrade pas la qualité de l'image transmise. Dans le cas d'un taux de perte acceptable, la retransmission des données n'est pas nécessaire. Par contre, un flux de données (*web browsing* par exemple) est très sensible aux pertes. Un paquet perdu nécessite une retransmission au niveau de la couche RLC ce qui peut engendrer, d'une façon imprévue, une augmentation du trafic sur les interfaces Iub et Iur ainsi que sur l'interface radio. En fait, dans le cas de l'UTRAN, le protocole AAL2 n'utilise pas la sous-couche SSCS-ADT pour la retransmission des paquets perdus. La retransmission sera assurée uniquement par la couche RLC du réseau RNL. Donc, une mini-cellule AAL2 perdue entraîne la retransmission de tout le paquet RLC qui contient cette mini-cellule. Il est alors important de minimiser le taux de perte des flux sensibles aux pertes des données. On peut alors séparer les flux à la base de leur tolérance aux pertes.

5.3.1 Schéma d'association entre les classes des différents niveaux

D'après ce qui précède, il est alors possible de distinguer entre les différents flux transportés dans l'UTRAN en terme des besoins de qualité de service, même au niveau du transport. Par conséquent, nous pouvons transporter chaque type de flux sur une classe de service différente du niveau AAL2 et par suite, la qualité de service requise pour chaque classe sera fournie par les couches AAL2 et ATM. Les classes de service de l'AAL2 définies dans le chapitre 4 seront alors utilisées pour transporter les différentes classes de l'UMTS (*conversational, streaming, interactive, background*) selon leurs besoins de qualité de service. Une association (*mapping*) entre les classes de services de l'UMTS est celles du niveau AAL2 est alors nécessaire.

La classe *Conversational* de l'UMTS est utilisée pour les applications qui nécessitent une interaction temps-réel entre les deux extrémités de la communication comme les conversations téléphoniques, la voix sur IP et la vidéo-conférence. Cette classe est de type temps-réel, c'est à dire les exigences temporelles sont strictes. Elle doit être alors associée à la classe SRT de l'AAL2.

La classe *Streaming* est utilisée pour transporter des applications de diffusion unidirectionnelles et qui n'ont pas une nature conversationnelle. Des exemples d'application de cette classe sont la diffusion de la vidéo (*video-streaming* : émissions télévisées, films, bandes d'annonce, etc.) et la diffusion de la voix (*radio broadcasting* : chaînes radio, service météo, service *news*, etc.). Ce type d'application tolère des délais de transfert plus grands que la classe conversationnelle. En plus, ces applications sont plus tolérantes à la variation du délai. En effet, on peut mémoriser les paquets reçus dans les tampons du récepteur avant de les envoyer à leur destination finale pour amortir l'effet de la variation du délai. Cette mémorisation peut entraîner un délai supplémentaire, mais puisque ces flux sont tolérants au délai, alors cela ne dégrade pas la qualité de service des applications transportées. Cette classe peut être transportée par la classe TRT du niveau AAL2.

La classe *Interactive* de l'UMTS est utilisée pour les applications non temps-réel mais qui ont un aspect interactif comme le *web browsing*. Ces applications sont très sensibles aux taux de perte mais ne sont pas sensibles au délai. Mais il est toujours nécessaire que le délai soit borné pour que l'utilisateur d'une extrémité de la connexion n'attende pas un temps trop long avant la réception de l'information demandée. De toute façon, les paquets de cette classe acquièrent un aspect plus ou moins temps-réel lors de leur passage sur les interfaces Iub et Iur à cause des mécanismes de synchronisation de l'UTRAN. Les paquets de cette classe peuvent alors tolérer un certain seuil de délai mais demandent un taux de perte faible. Cette classe peut être transportée par la classe TRT de l'AAL2.

La classe *Background* tolère des délais élevés mais la contrainte du taux de perte est très stricte. Mais à cause des mécanismes de synchronisation de l'UTRAN, les paquets de cette classe deviennent sensibles au délai de transfert sur les interfaces Iub et Iur. Alors cette classe doit être supportée par la classe TRT qui garantit une borne supérieure du délai de transfert dans le réseau de transport.

Les classes NRT et BE du niveau AAL2 ne sont pas utilisées dans le contexte de l'UTRAN parce qu'elles ne peuvent pas satisfaire les contraintes de l'UTRAN. Alors seulement deux grandes catégories sont utilisées dans l'UTRAN : SRT et TRT.

Au niveau ATM, la classe 1 (*Stringent class*) est utilisée pour transporter les deux classes de l'AAL2 dans un VC. Il est toujours possible d'utiliser la classe 2 de l'ATM (*Tolerant class*) s'il n'y a pas des flux AAL2 de classe SRT dans le VC ATM.

La figure 5.4 représente le schéma d'association entre les classes des différents niveaux.

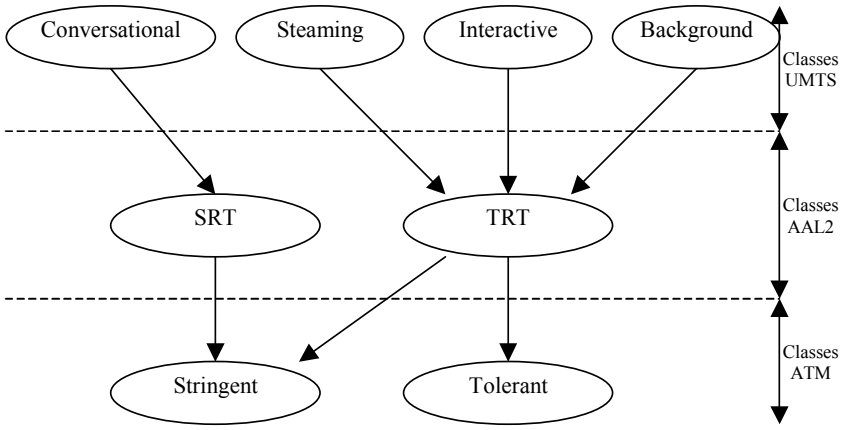


Figure 5.4 *Mapping* entre les classes des différents niveaux

5.3.2 Association entre les applications et les capacités de transfert AAL2

Il faut aussi prendre en compte la problématique d'allocation des ressources. En effet, le réseau de transport ne doit pas constituer le goulot d'étranglement (*bottleneck*) une fois que le trafic a trouvé les ressources radio nécessaires. Puisque la majorité des flux transportés dans l'UTRAN sont à débit variable, qu'ils soient temps-réel ou non temps-réel, l'allocation des ressources ne peut pas être faite à la base du débit crête pour ne pas gaspiller les ressources disponibles. Pour bien utiliser les ressources de ce sous-réseau, l'allocation des ressources pourra être faite en se basant sur le descripteur de trafic des classes de service des niveaux AAL2 et ATM. Il faut alors établir un schéma d'association entre les différentes applications et les capacités de transfert.

Il n'y a pas une association spécifique entre les classes de l'UMTS et les capacités de transfert aux niveaux AAL2 et ATM. En effet, cette association dépend du profil du trafic de l'application transportée. Par exemple, pour une même classe, on peut trouver des applications à débit constant et d'autres à débit très variable. Lors de l'établissement de la connexion, l'utilisateur communique au réseau la classe et la capacité de transfert demandée sous forme d'un contrat de trafic.

Les applications de la classe conversationnelle peuvent être à débit variable comme les flux AMR où des mécanismes de suppression du silence sont mis en place. Ces applications peuvent alors être transportées par la capacité de transfert AAL-VBR. Dans ce cas, l'allocation des ressources sera faites à la base du débit moyen et de la longueur de rafale. Une sur-allocation des ressources est appliquée en espérant pouvoir profiter du gain statistique du réseau. Il est aussi possible que les applications de cette classe soient à débit constant comme par exemple la vidéo-conférence avec des codeurs MPEG-4 ou H.263 (ce sont les codeurs proposés pour la compression de la vidéo dans l'UMTS). Dans ce cas, le débit à la sortie du codeur est relativement constant et l'application peut être transportée par l'AAL2-CBR. Dans ce cas, l'allocation des ressources sera faite à la base du débit crête et aucune sur-allocation n'est faite. La bande passante totale nécessaire est alors la somme des débits crêtes des différents flux transportés.

Les applications de la classe *Streaming* peuvent aussi avoir des profils à débit constant comme la vidéo-*streaming* compressée ou à débit variable comme la diffusion de la voix avec suppression du silence. Dans ce cas, on peut transporter les applications de cette classe sur l'AAL2-CBR ou sur l'AAL2-VBR selon le profil de l'application.

Les applications des classes *Interactive* et *Background* sont généralement à débit variable et seront transportées par la capacité de transfert AAL2-VBR. Il est toujours possible de transporter quelques applications sur l'AAL2-CBR en allouant les ressources pour une courte durée qui est la durée de transfert des fichiers (FTP par exemple).

La capacité de transfert AAL2-ABR ne peut pas être utilisée dans l'UTRAN parce qu'elle ne peut pas garantir un débit minimum. En fait, elle utilise la bande passante résiduelle et ce cas ne convient pas au transport des flux dans l'UTRAN.

5.3.3 Association entre les capacités de transfert des niveaux AAL2 et ATM

Pour la capacité de transfert AAL2-CBR, l'allocation des ressources est faite à la base du débit crête des connexions AAL2. Dans un VC ATM, la bande passante réservée pour un faisceau de connexions AAL2 est égale à la somme des débits crêtes au niveau ATM des différentes connexions. Par conséquent, au niveau ATM, la bande passante doit être réservée à la base de cette somme qui est un débit constant. L'ATC (*ATM Transfert Capability*) utilisée au niveau ATM doit être alors le DBR. Le PCR du VC ATM et alors lié aux PMBR des différentes connexions AAL2 par la relation:

$$PCR=\frac{53}{47}\times\sum_{i=1}^{N}PMBR_i \text{ (octet/s)}$$

Équation 5.1

où N est le nombre des connexions AAL2 dans le VC et $PMBR_i$ est le débit "mini-cellulaire" crête de la connexion AAL2 numéro "i". Le facteur $\frac{53}{47}$ représente les *overheads* qui ne sont pas comptés parmi la bande passante offerte à la couche AAL2. En effet, la bande passante disponible au niveau de la couche AAL2 est représentée par les octets de charge utile dans chaque cellule ATM qui sont au nombre de 47 (sans compter l'octet du champ STF).

Dans cette équation, nous considérons que le taux de remplissage est maximal (100%). C'est le cas d'un Timer-CU infini ou d'un VC fortement chargé où le taux de remplissage peut atteindre sa valeur maximale. En fait, si le VC est faiblement chargé, des cellules ATM peuvent être envoyées après l'expiration du Timer-CU (si sa valeur est faible) et par suite, elles sont complétées par des octets de bourrage. Cette bande passante supplémentaire due au bourrage n'est pas considérée dans l'équation 5.1. A faible charge, le débit crête au niveau ATM d'une connexion AAL2 peut être plus grand que cette valeur calculée. Mais puisque les mécanismes de réservation des ressources et de contrôle d'admission sont critiques dans le cas d'un VC à forte charge, alors cette hypothèse reste justifiable quelle que soit la valeur du Timer-CU.

Les connexions AAL2 dont la capacité de transfert est l'AAL2-VBR peuvent être supportées par l'ATC DBR au niveau ATM. Dans ce cas, le PCR du VC ATM est déduit des débits crêtes des différentes connexions AAL2 par l'équation 5.1. L'avantage de ce *mapping* est sa simplicité mais son inconvénient est la bande passante non utilisée. En effet, la bande passante réservée dans le VC est équivalente à la somme des débits crêtes mais les sources VBR n'émettent pas toujours en même temps à ce débit. Pour remédier à ce problème, les connexions AAL2-VBR sont transportées sur l'ATC SBR au niveau de la couche ATM. Dans ce cas, le PCR du VC ATM est calculé suivant l'équation 5.1. Le SCR ou le débit moyen du VC SBR peut être aussi calculé de la même manière en fonction des débits moyens MMBR des connexions AAL2. La longueur d'une rafale tolérée pour le VC SBR peut être déduite des rafales tolérées pour les connexions AAL2-VBR. En effet, dans un VC ATM transportant des flux à débits variables, le débit total atteint le débit crête quand tous les utilisateurs émettent à débits crêtes, c'est à dire durant leurs rafales. Alors la durée d'une rafale au niveau ATM est équivalente à la durée de la plus petite rafale des connexions AAL2 du VC.

5.4 Gestion des flux dans l'UTRAN

Comme nous l'avons déjà expliqué dans les paragraphes précédents, les flux dans l'UTRAN peuvent se décomposer en deux grandes catégories en fonction de leurs besoins de qualité de service: les flux à contraintes temporelles très strictes (classe SRT) et les flux tolérants au délai (classe TRT). En partant de ce schéma de distinction entre les flux, nous proposons trois schémas pour le multiplexage des connexions AAL2 dans les VC ATM. Un schéma d'allocation des ressources est alors nécessaire selon le mode d'agrégation des flux.

5.4.1 Schémas d'agrégation des flux AAL2

5.4.1.1 Schéma 1 : VC mono-service homogène

Le premier schéma consiste à agréger tous les flux homogènes et de même classe dans un même VC au niveau de la couche ATM. Deux flux sont dits homogènes s'ils ont le même profil de trafic.

Dans ce cas, on peut par exemple agréger tous les flux de voix AMR dans un même VC, tous les flux de vidéo dans un autre, tous les flux de *web browsing* dans un troisième, et ainsi de suite. L'avantage de cette solution est qu'on peut obtenir des flux homogènes dans un même VC ce qui facilite le traitement au niveau AAL2. En fait, au niveau AAL2, des mécanismes d'ordonnancement entre les flux peuvent être implémentés. Dans ce cas, et puisque les flux sont homogènes, alors ils doivent être traités avec la même priorité. Des mécanismes d'ordonnancement simples comme FIFO (*First In First Out*) ou RR (*Round Robin*) seront suffisants. L'inconvénient de ce schéma est qu'on a besoin d'un VC pour chaque type d'application. Dans ce cas, le schéma de partage de la bande passante ne sera pas très efficace. Dans ce schéma, les VC sont appelés mono-services homogènes. La différenciation des différentes classes sera faite au niveau ATM. Des mécanismes d'ordonnancement plus sophistiqués tenant compte des contraintes temporelles et des besoins de bande passante sont alors nécessaires au niveau ATM, c'est à dire entre les VC. Les mécanismes proposés sont WRR (*Weighted Round Robin*) et EDF (*Earliest Deadline First*). Nous allons voir ces mécanismes en détail dans le chapitre suivant.

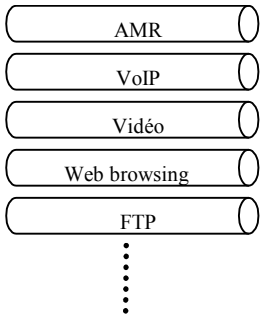


Figure 5.5 Schéma 1 : VC mono-service homogène

5.4.1.2 Schéma 2 : VC mono-service hétérogène

Dans ce schéma, les flux de même classe de QoS sont agrégés dans un même VC ATM. Par exemple, tous les flux de la classe SRT, que ce soit de la voix ou de la vidéo, sont agrégés dans le même VC. Ce schéma réduit le nombre de VC parce qu'on a besoin d'un VC par classe. Dans le cas de l'UTRAN, deux VC sont alors suffisants. Ces VC sont appelés mono-services parce qu'ils transportent des flux d'une même classe de service. Dans ce schéma, des flux à débit constant et d'autre à débit variable peuvent être agrégés dans le même VC. Plusieurs AAL2-TC peuvent alors être transportées dans le même VC ce qui augmente la complexité du traitement. De toute manière, si des AAL2-CBR et des AAL2-VBR sont agrégées dans un même VC, l'ATC au niveau ATM doit être le DBR à cause de la présence de l'AAL2-CBR. Les besoins des différents types de flux en terme de bande passante ne sont pas les mêmes, ce qui nécessite l'implémentation des mécanismes d'ordonnancement au niveau AAL2 pour partager la bande passante disponible entre les différents flux. Des mécanismes sophistiqués de type WFQ (*Weighted Fair Queueing*), WRR ou EDF peuvent être utilisés. Le VC transportant les flux AAL2 de classe SRT doit avoir une classe ATM de type *Stringent*. L'autre VC transportant les flux TRT peut avoir une classe ATM de type *Stringent* ou de type *Tolerant*. Alors, une différenciation des services au niveau ATM est nécessaire. Des mécanismes d'ordonnancement seront implémentés pour ordonnancer les cellules des différents VC ATM.

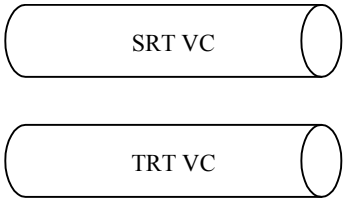


Figure 5.6 Schéma 2 : VC mono-service hétérogène

5.4.1.3 Schéma 3 : VC multi-service

Dans ce schéma, tous les flux de toutes les classes et quels que soient leurs profils de trafic sont agrégés dans un même VC ATM. Le VC est appelé multi-service. L'avantage de ce schéma et qu'on simplifie le schéma de partage de la bande passante. En effet, un seul VC est suffisant pour transporter tous les flux. En plus, ce schéma permet une utilisation efficace de la bande passante disponible puisqu'il n'y a pas un partage fixe préalable de cette bande passante. L'inconvénient de ce schéma est la complexité du traitement au niveau AAL2. En effet, les différentes classes sont multiplexées dans la même sous-couche CPS, sachant qu'elles ont des besoins différents en terme de qualité de service. Des mécanismes de priorité doivent être implémentés pour privilégier les flux SRT par rapport aux flux TRT. Cela peut se faire avec des mécanismes d'ordonnancement qui tiennent en compte les contraintes temporelles des paquets comme l'algorithme EDF par exemple. Puisque les profils des différents flux et leurs besoins en bande passante ne sont pas les mêmes, alors des mécanismes d'ordonnancement de type WFQ ou WRR sont aussi envisageables. Dans ce schéma, la classe de QoS au niveau ATM doit être la classe 1 (*Stringent*) et l'ATC est le DBR. Dans ce cas, le traitement au niveau ATM est simple puisqu'il n'y a qu'un seul type d'ATC et une seule classe.

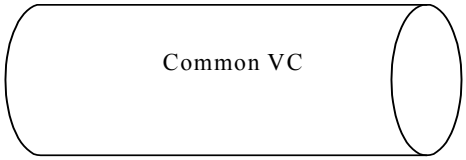


Figure 5.7 Schéma 3 : VC multi-service

5.4.1.4 Conclusion

Dans le premier schéma, le traitement au niveau AAL2 est simple et la différenciation des services est réalisée au niveau ATM. Le deuxième schéma est le plus compliqué parce qu'il nécessite un traitement complexe au niveau AAL2 et un autre traitement au niveau ATM. Le troisième schéma simplifie le traitement au niveau ATM et la couche AAL2 sera en charge de la différenciation des services.

5.4.2 Allocation des ressources et contrôle d'admission

Sur un lien entre deux nœuds du réseau UTRAN, les ressources disponibles doivent être partagées d'une manière efficace et selon le schéma d'agrégation choisi. Il faut prendre en compte les paramètres des capacités de transfert des niveaux AAL2 et ATM. Donc, il y a deux étapes pour l'allocation des ressources : la première pour les connexions AAL2 dans les VC ATM et la deuxième pour les VC dans les VP ATM.

L'allocation des ressources des connexions AAL2 individuelles est dynamique. Lors de l'établissement d'une nouvelle connexion, le réseau vérifie s'il possède les ressources nécessaires pour cette connexion. Dans le cas positif, il accepte la connexion et il réserve les ressources nécessaires. Dans le cas négatif, il refuse la connexion. Pour que le réseau accepte une nouvelle connexion, il faut d'abord que cette connexion lui envoie son descripteur de trafic avec la classe de QoS demandée. Pour les connexions AAL2 de capacité de transfert AAL2-CBR, le réseau vérifie si la bande passante disponible est supérieure au débit crête de la nouvelle connexion. Dans le cas positif, il l'accepte et réserve une bande passante égale à ce débit crête, dans le cas négatif il refuse la connexion ou il lui communique le débit crête possible de réserver et dans ce cas, l'utilisateur décide s'il accepte ou s'il refuse cette proposition. A la fin de la communication, toutes les ressources réservées sont libérées. Pour les connexions de type AAL2-VBR, le contrôle se fait sur la base du débit crête, du débit moyen

et de la longueur de rafale. De cette manière, on peut effectuer une fonction de contrôle d'admission des connexions AAL2.

Il existe une deuxième méthode pour la fonction de contrôle d'admission et d'allocation des ressources. Elle est basée sur la bande passante équivalente. Cette méthode est utilisée pour les VC de type DBR. La notion de la bande passante équivalente EBW (*Equivalent BandWidth*) peut être utilisée pour tous types de flux et elle est définie par le rapport entre le débit ATM total et le nombre des connexions AAL2 actives dans un VC fortement chargé. La fonction de contrôle d'admission CAC (*Connection Admission Control*) est définie comme suit : lors de l'établissement d'une nouvelle connexion AAL2, la fonction CAC vérifie si la somme des bandes passantes équivalentes de toutes les connexions actives, y compris la bande passante de la nouvelle connexion, est inférieure ou égale à la bande passante disponible B. Cette bande passante B constitue une partie du PCR du VC : $B = \alpha.PCR$; où α est un facteur qui est déterminé expérimentalement ou par simulation. Ce facteur dépend de la valeur du PCR et du type du trafic transporté dans le VC.

La bande passante équivalente représente le débit moyen à long terme au niveau ATM de la connexion considérée. Elle peut être calculée analytiquement dans certain cas. Par exemple, pour un VC mono-service homogène transportant des flux AMR, la bande passante équivalente est donnée par l'équation suivante:

$$EBW = \left(\left\lceil \frac{D \times TTI}{8} \right\rceil + O\_MAC + O\_FP + O\_CPS \right) \times \frac{8}{TTI} \times \frac{53}{47} \times \frac{ON}{ON+OFF} \qquad \textbf{Équation 5.2}$$

où:

D [bit/s] est le débit de la source en cas d'activité (période ON).

TTI : *Transmission Time Interval* [sec] pour le canal radio transportant ce flux.

O_MAC, O_FP et O_CPS représentent respectivement les tailles des en-têtes MAC, FP et CPS [en octet].

ON et OFF sont respectivement les durées moyennes des périodes ON et OFF de la source AMR [en sec].

Dans ce cas, on considère que toutes les cellules ATM sont pleines et qu'il n'y a pas des octets de bourrage dans la charge utile des cellules. Il n'est pas évident de calculer analytiquement les bandes passantes équivalentes de tous types de flux surtout lorsque le Timer-CU a une valeur faible et par suite, les cellules ATM sont remplies par des octets de bourrage. La valeur calculée analytiquement est la valeur minimale que peut avoir la bande passante équivalente dans le cas où le remplissage des cellules ATM serait parfait. La bande passante équivalente peut être mesurée expérimentalement ou par simulation dans différents contextes.

Cette méthode de contrôle d'admission et d'allocation des ressources nécessite la connaissance préalable de la valeur de la bande passante équivalente de chaque type de flux ainsi que la valeur du facteur α du VC. Dans le chapitre suivant, on va présenter quelques résultats de simulation pour la mesure de la bande passante équivalente ainsi que pour le facteur α.

Un schéma de partage des ressources semi-statique est réalisé par la couche de gestion du réseau. En effet, lors de la configuration du réseau, la bande passante disponible est partagée entre les différents ensembles de services. Ce partage peut être modifié si la couche de gestion du réseau trouve une nécessité. Ce cas peut arriver si par exemple on alloue une certaine bande passante à un type de trafic et après un certain temps, on trouve que le débit de ce trafic est beaucoup plus petit que la bande passante allouée. Dans ce cas, il faut reconfigurer le réseau. Ce partage de la bande passante se fait au niveau ATM. Au niveau AAL2, toute la bande passante disponible sera allouée au fur et à mesure aux

nouvelles connexions AAL2 après le contrôle d'admission. Donc, aucun schéma de partage fixe de la bande passante n'est effectué au niveau de l'AAL2.

Le partage au niveau ATM est important lorsqu'on a plusieurs types de VC. Dans ce cas, si on connaît préalablement la portion de chaque type de trafic, on peut établir des VC avec la bande passante correspondante d'une manière semi-statique pour réduire le temps d'établissement des connexions AAL2. En effet, pour établir une connexion AAL2, il faut d'abord établir le VC dans lequel cette connexion doit être transportée. Si le VC est déjà établi, on réduit la latence du temps d'établissement des connexions.

Dans le cas d'un VC mono-service homogène (schéma 1), on alloue une bande passante à chaque VC selon l'estimation du trafic qui sera acheminé dans ce VC. Dans le cas d'un VC mono-service hétérogène (schéma 2), la bande passante totale est partagée entre deux gros VC. Il est possible de créer plusieurs petits VC (de faible bande passante) pour la même classe de service, mais cette solution n'est pas très efficace en terme de gain statistique. Il est alors préférable de créer des VC avec la bande passante la plus élevée possible. Dans le cas du VC multi-service (schéma 3), la bande passante totale est allouée à l'unique VC.

5.5 Conclusion

Dans ce chapitre, nous avons présenté une méthode pour gérer les flux dans l'UTRAN ainsi que pour fournir la qualité de service nécessaire pour chaque type de flux en tenant compte des contraintes strictes de l'UTRAN. En se basant sur les propositions de ce chapitre, nous allons étudier le protocole AAL2 dans le contexte de l'UTRAN pour évaluer ses performances dans un contexte spécial qui est le transport des canaux radio. Cette étude sera réalisée en se basant sur des modèles de simulation et elle fera l'objet du chapitre suivant.

Chapitre 6

6 Les performances de l'AAL2 dans l'UTRAN

6.1 Introduction

La couche d'adaptation AAL2 constitue un vrai protocole de transport puisque les connexions AAL2 sont définies par leur identificateur CID. Ce protocole a des paramètres dont les valeurs ont une incidence directe sur ses performances et sur la qualité de service qu'il pourra fournir pour les applications transportées. Le Timer-CU est l'un des paramètres important d'où une étude de sa valeur optimale est nécessaire. En outre, comme nous l'avons déjà présenté dans les chapitres précédents, la différenciation des services peut avoir lieu au niveau AAL2 et dans ce cas le choix d'un mécanisme d'ordonnancement convenable est nécessaire. La commutation AAL2 doit être aussi étudiée pour évaluer ses avantages et ses inconvénients puisque le protocole AAL2 est relativement nouveau et il n'y a pas beaucoup d'études sur la commutation AAL2. Des études sont réalisées dans [19] mais elles ne prennent pas en compte tous les aspects de la qualité de service et des performances de l'AAL2. L'évaluation des performances du protocole AAL2 est alors nécessaire dans le contexte de l'UTRAN qui est le sujet de notre étude, surtout à cause des aspects spéciaux de cet environnement. Ce chapitre traite des aspects liés à la performance de l'AAL2 dans l'UTRAN comme ceux cités ci-dessus, ainsi que d'autres sujets concernant le dimensionnement des liens AAL2 de l'UTRAN qui peut servir pour le déploiement des premiers réseaux UTRAN.

6.2 Entre modélisation analytique et simulation

La modélisation analytique du protocole AAL2 est compliquée, voire très difficile. En fait, un modèle analytique est développé dans [8,9 et 10], mais il suppose que l'axe de temps est divisé en slots (système à temps discret) ce qui n'est pas le cas réel. En plus, ce modèle suppose plusieurs considérations simplificatrices : toutes les mini-cellules multiplexées ont la même taille, une cellule ATM contient un nombre entier de mini-cellules et par conséquent le chevauchement sur deux cellules consécutives n'est pas considéré. Ce modèle donne des bons résultats dans des cas spéciaux mais pour un modèle de réseau plus compliqué et plus général, ce modèle ne prend pas en compte certaines considérations importantes. Un modèle analytique complet qui prend en compte toutes les considérations et tous les détails d'un réseau complexe est difficile à manipuler avec les moyens mathématiques existants. En plus, dans les modèles analytiques, les valeurs calculées sont des valeurs moyennes avec des variances. Or, dans les spécifications du 3GPP, le critère temporel est le $95^{ème}$ percentile du délai (défini dans §4.3.1.1.1.1) et non pas le délai moyen parce que le $95^{ème}$ percentile donne une valeur plus représentative du délai de transfert. Ce délai ne peut pas être calculé par des modèles analytiques.

La simulation est alors un moyen sophistiqué qui recouvre les problèmes du modèle analytique pour évaluer les performances d'un réseau AAL2 complexe à condition de prendre en compte tous les aspects d'un réseau réel. Par programmation, nous pouvons toujours développer des modèles très complexes et très rigoureux en tenant compte des moindres détails du système réel. En plus, l'impact des couches radio sur les performances du protocole AAL2 peut être pris en compte dans le modèle de simulation. Ceci est très difficile analytiquement car un modèle analytique qui prend en compte tous

ces détails devient intraitable. L'inconvénient de la simulation est qu'elle est coûteuse en terme de temps de développement et de calcul. Notre étude est alors réalisée par simulation après avoir défini et développé un modèle de simulation convenable à l'environnement de l'UTRAN. Ce modèle est décrit dans le paragraphe §6.2.2.

6.2.1 Modélisation analytique

Dans ce paragraphe, nous allons présenter le modèle analytique proposé dans [10] pour démontrer la complexité de traitement d'un modèle précis. La sous-couche SSCS-SAR n'est pas considérée dans le modèle. La sous-couche CPS est composée de deux files d'attente : une pour l'assemblage des mini-cellules dans les cellules ATM et une autre pour la transmission des cellules sur le lien physique. La figure 6.1 représente le modèle considéré du multiplexeur CPS.

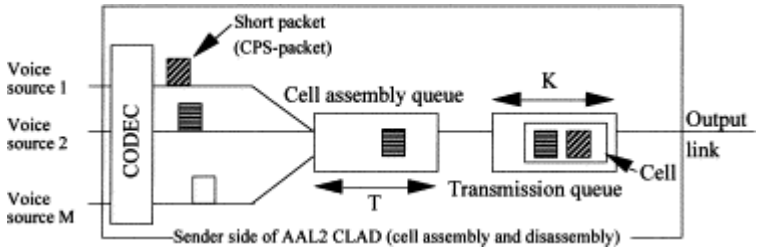


Figure 6.1 Modèle du multiplexeur CPS

Ce modèle est développé pour multiplexer des sources de voix. La période de silence n'est pas transmise. Pendant la période d'activité, les petits paquets (mini-cellules) arrivent dans la file d'attente d'assemblage et sont stockés selon une discipline de type FIFO (*First In First Out*). On suppose que la taille d'une mini-cellule est fixe. Une cellule ATM peut contenir un nombre maximal de mini-cellules noté "L". Un temporisateur est utilisé pour réduire le temps d'assemblage des cellules. Ce temporisateur correspond au Timer-CU. Si ce temporisateur expire avant le remplissage de la cellule, cette dernière est envoyée après remplissage par des octets de bourrage (*padding*). Après l'assemblage d'une cellule, elle sera envoyée vers la file d'attente de transmission (*buffer* de transmission). La taille de ce *buffer* est limitée et est notée K (en nombre de cellule). K inclut la cellule en cours de transmission. Les cellules sont transmises avec un débit fixe.

L'axe de temps est divisé en des intervalles fixes, les *timeslots* (système à temps discret). L'unité de temps est considérée égale au temps de transmission d'une cellule. Les mini-cellules peuvent arriver uniquement après le début de chaque *timeslot.* Les cellules pleines quittent la file d'attente d'assemblage juste après le début de chaque *timeslot*. Le temporisateur commence à zéro et il est incrémenté d'une unité au milieu de chaque *timeslot*. Sa valeur maximale est de T unités de temps. Une cellule partiellement remplie quitte la file d'attente d'assemblage au début d'un *timeslot* si le *timer* a une valeur de T-1 et s'il n'y a pas des mini-cellules en attente. Toutes les cellules (pleines ou partiellement remplies) sont placées dans le *buffer* de transmission. Elles sont transmises sur le lien selon une discipline FIFO. La transmission d'une cellule est effectuée juste avant la fin d'un *timeslot*. La figure 6.2 représente le modèle à temps discret utilisé dans [10].

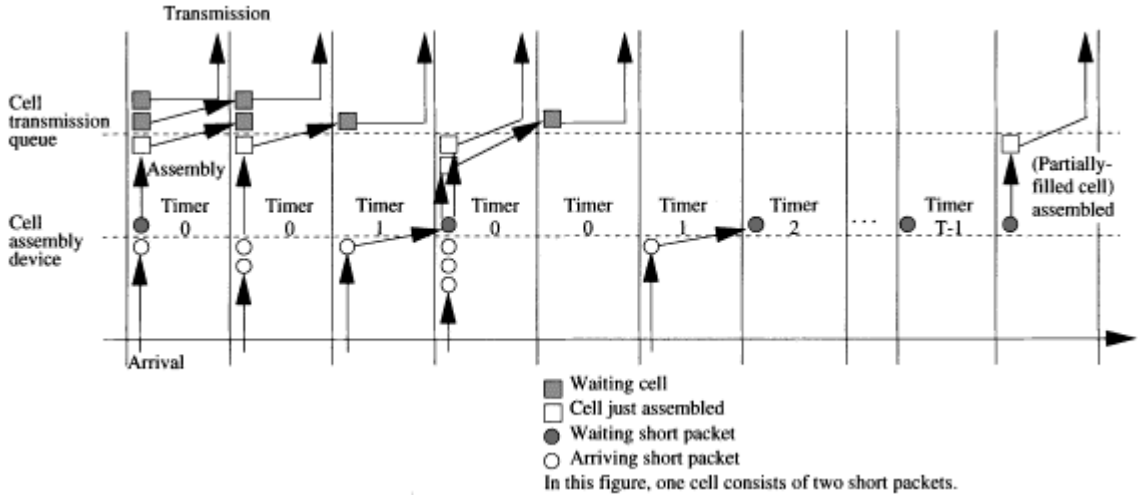


Figure 6.2 Modèle à temps discret

Lorsque n utilisateurs sont simultanément en période d'activité, les mini-cellules arrivent selon un processus de Poisson de moyenne nR/N. N est la taille d'une mini-cellule y compris l'en-tête AAL2. R est le débit de codage d'une source AMR (12,2 kbit/s par exemple). En d'autre terme, (N-3)/R est le temps moyen d'inter-arrivée de deux mini-cellules successives pour un même utilisateur et pendant sa période d'activité. "3" est la taille en octet de l'en-tête AAL2. Le processus de Poisson est une approximation pour l'arrivée des mini-cellules parce qu'elles sont en pratique générées avec des intervalles de temps réguliers. Si on veut prendre cette propriété déterministique dans la modélisation, le problème sera intraitable.

Supposons qu'on a M utilisateurs qui peuvent être en état de silence ou d'activité. Soit p_i (k , j) la probabilité pour que k mini-cellules soit générées à l'instant t et que le nombre d'utilisateurs en activité devienne j à l'instant t +1 sachant que le nombre d'utilisateurs en activité est i à l'instant t. Cette probabilité peut être représentée comme suit:

$$p_i(k,j)=\begin{cases} e^{-jR/N}\dfrac{(jR/N)^k}{k!}\displaystyle\sum_{m=0}^{Min(j,M-i)}\binom{i}{j-m}\alpha^{i-j+m}(1-\alpha)^{j-m}\binom{M-i}{m}\times\beta^m(1-\beta)^{M-i-m}\ ;\ i\geq j \\ e^{-jR/N}\dfrac{(jR/N)^k}{k!}\displaystyle\sum_{m=0}^{Min(i,M-j)}\binom{i}{m}\alpha^m(1-\alpha)^{i-m}\binom{M-i}{M-j-m}\times\beta^{j+m-i}(1-\beta)^{M-m-i};\ i<j \end{cases}$$

Le modèle d'arrivée est considéré comme un modèle Markovien à temps discret DMAP (*Discret time Model Arrival Process*). Ce modèle est composé de plusieurs états appelés "phases". La transition d'une phase à une autre suit une chaîne de Markov. Soit $d_i(k,j)$ la probabilité pour qu'il y ait k arrivées à l'instant t et que la phase à l'instant t+1 soit j sachant que la phase à l'instant t est i. Le générateur du processus DMAP est alors donné par $\{d_i(k,j)\}$. Soit la matrice $D_k=(d_i(k,j))_{i,j}$ avec k = 0,1,….Cette matrice représente la probabilité pour qu'il y ait k arrivées. Or, $\sum_{k=0}^{\infty}D_k e = e$ où e=(1,1,….,1)'. Dans ce modèle, le nombre d'utilisateurs en activité est considéré comme une phase, d'où:

$d_i(k,j) = p_i(k,j)$, où i,j = 0,1,…..,M et k = 0,1,…Dans ce modèle, le nombre de phase est de M+1.

Dans ce qui précède, nous avons présenté le modèle analytique d'un multiplexeur AAL2. L'analyse de ce modèle est décrite en détail dans [10] avec quelques résultats numériques. Cette analyse est "exacte" si on considère un modèle Markovien. En réalité, l'arrivée des mini-cellules suit une loi constante (D) et non pas une loi de Poisson.

Comme nous l'avons pu constater, ce modèle considère plusieurs hypothèses simplificatrices pour qu'il soit traitable. Il est un bon modèle pour des études comparatives avec d'autres modèles de réseau, mais il ne peut pas servir pour une étude fine et très rigoureuse surtout pour le dimensionnement des réseaux.

6.2.2 Modèle de simulation

Comme nous l'avons déjà mentionné plus haut, le modèle de simulation reste une bonne méthode fiable pour étudier un système avec une grande précision. Dans un tel modèle, on peut prendre en compte les détails du système réel autant que l'on veut. Pour notre étude, nous avons développé un modèle de simulation de l'UTRAN dans lequel les couches radio (RLC, MAC) ainsi que la couche FP sont implémentées avec quelques hypothèses simplificatrices. Par exemple, nous n'avons pas considéré les pertes sur le canal radio parce que ce sujet ne constitue pas le cœur de notre étude. La couche RLC n'assure pas la retransmission des paquets perdus ou erronés parce que ce sujet est un objet d'étude très large et n'entre pas dans le cadre de notre thèse.

La couche AAL2 avec les sous-couches SSCS-SAR et CPS sont implémentées d'une manière très fine et détaillée. Tous les détails décrits dans les automates de l'émetteur et du récepteur AAL2 sont pris en compte [99, 100]. La couche ATM est aussi implémentée avec le support de transmission. Tous les modèles de trafic décrits dans le chapitre 3 sont implémentés dans les générateurs de trafic de notre simulateur.

L'outil de simulation que nous avons utilisé s'appelle *Omnet++* [2]. C'est un logiciel libre en cours de développement basé sur un noyau écrit en *C++*. La définition des modèles de réseau est faite à l'aide d'un langage d'interface appelé NED (*NEtwork Descriptor*). *Omnet++* définit la notion d'un module qui consiste en une entité dans laquelle on peut définir les fonctionnalités que l'on veut. Un module peut être un nœud, un protocole, une couche, etc. *Omnet++* définit aussi la notion de message. C'est une entité mobile qui se déplace d'un module à un autre. Un message peut être considéré comme un paquet, une cellule, une mini-cellule ou une trame. Dans chaque message, on peut transporter des informations autant que l'on veut. Deux modules peuvent être reliés par une ou plusieurs liaisons. Une liaison est un lien d'interconnexion entre deux portes (*Gates*) de deux modules. Un module peut avoir plusieurs portes (jusqu'à 256). Les messages sont transportés entre les modules sur les liaisons établies. *Omnet++* est un simulateur à événements discrets dans lequel le temps simulé avance chaque fois qu'il y a un événement à réaliser. Un événement peut être l'envoie ou la réception d'un message dans un module ou l'expiration d'un temporisateur.

Les couches RLC, MAC, FP, AAL2 et ATM n'existent pas dans *Omnet++*. Nous avons alors créé des nouveaux modules dans lesquels nous avons développé les fonctionnalités de chaque protocole. Nous avons aussi défini deux nouveaux types de messages (les cellules et les mini-cellules) dans lesquels nous avons introduit les paramètres nécessaires comme les champs VPI et VCI de l'en-tête ATM et les champs CID, LI, STF de l'AAL2. Nous avons développé plusieurs modules pour créer des générateurs de trafic (AMR, WWW, etc.). Le trafic de chaque module est généré selon les modèles décrits dans le chapitre 3. Dans l'annexe C, nous présentons quelques exemples de code source des modules et protocoles que nous avons développés pour nos études.

Le modèle de réseau, écrit en langage NED, consiste en un Node B relié à un RNC par un VC ATM de type DBR. Les couches protocolaires de l'interface radio sont aussi implémentées avec les générateurs de trafic ainsi que la couche ATM. Le schéma de la figure 6.3 représente le modèle fonctionnel du simulateur. Ce modèle est valable dans le sens montant et dans le sens descendant. La

différence entre les deux sens réside dans le nombre d'octets d'*overheads* de la couche FP (2 octets de plus dans le sens montant).

Nous avons considéré trois types de VC : un VC mono-service homogène transportant du trafic de voix AMR, un VC mono-service homogène transportant du trafic de données (*web browsing*) et un VC multi-service transportant du trafic de voix AMR et des données (*web browsing*). D'autres types de VC peuvent être considérés mais dans ce cas, on aura un très grand nombre de scénarios à simuler. Nous avons donc choisi uniquement ces trois types pour plusieurs raisons. Tout d'abord, les flux AMR seront majoritaires dans l'UTRAN et ils sont très exigeants en terme de délai de transfert. En plus, ils ont une petite taille de paquet ce qui peut avoir une incidence sur le temps de remplissage des cellules ATM. Le cas d'un VC transportant uniquement du trafic AMR est alors critique. Les flux *web browsing* peuvent aussi constituer une partie importante du trafic de l'UTRAN. Ce trafic devient critique surtout lorsque les paquets sont de petites tailles comme dans le cas du mode UDD 64. Pour un VC multi-service, les flux des données ayant des paquets de grandes tailles peuvent aider à accélérer le remplissage des cellules ATM. Donc l'étude d'un VC multi-service est critique.

Pour le trafic de voix AMR, nous avons considéré le mode AMR 12,2. Le modèle de trafic et la taille des paquets sont donnés dans le chapitre 3. Deux modes UDD du trafic des données (*web browsing*) sont utilisés : le mode UDD 64 et le mode UDD 384. Le débit crête PCR du VC DBR est un paramètre qui sera donné pour chaque scénario. Le support physique est considéré comme un lien STM-1 (155,52 Mbit/s; 149,76 Mbit/s de débit utile). La taille des *Buffers* est supposée très grande pour que le taux de perte soit négligeable. En effet, le protocole de transport ne doit pas générer des taux de perte élevés parce qu'il n'y a pas des mécanismes de retransmission au niveau transport. La retransmission est effectuée au niveau de la couche radio RLC. Dans le cas de perte au niveau de transport, le trafic sur l'interface radio risque d'augmenter d'une manière considérable. Nous avons alors intérêt à minimiser les pertes au niveau transport. Notre hypothèse est justifiable parce que dans ce cas, le critère essentiel de qualité de service est le délai de transfert.

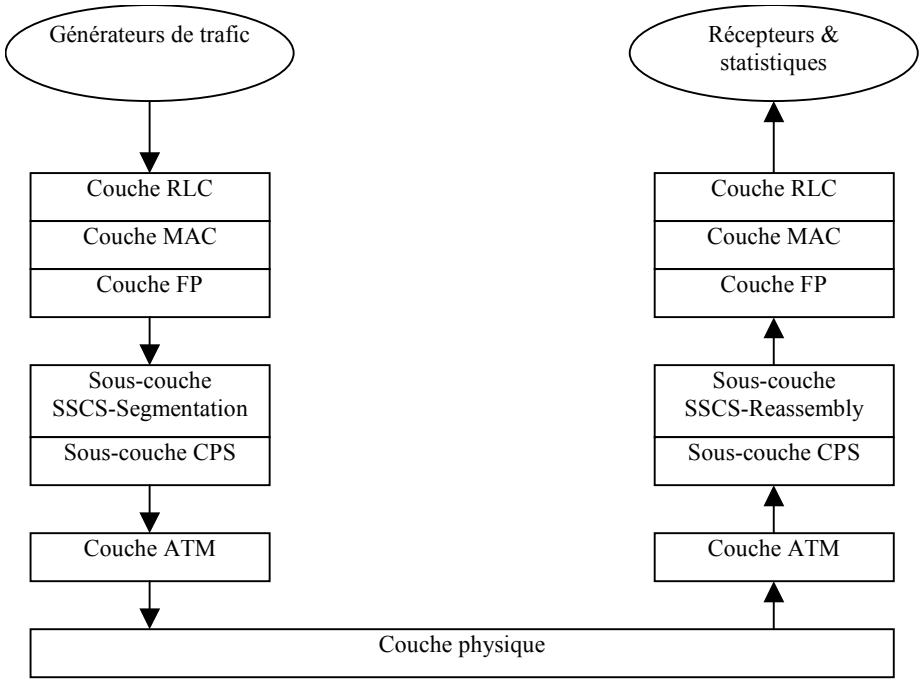


Figure 6.3 Modèle fonctionnel du simulateur

6.3 Le Timer-CU

Le temporisateur du protocole AAL2 (*Timer-CU: Timer-Combined Use*) est un paramètre très important. Il a une incidence directe sur les critères de qualité de service comme le délai et la gigue ainsi que sur les performances de l'AAL2 en terme d'efficacité de l'utilisation de la bande passante. En effet, la valeur du Timer-CU n'est pas spécifiée dans les recommandations de l'AAL2 et elle est laissée au choix des opérateurs des réseaux. Une valeur élevée du Timer-CU peut retarder les mini-cellules dans le multiplexeur CPS surtout quand ces mini-cellules ont des tailles plus petites que la taille de la charge utile d'une cellule ATM (47 octets sans l'octet STF). Dans ce cas, une mini-cellule encapsulée dans le paquet CPS-PDU doit attendre l'arrivée d'une nouvelle mini-cellule pour remplir le CPS-PDU. Ce dernier ne peut pas être envoyé vers la couche ATM s'il n'est pas plein ou si le Timer-CU n'a pas encore expiré. Si le trafic est faible, la mini-cellule encapsulée dans le CPS-PDU risque d'attendre très longtemps avant d'être envoyée vers la couche ATM. Ceci peut augmenter le délai de transfert des mini-cellules et par suite dégrader la qualité de service fournie par le protocole AAL2. Une faible valeur du Timer-CU peut résoudre ce problème de latence, mais cela peut générer des pertes de bande passante en terme d'octets de bourrage dans le CPS-PDU. En fait, si le Timer-CU expire avant que le CPS-PDU soit plein, ce dernier serait rempli par des octets de bourrage pour compléter sa charge utile. Ceci peut gaspiller la bande passante disponible sur l'interface Iub (ou Iur) qui est une ressource chère. Le choix d'une valeur optimale est alors nécessaire pour satisfaire les contraintes temporelles ainsi que les critères d'efficacité de la bande passante.

La vitesse de remplissage des CPS-PDU dépend de plusieurs facteurs comme la taille des paquets entrants dans la couche AAL2. En d'autre terme, elle dépend du type des flux transportés ainsi que du nombre des flux et de la charge du VC. Dans ce paragraphe, nous allons étudier par simulation l'impact de la valeur du Timer-CU sur les performances du protocole AAL2 et cela dans plusieurs contextes pour en tirer une conclusion sur la valeur optimale de ce paramètre.

6.3.1 Résultats et discussion

Les résultats des simulations sont les paramètres de performance les plus critiques. Nous présentons les résultats du taux de remplissage FR et du taux d'utilisation de la bande passante UR. Le taux de remplissage reflète l'impact de la valeur du Timer-CU sur l'efficacité de la bande passante. Le taux d'utilisation reflète les bénéfices en terme de bande passante qu'on peut gagner lorsque le remplissage des cellules est maximal. Les délais sont mesurés pour les trames FP-PDU, c'est à dire les paquets SSCS. Le 95ème percentile et le délai moyen du ASTD représentent le délai de transfert. La variation du délai ASDV est représentée sous la forme de l'écart-type (StdDev). L'ASDV n'est pas mesurée pour les paquets des données parce qu'ils ne sont pas sensibles à la variation du délai. Dans ce qui suit, nous allons présenter les résultats des scénarios ainsi que les commentaires nécessaires.

6.3.1.1 Scénario d'un VC mono-service de voix AMR 12,2

VC DBR avec PCR = 2 Mbit/s.

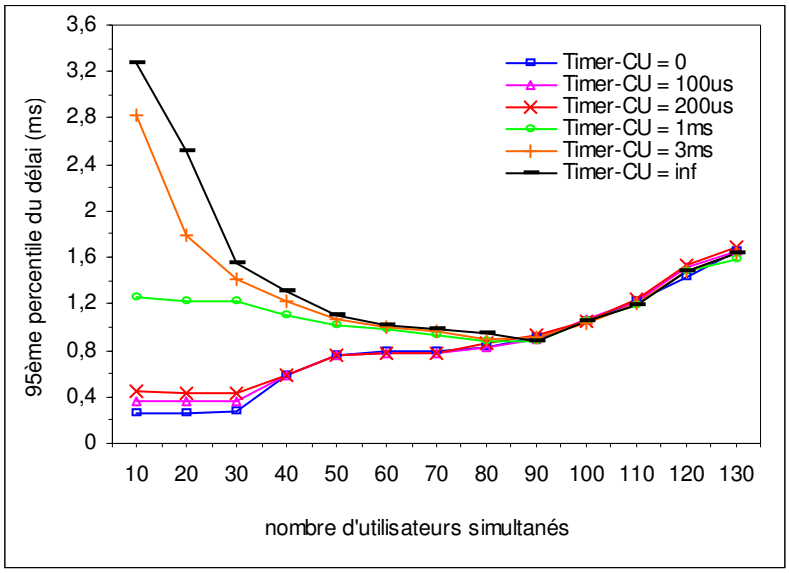


Figure 6.4 ASTD (95ème percentile), VC mono-service de voix

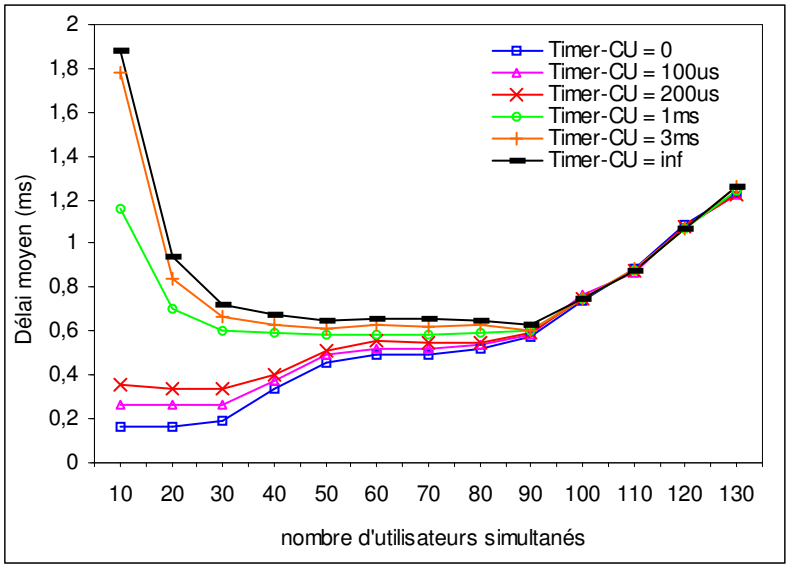


Figure 6.5 ASTD (délai moyen), VC mono-service de voix

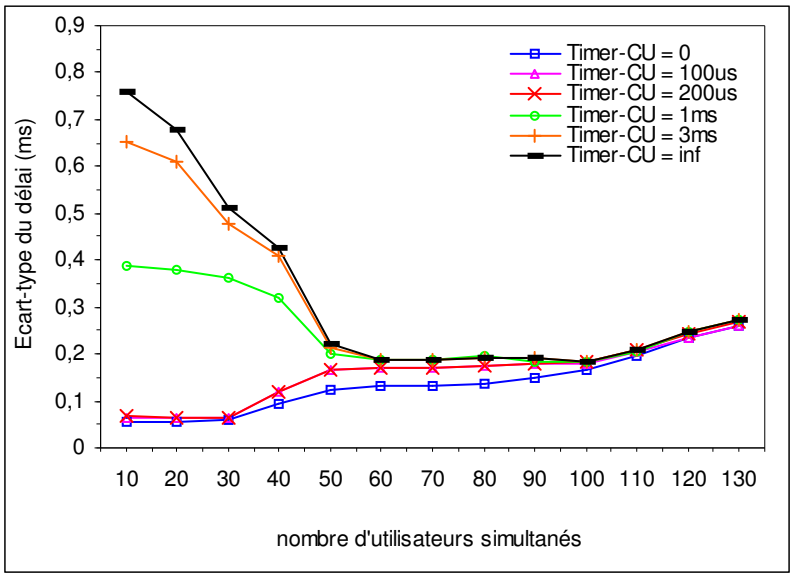


Figure 6.6 Ecart-Type du délai ASDV, VC mono-service de voix

Pour un nombre faible d'utilisateurs (faible charge du VC), il est clair sur les figures 6.4 et 6.5 que pour des valeurs élevées du Timer-CU, la composante la plus importante du délai de transfert est pratiquement le délai d'attente dans le multiplexeur CPS. Prenant par exemple le cas d'un Timer-CU de valeur infinie. Dans ce cas, le CPS-PDU n'est envoyé à la couche ATM que lorsqu'il est plein. Quand le nombre d'utilisateurs augmente, le débit des paquets augmente et par conséquent, le délai de multiplexage diminue considérablement. Par exemple, le 95ème percentile du délai diminue de 3.3 ms pour 10 utilisateurs jusqu'à 0,8 ms pour 90 utilisateurs. On remarque aussi que les courbes correspondantes à des faibles valeurs de Timer-CU (0, 100µs et 200µs) sont très proches les unes des autres. Le délai moyen, le 95ème percentile du délai et l'écart-type ont quasiment la même allure de la courbe. Pour une même charge (même nombre d'utilisateurs), le délai varie considérablement pour deux valeurs éloignées du Timer-CU. Pour un VC à très faible charge, le délai de multiplexage doit être limité par la valeur du Timer-CU. A partir d'un certain point, toutes les courbes convergent et les courbes du délai tendent vers des valeurs élevées. Sur cette partie des courbes, le délai est dû principalement au temps d'attente dans les files d'attente de la sous-couche CPS à cause de la forte charge du VC. Le temps de multiplexage devient négligeable.

Le taux de remplissage (*Filling Ratio : FR*) est représenté sur la figure 6.7. Pour une faible charge du VC, le taux de remplissage varie considérablement en fonction de la valeur du Timer-CU. Pour 10 utilisateurs par exemple, le taux de remplissage est de 62% pour une valeur de Timer-CU égale à zéro tandis qu'il est de 81% pour une valeur de 3ms de Timer-CU et de 100% pour un Timer-CU infini. Pour un taux de remplissage de 62% par exemple, il y a 38% de perte de la bande passante disponible au niveau AAL2. Cette perte est due aux octets de bourrage dans les cellules ATM. Pour une charge élevée du VC, toutes les courbes convergent pour atteindre une valeur maximale de taux de remplissage. Il est alors clair qu'une valeur optimale du Timer-CU doit être considérée comme compromis entre le délai et le taux de remplissage.

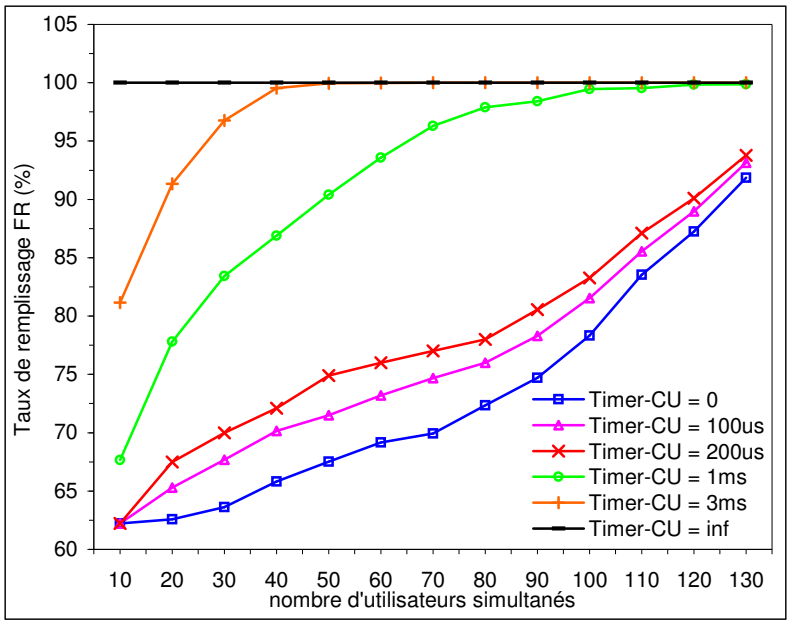


Figure 6.7 Taux de remplissage (FR), VC mono-service de voix

Ces résultats sont réalisés pour un VC de débit crête PCR égal à 2 Mbit/s. Pour affiner l'étude, plusieurs valeurs de PCR sont considérées. Puisque le Timer-CU a un impact considérable dans le cas d'un VC faiblement chargé, alors nous considérons des VC à faibles charges ($\rho = 0,1$) avec des PCR différents.

Pour cette étude, nous avons considéré des valeurs de Timer-CU qui sont en fonction de la période T du VC DBR. T est mesurée en seconde et elle est définie comme l'inverse du PCR (qui est mesuré en cellule/s) : T = 1 / PCR. Elle représente le temps minimal entre deux cellules successives. En effet, la couche ATM envoie une primitive vers la couche AAL2 toutes les T secondes. Cette primitive s'appelle "*MAAL-Send.request*". Le rôle de cette primitive est de donner l'autorisation à la couche CPS pour envoyer le paquet CPS-PDU en cours d'assemblage vers la couche ATM. Ensuite, le CPS-PDU sera encapsulé dans la cellule ATM et envoyé sur le support physique. Sans cette autorisation, le paquet CPS-PDU ne peut pas être envoyé même s'il est plein et si le Timer-CU a déjà expiré. Donc, cette primitive synchronise le débit cellulaire suivant le PCR du VC DBR. Différentes valeurs de Timer-CU sont considérées en fonction de T. Sur l'axe des abscisses des graphes, les nombres désignent les multiples de la période T du PCR considéré. Les figures 6.8, 6.9 et 6.10 représentent respectivement le délai moyen, le 95ème percentile du délai et le taux de remplissage FR.

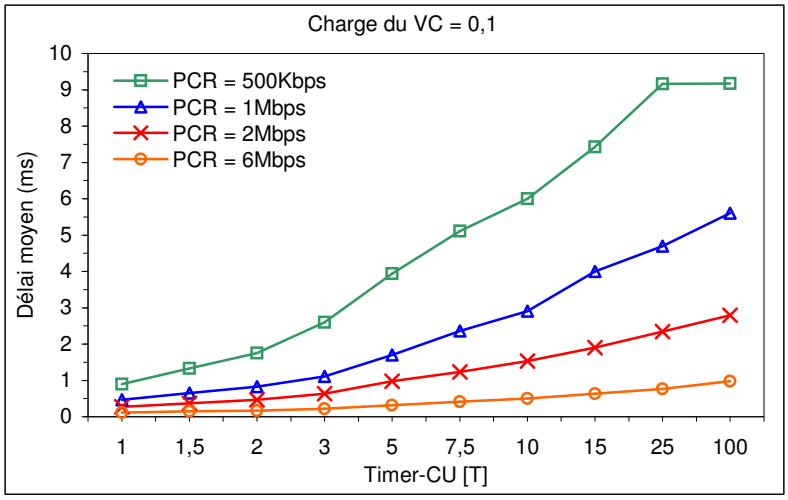


Figure 6.8 Délai moyen (ASTD), VC mono-service de voix, différentes valeurs de PCR

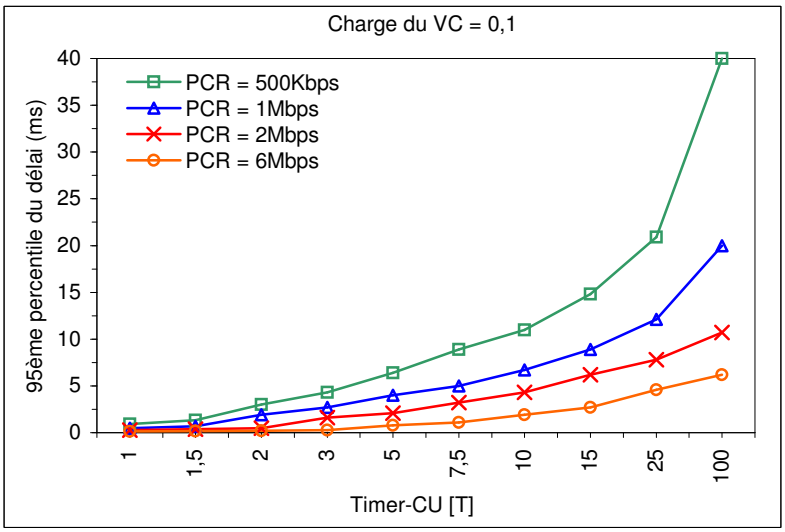


Figure 6.9 95ème percentile du délai (ASTD), VC mono-service de voix, différentes valeurs de PCR

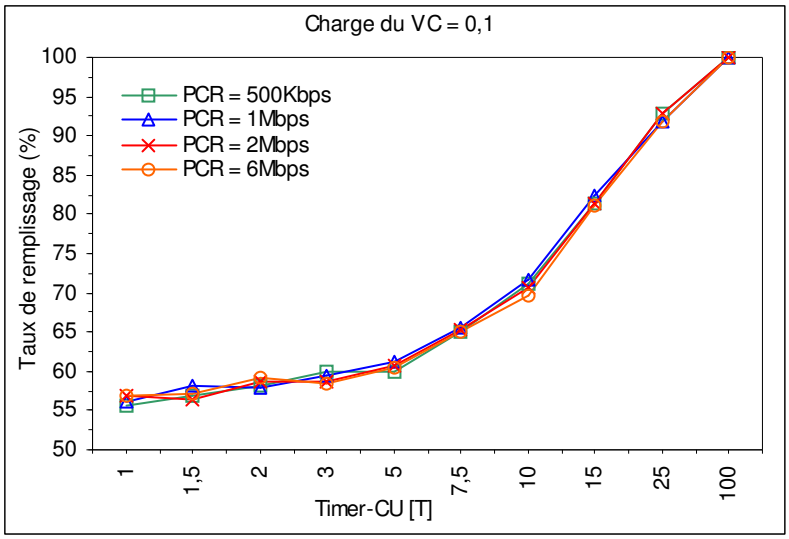


Figure 6.10 Taux de remplissage, VC mono-service de voix, différentes valeurs de PCR

On remarque sur la figure 6.10 que pour un même multiple de la période T correspondant au PCR de chaque VC, le taux de remplissage est le même pour tous les VC.

- *Interprétation* :

Soit deux VC dont les débits cellulaires crêtes sont respectivement PCR1 et PCR2 avec: PCR2 = ½.PCR1.

Les périodes respectives sont alors T1 et T2 avec : T2 = 2.T2.

Nous avons considéré des VC de même charge, alors les charges ρ1 = ρ2 avec :

ρ1 = D1/(53.PCR1) et ρ2 = D2/(53.PCR2); D1 et D2 sont les débits (en octet/s) respectifs des trafics entrant dans les deux VC.

Pour le VC1, le Timer-CU TCU1 est un multiple de T1, soit TCU1= n.T1.

Pour le VC2, le Timer-CU TCU2 est aussi un multiple de T2 avec le même cœfficient n: TCU2 = n.T2. (on prend deux points de même abscisse sur deux courbes).

Le nombre moyen d'octets arrivant dans le multiplexeur du VC1 pendant la période TCU1 est:

N1 = D1.TCU1 = 53.ρ1.PCR1.n.T1

et le nombre moyen d'octets arrivant dans VC2 pendant TCU2 est: N2 = 53.ρ2.PCR2.n.T2; donc: N2 = 53.ρ1.½.PCR1.n.2.T1 = 53.ρ1.PCR1.n.T1 = N1.

Puisque le VC est faiblement chargé, l'envoi du paquet CPS-PDU est dû à l'expiration du Timer-CU. Donc pour les deux VC, le nombre d'octets d'informations dans chaque CPS-PDU est égal au nombre d'octets arrivant pendant la durée du Timer-CU (N1 ou N2). Par suite, il est logique que le taux de remplissage soit le même pour des VC de même charge et dont le Timer-CU est le même multiple de la période T correspondante au PCR du VC.

Concernant le délai sur la figure 6.8 ou la figure 6.9, il dépend du débit du trafic et de la valeur du Timer-CU indépendamment de la période T. Pour deux VC DBR de débits crêtes PCR1 > PCR2 et pour une même abscisse (multiple de T), le TCU1 est inférieur à TCU2 (TCU1=n.T1, TCU2=n.T2 et T1<T2). Puisque la charge est faible, l'envoi du CPS-PCU est dû à l'expiration du Timer-CU dans la plupart des cas, alors il est normal que le délai pour VC1 soit inférieur au délai dans VC2. Pour des valeurs de Timer-CU très élevées (infinies), l'envoi du CPS-PDU est dû au remplissage du paquet. Alors dans le cas du VC1, le remplissage est plus rapide parce que le débit du trafic est plus élevé.

6.3.1.2 Scénario d'un VC mono-service de données UDD 64

Nous allons présenter l'impact du Timer-CU sur le trafic UDD 64 seulement parce que les paquets dans ce cas sont les plus petits parmi les 3 modes UDD utilisés. Pour les modes UDD 144 et UDD 384, les paquets sont plus longs et le remplissage est plus rapide. Donc le Timer-CU a une plus grande incidence dans le cas des données UDD 64.

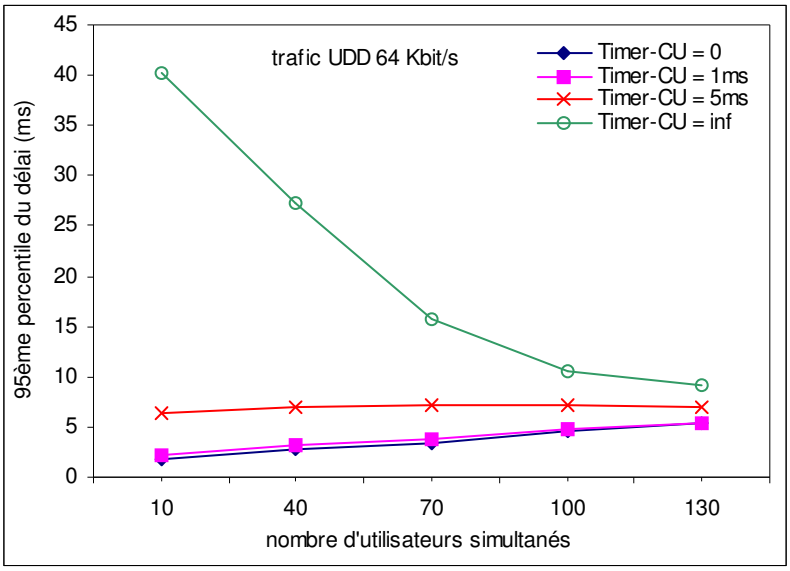


Figure 6.11 95ème percentile du délai (ASTD), VC mono-service de données UDD 64

Sur la figure 6.11, on remarque qu'à faible charge, le délai pour un Timer-CU infini est beaucoup plus grand que le délai pour un Timer-CU de faible valeur. C'est surtout à cause de l'aspect sporadique des sources UDD. En effet, si le dernier paquet d'un *packet-call* ne remplit pas le CPS-PDU, il doit attendre l'arrivée d'un nouveau *packet-call*, et puisque la valeur moyenne du *reading-time* est relativement élevée, ce paquet risque d'attendre très longtemps surtout lorsque le trafic est faible.

Le taux de remplissage est présenté sur la figure 6.12. On peut observer que sa valeur varie entre 97% et 100%, le FR est très proche de 100% et cela est dû à la taille relativement grande des paquets UDD 64. Donc le Timer-CU n'a pas une grande influence sur le taux de remplissage pour un VC mono-service de données. Pour choisir une valeur optimale du Timer-CU, il suffit de prendre en considération les critères de délai.

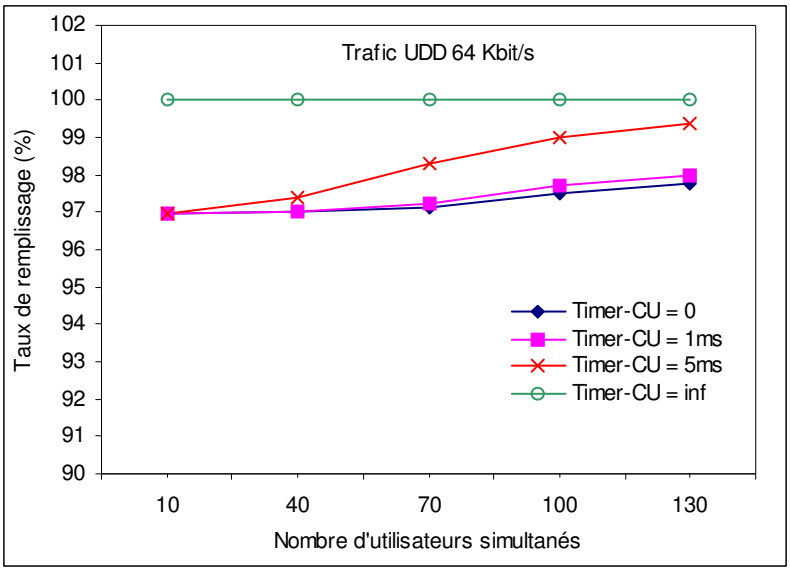


Figure 6.12 Taux de remplissage FR, VC mono-service de données UDD 64

6.3.1.3 Scénario d'un VC multi-service

Dans ce scénario, il y a plusieurs cas selon le pourcentage de chaque type de trafic et selon le mode UDD utilisé. La valeur du Timer-CU est considérée comme multiple de la période T du VC DBR. Plusieurs valeurs de PCR sont considérées. Dans le multiplexeur CPS, un ordonnanceur (*Scheduler*) est implémenté pour donner la priorité aux paquets de voix.

6.3.1.3.1 20% UDD 64 et 80% AMR 12,2

La charge du VC est de 15% environ.

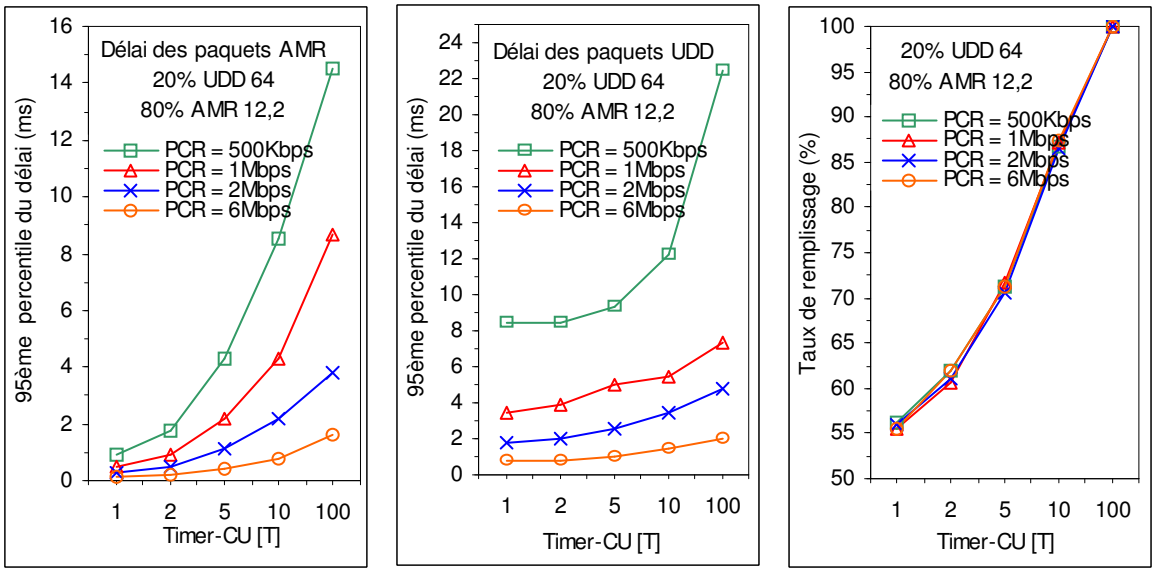


Figure 6.13 VC multi-service 20% UDD64 et 80% AMR12,2

6.3.1.3.2 20% UDD 384 et 80% AMR 12,2

La charge du VC est de 15% environ.

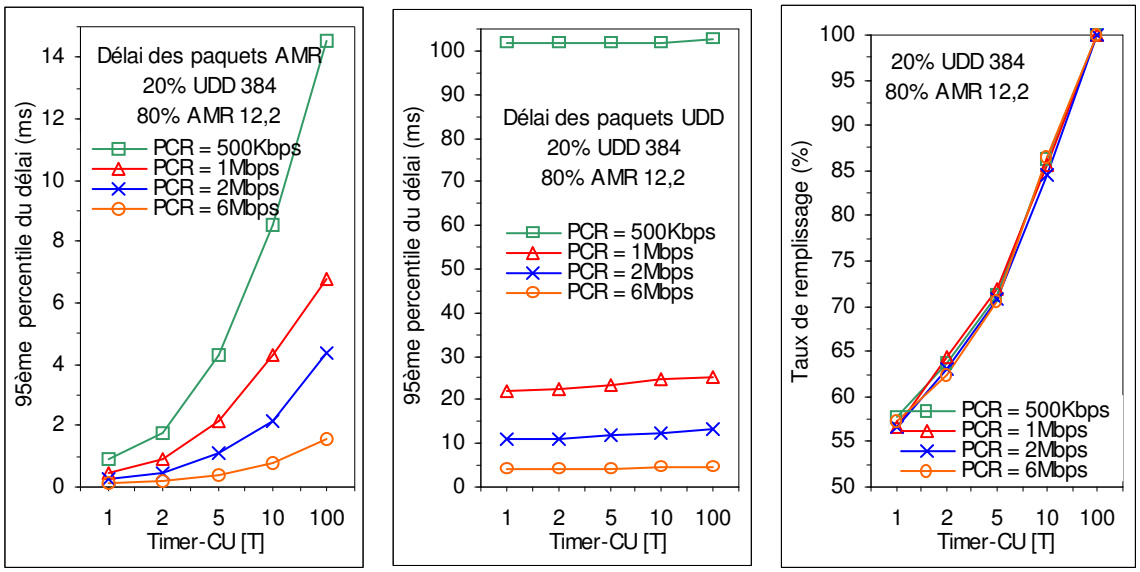


Figure 6.14 VC multi-service 20% UDD384 et 80% AMR12,2

6.3.1.3.3 80% UDD 64 et 20% AMR 12,2

La charge du VC est de 8% environ.

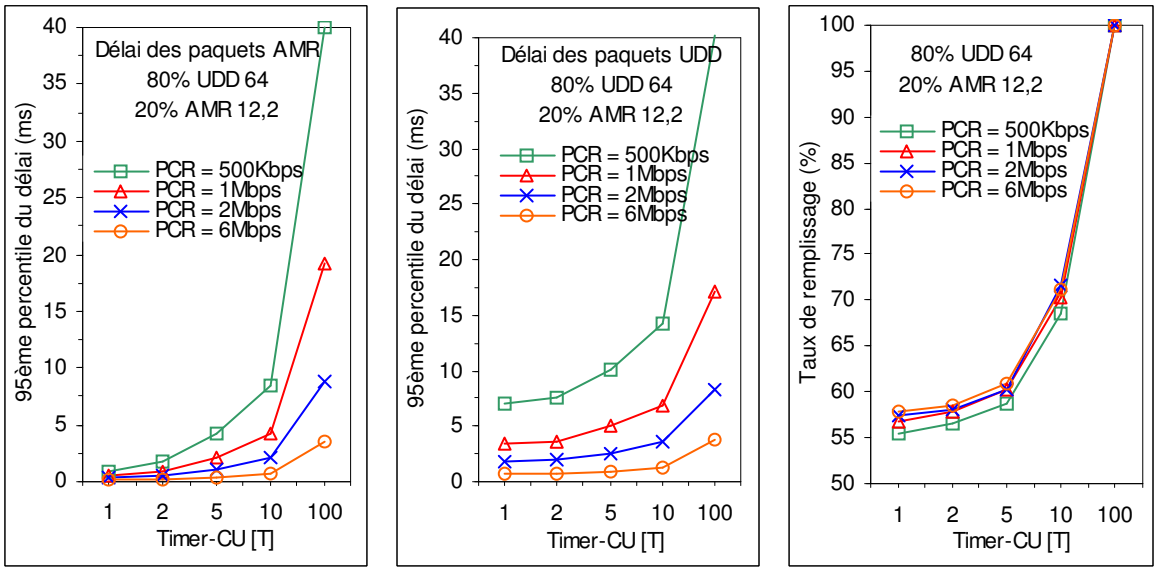


Figure 6.15 VC multi-service 80% UDD64 et 20% AMR12,2

6.3.1.3.4 80% UDD 384 et 20% AMR 12,2

La charge du VC est de 8% environ.

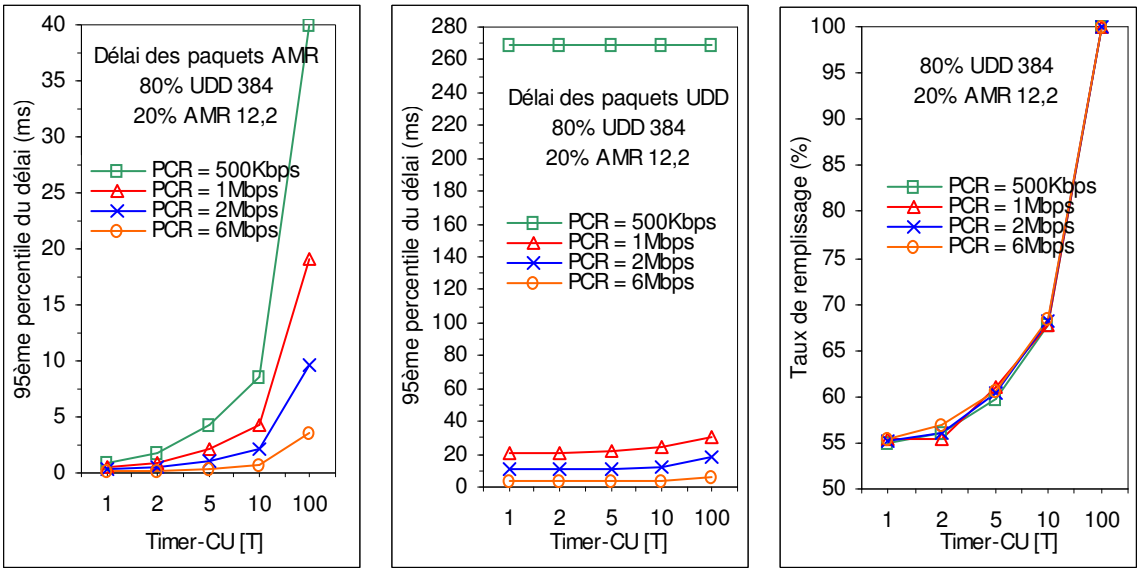


Figure 6.16 VC multi-service 80% UDD384 et 20% AMR12,2

6.3.1.3.5 Synthèse des résultats pour un VC multi-service

Les figures 6.13 et 6.14 représentent les résultats pour un VC à 15% de charge dont le trafic AMR est majoritaire (80%). Dans la première, le mode UDD 64 est utilisé tandis que dans la deuxième, le mode UDD 384 est utilisé. Pour le délai des paquets de voix, il n'y a pas de grande différence entre les deux cas. De même pour le taux de remplissage. Donc, les paramètres de performance ne sont pas très sensibles au mode UDD utilisé. Evidemment, le délai pour les paquets UDD 384 est plus élevé que celui des paquets UDD 64 à cause de leur longueur plus élevée.

Les figures 6.15 et 6.16 représentent les résultats pour un VC à 8% de charge dont le trafic UDD est majoritaire (80%). Les mêmes remarques sont tirées. Mais puisque la charge est plus faible que dans les deux cas précédents, il est alors normal que les délais soient plus élevés et le taux de remplissage soit plus faible. De toute façon, quel que soit le scénario, le taux de remplissage est le même pour tous les VC dont le Timer-CU est le même multiple de la période du VC.

Le trafic des données de nature sporadique est mélangé avec le trafic de voix, il donne des meilleures performances en terme de délai pour les flux de voix. Par exemple, comparons la figure 6.9 avec les résultats du paragraphe §6.3.1.3. Prenons le cas d'un VC de 2 Mbit/s et d'un Timer-CU égal à 10 fois la période T. Le délai des paquets de voix est de 4,3 ms dans le cas de trafic de voix seul (figure 6.9). Ce délai est inférieur à 2,8ms dans tous les scénarios du paragraphe §6.3.1.3 (figures 6.13, 6.14, 6.15 et 6.16). Cela est dû au trafic des données qui vient pour remplir les cellules ATM plus rapidement à cause de son comportement sporadique. Quand un paquet de voix est inséré dans le paquet CPS-PDU, il y a une probabilité plus élevée qu'il y ait des paquets de données dans la file d'attente correspondante. Ces paquets servent pour remplir le CPS-PDU et envoyer rapidement le CPS-PDU, ce qui diminue le délai des paquets de voix. Donc à faible charge, il est avantageux de multiplexer le trafic de voix avec celui des données pour diminuer le délai de multiplexage. Une valeur de Timer-CU choisie dans le cas d'un VC mono-service va donner certainement des bons résultats dans le cas d'un VC mutli-service.

6.3.2 Conclusions sur la valeur du Timer-CU

La première conclusion confirme que la valeur optimale du Timer-CU peut être choisie dans le cas d'un VC mono-service transportant des flux temps-réel (AMR par exemple). Cette valeur donne des bons résultats dans le cas d'un VC multi-service.

Concernant le taux de remplissage à très faible charge ($\rho < 0,1$), une valeur du Timer-CU de l'ordre de 10 fois la période T du PCR donne un taux supérieur à 70%. Il est alors préférable de prendre une valeur de Timer-CU supérieure à 10 fois la période T du PCR sauf si cette valeur dépasse largement le délai toléré dans le nœud AAL2. Dans ce cas, il faut privilégier la contrainte du délai. En effet, dans les spécifications du 3GPP [91], le délai de transport entre le Node B et le RNC ne doit pas dépasser les 5 ms pour les paquets de voix. En tenant compte des temps de traitement et de transmission, le délai de multiplexage dans le module de multiplexage CPS ne doit pas dépasser 2 à 3 ms quand il y a un trafic de voix, ce qui est souvent le cas. Dans le cas où le Node B serait lié au RNC à travers plusieurs nœuds intermédiaires d'un sous-réseau, le temps de transit global (cumulé) dans le sous-réseau ne doit pas dépasser 5 ms et par conséquent, le temps de multiplexage dans un nœud AAL2 ne doit pas dépasser 1 ms environ. Nous pouvons dire enfin qu'une valeur de Timer-CU comprise entre 1 ms et 2 ms semble une valeur raisonnable.

Puisque la valeur du Timer-CU n'a d'incidence que dans le cas de faible charge, nous pouvons dire alors que dans ce cas, nous pouvons choisir une faible valeur du Timer-CU (voire nulle) pour limiter le délai sans tenir compte du taux de remplissage. En effet, puisque le VC est faiblement chargé, alors même si une partie de la bande passante est perdue en terme d'octets de bourrage, cela n'a pas d'importance. Ceci est correct si la connexion AAL2 ne traverse pas des commutateurs ATM. En fait, dans le cas de l'UTRAN, il est possible d'agréger les flux de plusieurs Node B (plusieurs VC) dans un nœud de concentration pour sortir avec un seul tuyau vers le RNC. Dans le cas où le nœud de concentration serait un commutateur ATM et si les liens entrants sont faiblement chargés et ont des valeurs faibles de Timer-CU, alors il y aura beaucoup de perte de bande passante en terme d'octets de bourrage sur le lien sortant. Dans ce cas, les valeurs faibles des Timer-CU ont une grande incidence sur le lien entre le concentrateur (commutateur ATM) et le RNC même si les liens entre les Node B et le concentrateur sont faiblement chargés. Dans ce cas, il faut choisir une valeur de Timer-CU dans la marge 1-2 ms pour avoir un taux de remplissage acceptable tout en respectant les limites temporelles.

6.4 Les mécanismes d'ordonnancement (*Scheduling*)

Dans le chapitre précédent, nous avons présenté trois schémas pour l'agrégation des flux AAL2 dans l'UTRAN. Dans tous les schémas, un mécanisme d'ordonnancement est nécessaire au niveau AAL2 pour pourvoir garantir les délais requis par les mini-cellules. En effet, l'ordonnancement des paquets est une fonction qui vise deux objectifs. Le premier est de fixer le délai de mise en file d'attente pour un flux. Le second consiste à répartir la bande passante disponible entre les flux selon une méthode spécifique. Dans le multiplexeur CPS de la couche AAL2, des paquets appartenant à différents flux se présentent simultanément pour être insérés dans les CPS-PDU avant d'être envoyés vers la couche ATM. Dans ce cas, il faut choisir un seul paquet et par suite les autres paquets se font attendre. Le choix de ce paquet doit être basé sur des critères de performance et de qualité de service (surtout le délai). Un mécanisme d'ordonnancement approprié permet de garantir la qualité de service de chaque flux selon ses exigences. Un mécanisme d'ordonnancement est aussi nécessaire au niveau ATM dans le cas d'agrégation de plusieurs VC dans un même VP.

6.4.1 Les mécanismes étudiés

Dans ce qui suit, nous allons présenter cinq mécanismes d'ordonnancement et leur intégration au niveau AAL2 ainsi que nous allons examiner leurs performances dans différents contextes. Les mécanismes étudiés sont : FIFO, RR, WRR, EDF et priorité.

6.4.1.1 Le mécanisme FIFO (First In First Out)

C'est le mécanisme le plus simple et le plus classique. Tous les flux sont agrégés dans une même file d'attente. Les paquets sont servis l'un après l'autre en commençant par le bas de la file d'attente. Un nouveau paquet, quel que soit le flux auquel il appartient, est empilé dans le haut de la file d'attente. Les paquets sont servis avec une vitesse correspondant à leur taille et à la capacité du lien de sortie. Un ordonnanceur FIFO peut être représenté comme dans le schéma de la figure 6.17. Le paquet qui arrive le premier est servi le premier. Ce mécanisme peut être utilisé pour la file d'attente de la couche CPS, c'est à dire avant le processus de multiplexage des mini-cellules dans les CPS-PDU.

Dans la sous-couche CPS, quand le multiplexeur crée un nouveau paquet CPS-PDU, il prend la mini-cellule qui est en bas de la file d'attente pour remplir la charge utile du CPS-PDU. Cette unité de multiplexage est représentée par le serveur sur la figure 6.17. Si la file d'attente est vide, le multiplexeur attend l'arrivée d'une nouvelle mini-cellule. Dans ce cas le CPS-PDU est gardé dans le serveur tant que le Timer-CU n'a pas expiré.

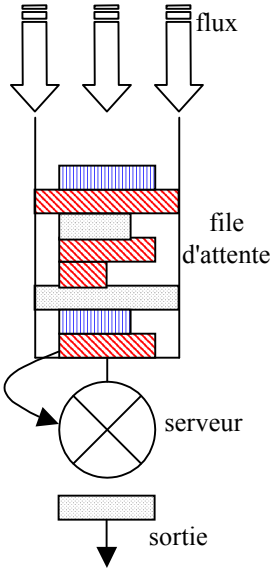


Figure 6.17 Schéma d'un ordonnanceur FIFO

Les avantages du mécanisme FIFO sont:

- Il est simple à implémenter par *hardware* et par *software* et ne nécessite pas beaucoup de temps de traitement.
- L'ordre des paquets est conservé et le délai peut être prévisible. En fait, le délai maximal correspond à la profondeur de la file d'attente.

Les inconvénients de FIFO sont:

- Il ne peut pas faire la différenciation entre plusieurs classes de service.
- Il n'y a pas de séparation entre les flux et par conséquent, un flux peut perturber le fonctionnement des autres flux.

- Un flux de nature sporadique peut consommer toute la bande passante disponible et par suite, le délai et la gigue des flux temps-réel augmentent considérablement.

6.4.1.2 Le mécanisme de priorité (PQ : Priority Queueing)

Ce mécanisme d'ordonnancement prend en compte les différentes classes de service. Ces classes sont classées par ordre de priorité basé sur la borne supérieure du délai de transfert. La file d'attente correspondante à la classe de priorité la plus élevée (priorité 1) est servie en premier tant qu'elle n'est pas vide. Quand elle sera vide, l'algorithme d'ordonnancement sert la file d'attente de priorité 2 et ainsi de suite. Pendant le temps de service de la file d'attente de priorité 2, si un paquet arrive dans la file d'attente de priorité 1, il sera servi puis la classe 2 prend la main si la file de la classe 1 devient vide. Le schéma de la figure 6.18 représente ce type de mécanisme.

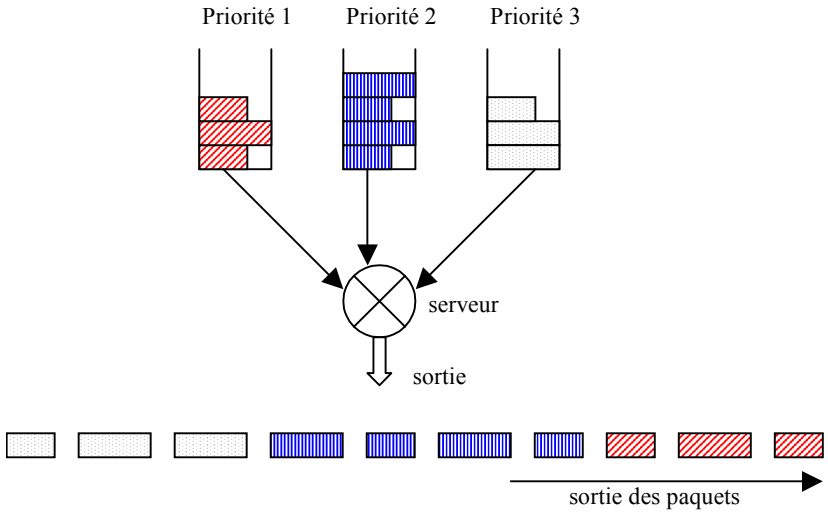


Figure 6.18 Schéma d'un ordonnanceur PQ

Dans chaque file d'attente, plusieurs flux de même classe peuvent être agrégés. Les paquets d'une même classe sont servis suivant une politique FIFO.

Les avantages de cet algorithme d'ordonnancement sont:

- PQ est simple à implémenter par *hardware* et par *software* et ne nécessite pas beaucoup de temps de traitement.
- Il permet de gérer plusieurs classes et de fournir des traitements différents pour les paquets des différentes classes.

Ses inconvénients sont:

- Si le trafic de plus haute priorité n'est pas contrôlé à l'entrée du réseau, les paquets des autres priorités peuvent subir des délais très élevés.
- Tous les flux d'une même classe sont agrégés dans une même file d'attente. Par conséquent, un seul flux peut perturber le fonctionnement des autres flux de la même classe.

6.4.1.3 Le mécanisme RR (Round Robin)

C'est un mécanisme cyclique qui assure la séparation entre les flux. A chaque tour de l'ordonnanceur, un seul paquet de chaque flux (de chaque file d'attente) est envoyé. Si une file d'attente est vide, alors l'ordonnanceur passe vers la file d'attente suivante. Il existe une file d'attente

indépendante pour chaque flux individuel. Le schéma de la figure 6.19 représente un ordonnanceur de type *Round Robin*. Ce mécanisme est appelé encore *Fair Queueing* (FQ).

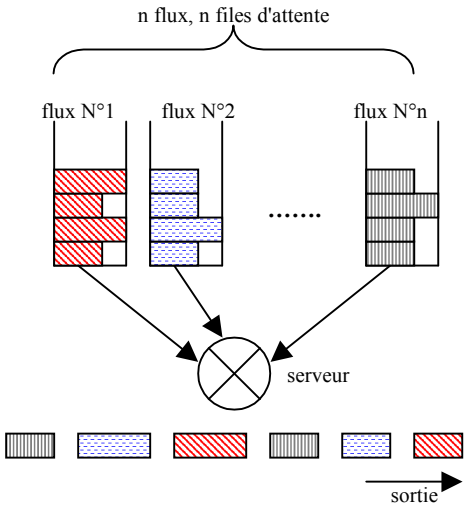


Figure 6.19 Schéma d'un ordonnanceur RR

Le principal avantage de ce mécanisme est la possibilité de séparer les flux. Chaque flux est isolé et son comportement n'altère pas le fonctionnement des autres flux.

D'ailleurs, ce mécanisme a des inconvénients:

- Quand le nombre des flux augmente, la gestion d'un grand nombre de file d'attente sera plus compliquée.
- L'objectif de RR est de partager équitablement la bande passante entre les flux. Cela ne peut pas être assuré que si tous les paquets de tous les flux ont la même taille. En plus, RR n'est pas conçu pour supporter des flux avec différents besoins de bande passante.

6.4.1.4 Le mécanisme WRR (Weighted Round Robin)

Ce mécanisme, appelé aussi CBQ (*Class-Based Queueing*), prend en compte les classes et le partage de la bande passante. L'avantage par rapport au mécanisme *Round Robin* est sa capacité à allouer différentes bandes passantes selon les exigences de chaque classe. Son avantage par rapport au mécanisme de priorité est qu'il garantit une bande passante minimale pour les flux de basse priorité.

En revanche, le premier inconvénient de ce mécanisme est qu'il ne peut partager équitablement la bande passante que si tous les paquets de tous les flux ont la même taille. Le schéma de la figure 6.20 représente le mécanisme WRR.

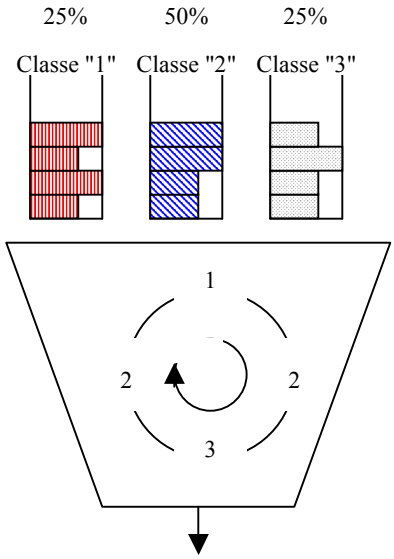


Figure 6.20 Schéma d'un ordonnanceur WRR

Sur ce schéma, 25% de la bande passante sont alloués à la classe 1, 50% à la classe 2 et 25% à la classe 3. L'ordonnanceur choisit à chaque cycle un paquet de la classe 1, deux paquets de la classe 2 et un paquet de la classe 3. Le poids d'une classe correspond au pourcentage de bande passante alloué à cette classe. Dans l'exemple de la figure 6.20, les poids correspondant aux classes 1 et 3 sont de ¼. Celui de la classe 2 et de ½. Dans ce cas, le cycle est de longueur 4. Pour la classe 2 par exemple, deux paquets sont servis à chaque cycle. Ces deux paquets peuvent être servis en même temps lors d'un passage par la file d'attente 2, ou un seul paquet est servi par chaque passage, mais plusieurs visite de la file d'attente 2 sont faites dans chaque cycle (c'est le cas de la figure ci-dessus).

6.4.1.5 Le mécanisme EDF (Earliest Deadline First)

L'idée de base de ce mécanisme d'ordonnancement est d'affecter une date d'échéance (*deadline*) pour chaque paquet arrivant dans la file d'attente. Cette date d'échéance est calculée en ajoutant la borne maximale du délai toléré pour ce flux au temps d'arrivée du paquet. Quand l'ordonnanceur veut envoyer un paquet sur le lien, il choisit le paquet ayant la plus petite date d'échéance. Ce schéma représente une priorité dynamique des paquets qui évolue avec le temps. Ce mécanisme est très compliqué parce qu'il faut assigner une date pour chaque paquet et cela consomme beaucoup de ressources et de temps de traitement par rapport aux autres mécanismes. Mais son avantage essentiel est la possibilité de borner le délai de chaque paquet. Ce mécanisme ne peut être utilisé que si on connaît en avance les *deadlines* pour chaque connexion. Pratiquement, lors de l'entrée d'un paquet dans l'ordonnanceur, l'algorithme EDF calcule son *deadline* en ajoutant l'instant d'arrivée à la limite de délai correspondante au flux de ce paquet. Le résultat est alors enregistré dans un en-tête appelé "*timestamp"* qui est ajouté au début du paquet (ou mini-cellule). Quand l'algorithme veut choisir une mini-cellule, il sélectionne la mini-cellule ayant le plus petit *timestamp*. A la sortie de l'ordonnanceur, le *timestamp* est enlevé et le paquet original est envoyé.

6.4.2 Ordonnancement au niveau AAL2

6.4.2.1 VC mono-service homogène

Dans le cas d'un VC mono-service homogène, les mécanismes d'ordonnancement basés sur la différenciation entre les classes de service comme WRR, EDF et PQ ne sont pas nécessaires. En effet, tous les flux AAL2 ont le même profil et les mêmes contraintes de qualité de service, donc ils doivent être traités de la même manière. Dans ce cas, les deux mécanismes FIFO et RR sont envisageables

puisqu'ils ne distinguent pas entre les flux. FIFO est plus simple à implémenter mais il n'assure pas la séparation des flux. Par suite, un flux qui ne respecte pas son contrat de trafic peut perturber tous les autres flux et par conséquent, dégrader la qualité de service fournie pour ces flux. Dans le cas où le trafic serait contrôlé à l'entrée du réseau, c'est à dire tous les flux respectent leurs contrats de trafic et par suite leurs profils sont les mêmes, le mécanisme FIFO peut être suffisant. Pour vérifier ce fait, nous avons considéré deux VC transportant des flux AMR 12,2 ayant le même profil. Les deux VC sont de type DBR ayant le même débit 2 Mbit/s. A l'entrée du multiplexeur CPS du premier VC, nous avons implémenté un mécanisme d'ordonnancement de type FIFO. Dans le deuxième VC, le mécanisme RR est implémenté. Le Timer-CU pour les deux VC est le même et a une valeur très faible (proche de zéro).

Nous avons réalisé des simulations pour comparer les deux mécanismes FIFO et RR. La figure 6.21 représente le délai des mini-cellules dans les deux VC. La figure 6.22 représente la variation du délai sous la forme de l'écart-type.

Les deux mécanismes donnent presque les mêmes résultats. Le mécanisme FIFO donne parfois des résultats légèrement meilleurs que RR. Ce fait peut être expliqué comme suit: si une mini-cellule du flux "i" arrive dans la file d'attente correspondante juste après que cette file d'attente est visitée par l'ordonnanceur, cette mini-cellule est alors obligée d'attendre un cycle complet pour que l'ordonnanceur visite de nouveau sa file d'attente. A ce moment, si une mini-cellule arrive dans la file d'attente "i+k" par exemple, elle sera servie avant la mini-cellule du flux "i" sachant que cette dernière est arrivée en premier. Ce fait peut augmenter le délai de certaines mini-cellules et surtout il a un impact sur la variation du délai. Dans le cas d'un mécanisme FIFO, cet effet n'existe pas, les mini-cellules sont servies dans leur ordre d'arrivée. Quand le nombre des flux est faible, cet effet est invisible, mais quand le nombre de flux augmente, cet effet devient de plus en plus clair parce que la durée d'un cycle devient plus longue. C'est pourquoi, on remarque des meilleures performances pour le mécanisme FIFO dans le cas d'un grand nombre d'utilisateurs.

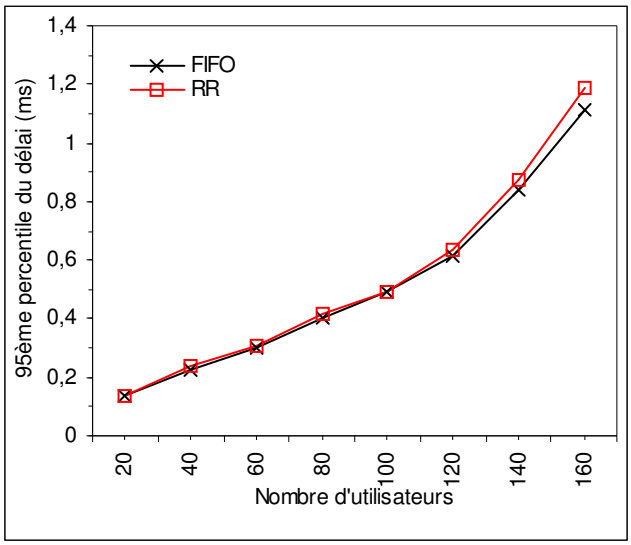


Figure 6.21 95ème percentile du délai des mini-cellules

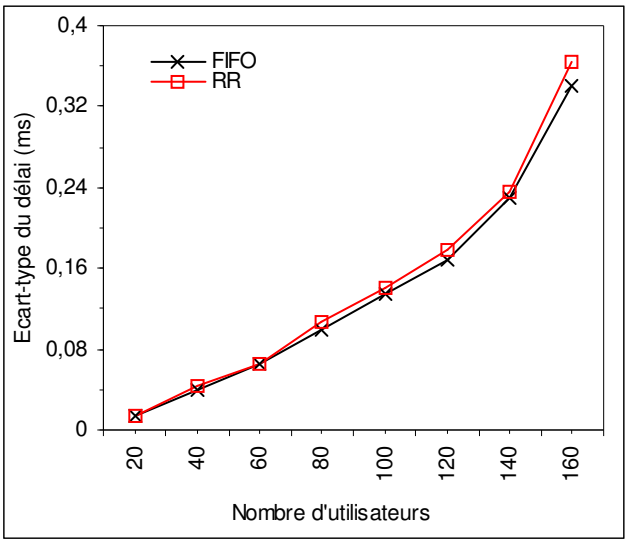


Figure 6.22 Ecart-type du délai

D'après ce qui précède, le mécanisme FIFO donne des bonnes performances dans le cas d'un VC mono-service homogène et il est plus simple à implémenter. Alors, nous adoptons ce type d'ordonnancement à condition de bien contrôler le trafic à l'entrée du réseau. Dans la suite, tous les flux homogènes de même classe seront agrégés dans une seule file d'attente de type FIFO.

6.4.2.2 VC hétérogène

Dans le cas d'un VC hétérogène, les différents flux n'ont pas les mêmes contraintes de qualité de service. Il faut alors utiliser des algorithmes d'ordonnancement basés sur la différenciation entre les classes. En plus, puisque les flux ne sont pas homogènes, ils peuvent demander des bandes passantes différentes. Alors, des mécanismes d'ordonnancement qui tiennent compte d'un partage inéquitable de la bande passante sont aussi à considérés. Nous avons proposé deux mécanismes tenant compte de ces deux caractéristiques : WRR et EDF.

Pour analyser les performances de ces deux algorithmes, nous avons considéré plusieurs scénarios de VC hétérogènes afin d'étudier une large fourchette de cas possibles. En effet, il faut étudier chaque mécanisme dans plusieurs contextes pour s'assurer de sa robustesse. Nous avons considéré un trafic de données UDD agrégé avec un trafic AMR 12.2 kbit/s. Le VC est de type DBR dont le PCR est de 2 Mbit/s. La valeur du Timer-CU est de 1 ms. Plusieurs modes UDD sont utilisés ainsi que plusieurs combinaisons de trafic AMR et UDD. Les deux algorithmes WRR et EDF sont comparés avec les mécanismes FIFO et PQ.

Les distributions de probabilité du délai des paquets de voix et des données sont examinées. Ces courbes de distribution de probabilité donnent la probabilité pour que le délai d'un paquet soit supérieur à une valeur donnée. Notons que dans l'algorithme EDF, si le deadline d'un paquet est dépassé, ce paquet peut être rejeté ou laissé dans la file d'attente. Dans le deuxième cas, le service des paquets suit la même discipline, c'est à dire, on sert le paquet avec le deadline le plus petit.

6.4.2.2.1 Scénario 1: 20% AMR et 80% UDD 64

Dans ce scénario, le trafic des données UDD est majoritaire. La charge du VC est de 65% environ.

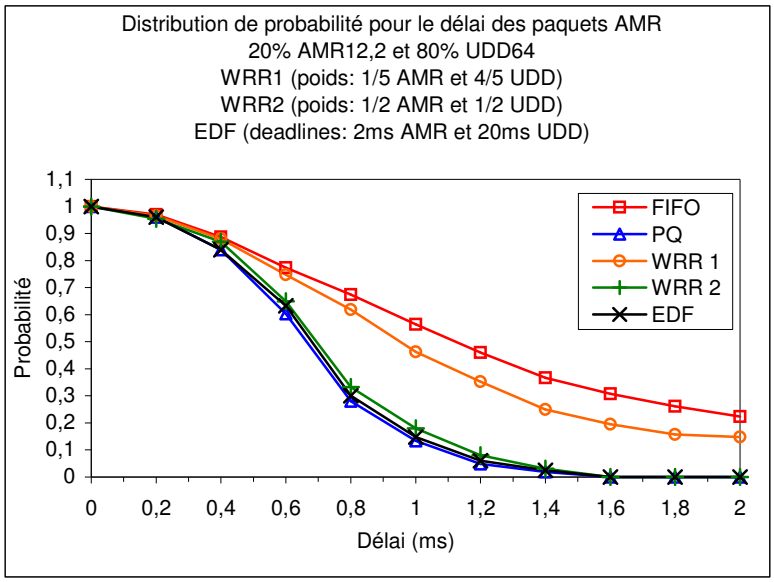


Figure 6.23 Scénario 1 : Distribution de probabilité du délai des paquets AMR

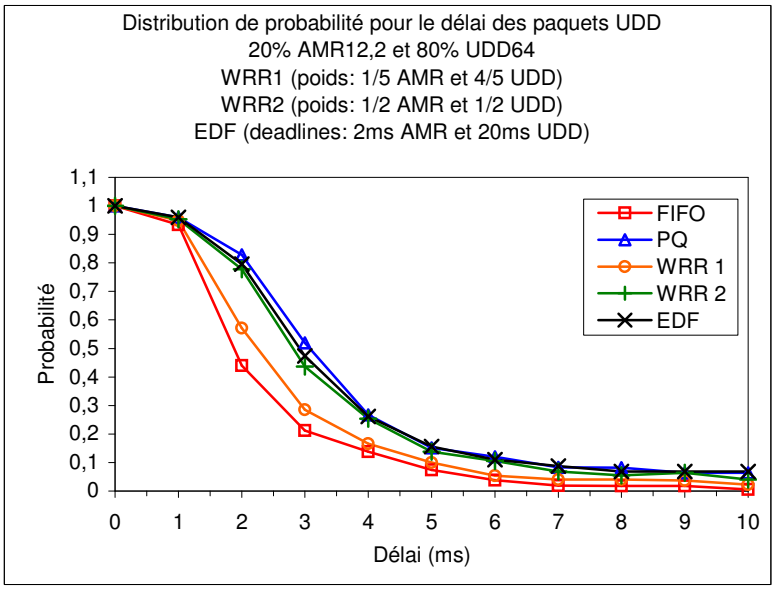


Figure 6.24 Scénario 1 : Distribution de probabilité du délai des paquets UDD

Il est clair que le mécanisme PQ qui privilégie les paquets de voix donne des meilleures performances pour ces paquets. Par contre, il pénalise les paquets des données. FIFO n'est pas adapté pour ce type de VC hétérogène parce q'un flux non temps-réel de nature sporadique peut occuper une grande partie de la file d'attente et par conséquent, retarder énormément les paquets temps-réel comme la voix. Pour les solutions intermédiaires WRR et EDF, les performances de chacune dépendent des paramètres choisis, c'est à dire des poids de WRR et des *deadlines* de EDF.

Pour EDF, nous avons choisi un *deadline* de 2 ms pour les paquets de voix AMR puisqu'ils sont très exigeants en terme de délai de transfert. En effet, comme nous l'avons déjà mentionné dans le paragraphe §6.3.2, le délai des paquets de voix ne doit pas dépasser 2 ms dans un multiplexeur AAL2. Pour les paquets UDD plus tolérants au délai, nous avons considéré un *deadline* de 20 ms. En effet, dans les spécifications du 3GPP [91], le délai toléré pour ce type de paquet sur l'interface Iub est de l'ordre de 50 ms. Nous pouvons alors considéré que le délai dans un nœud AAL2 ne doit pas dépassé 20 ms. Pour les paquets de voix, EDF donne des performances très proches du mécanisme PQ. Par contre, il est moins performant pour les paquets des données.

Pour le mécanisme WRR, nous avons considéré deux configurations:

- Dans la première configuration WRR1, et puisque le trafic de voix constitue 20% du trafic total, nous avons donné un poids de 1/5 (20%) pour la file d'attente des paquets de voix et de 4/5 (80%) pour la file d'attente des paquets des données. Cette configuration n'a pas donné des meilleures performances pour les paquets de voix parce qu'elle ne tient pas compte des contraintes temporelles de chaque type de flux. En fait, même si les flux UDD sont majoritaires, ils tolèrent un certain délai tandis que les flux AMR minoritaires sont très stricts en terme de délai. En plus, le partage de la bande passante dans WRR s'effectue sur la base du nombre des paquets servis. Mais puisque les paquets de voix sont plus courts que les paquets des donnés, alors le partage de la bande passante n'est pas effectivement équitable.
- Dans la deuxième configuration WRR2, nous avons essayé de remédier à ce problème en augmentant le poids de la file d'attente des flux de voix. Les poids sont: ½ pour les flux AMR et ½ pour le flux UDD. Les performances de WRR2 sont alors proches de celles de PQ ou de EDF.

Les poids du mécanisme WRR ne doivent pas être choisis uniquement en fonction du partage de la bande passante, mais aussi en fonction des contraintes temporelles de chaque classe. Dans tous les cas, quand un mécanisme d'ordonnancement donne des bonnes performances pour les paquets de voix, il donne des mauvaises performances pour les autres flux et vice-versa.

6.4.2.2.2 Scénario 2: 20% AMR et 80% UDD 384

La charge du VC est de 70% environ.

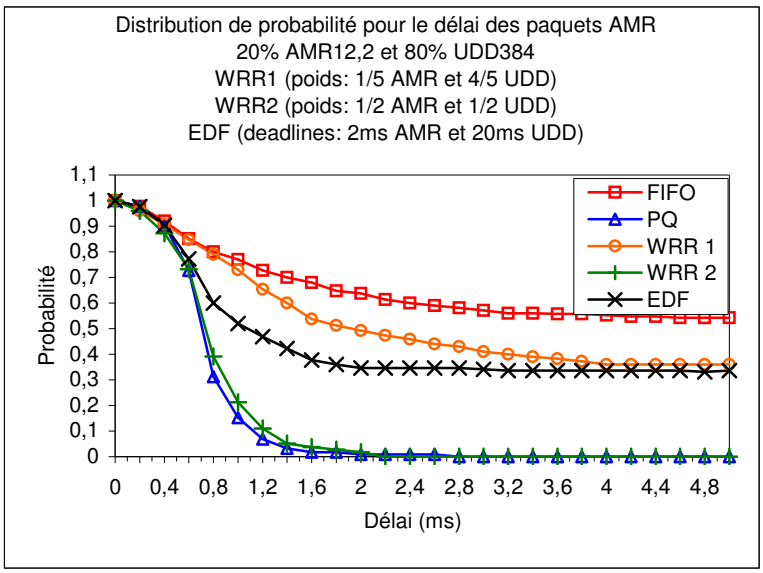


Figure 6.25 Scénario 2 : Distribution de probabilité du délai des paquets AMR

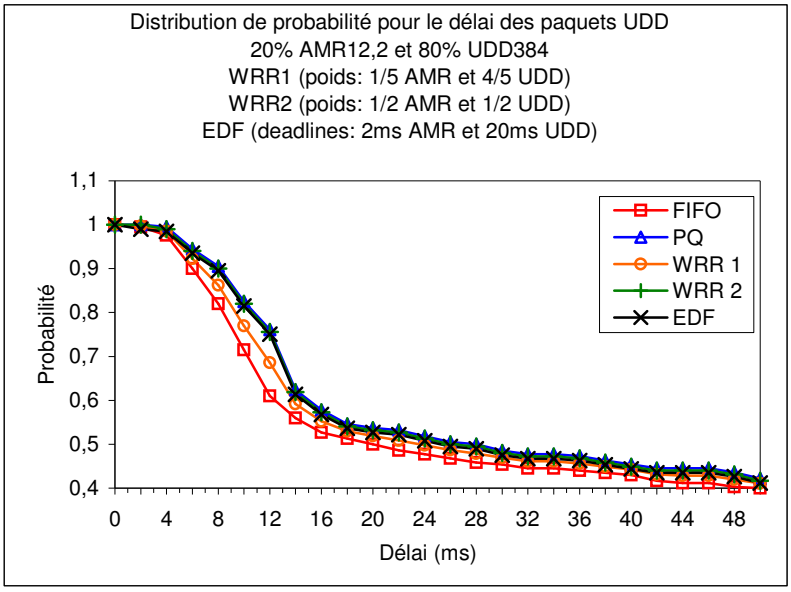


Figure 6.26 Scénario 2 : Distribution de probabilité du délai des paquets UDD

La figure 6.25 représente la distribution de probabilité du délai des paquets de voix AMR. Les courbes divergent et leur descente est plus lente. Cela signifie que les délais deviennent plus élevés. Toujours, les mécanismes PQ et WRR avec poids élevé pour les paquets AMR sont performants pour les flux AMR. FIFO et WRR avec poids faible pour les paquets AMR donnent des mauvaises performances pour les flux AMR comme dans le scénario 1. Les performances du mécanisme EDF sont meilleures dans le scénario 1. En effet, dans le scénario 2, on utilise des flux UDD 384 kbit/s. Les paquets FP-PDU ou les AAL2-SDU ont une très grande taille. Quand un paquet AAL2-SDU arrive dans la couche AAL2, il est décomposé en plusieurs mini-cellules qui sont envoyées dans la file d'attente convenable dans le multiplexeur CPS. Toutes ces mini-cellules arrivant en même temps acquièrent le même *deadline* "δ" parce qu'elles ont toutes la même contrainte temporelle puisqu'elles appartiennent à la même connexion. Supposons qu'un paquet AMR arrive à un instant donné et son *deadline* soit égal à "δ+ε", où ε est un nombre positif très petit. Quand l'ordonnanceur va choisir un paquet, il va évidemment choisir le paquet de plus faible *deadline*. Dans ce cas, tous les segments (les mini-cellules) du long paquet des données (dont le *deadline* est δ) seront servis avant le paquet de voix (dont le *deadline* est δ+ε) parce qu'ils ont tous un deadline plus faible. Ce fait pénalise les paquets de petites tailles. Donc, le mécanisme EDF n'est pas adapté à ce genre de situation où des paquets de grandes tailles seront mélangés avec des paquets temps-réel de petites tailles. Dans ce cas, il est préférable de séparer les flux des longs paquets et les flux des petits paquets. Le mécanisme EDF est bien adapté au cas où les flux auraient des contraintes temporelles différentes et des paquets de tailles relativement semblables.

6.4.2.2.3 Scénario 3 : 80% AMR et 20% UDD 64

Dans ce scénario, le trafic AMR est majoritaire. La charge du VC est de 60% environ.

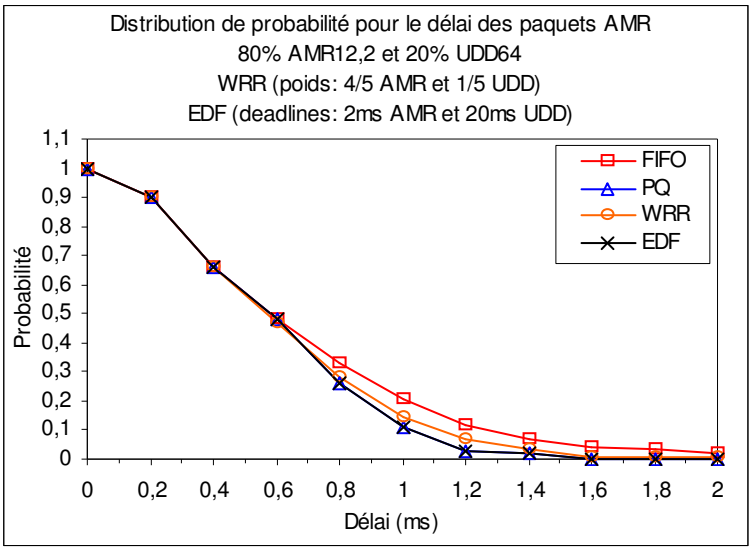


Figure 6.27 Scénario 3 : Distribution de probabilité du délai des paquets AMR

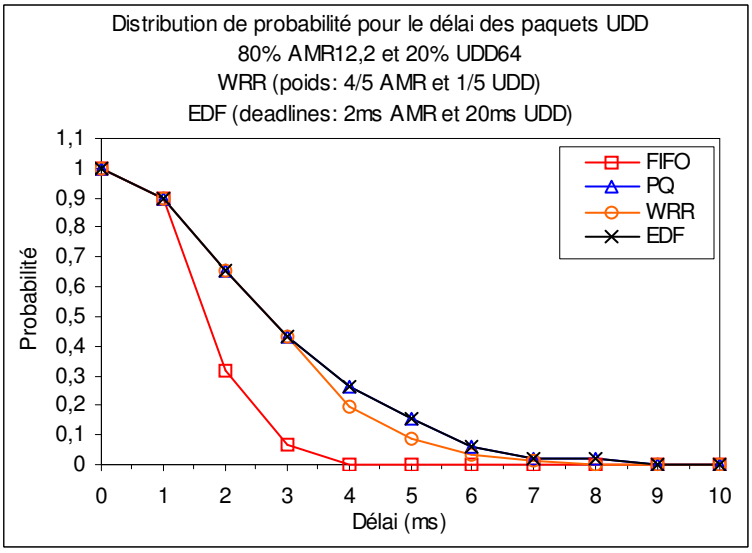


Figure 6.28 Scénario 3 : Distribution de probabilité du délai des paquets UDD

6.4.2.2.4 Scénario 4 : 80% AMR et 20% UDD 384

La charge du VC est de 60% environ.

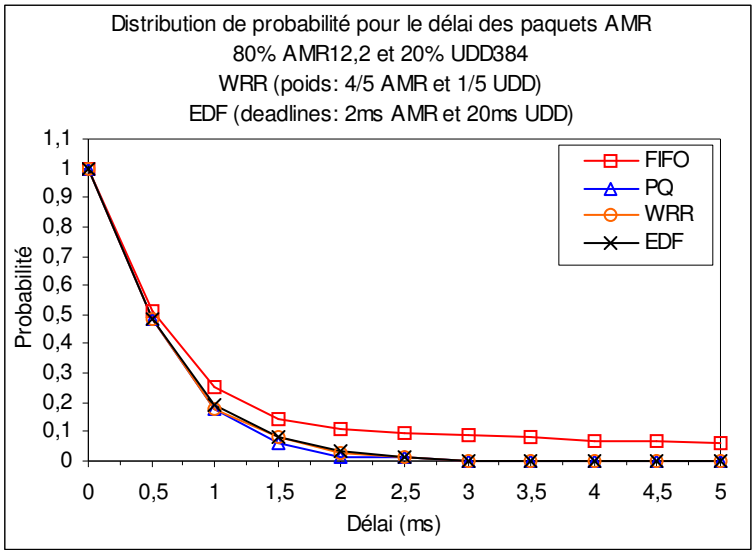


Figure 6.29 Scénario 4 : Distribution de probabilité du délai des paquets AMR

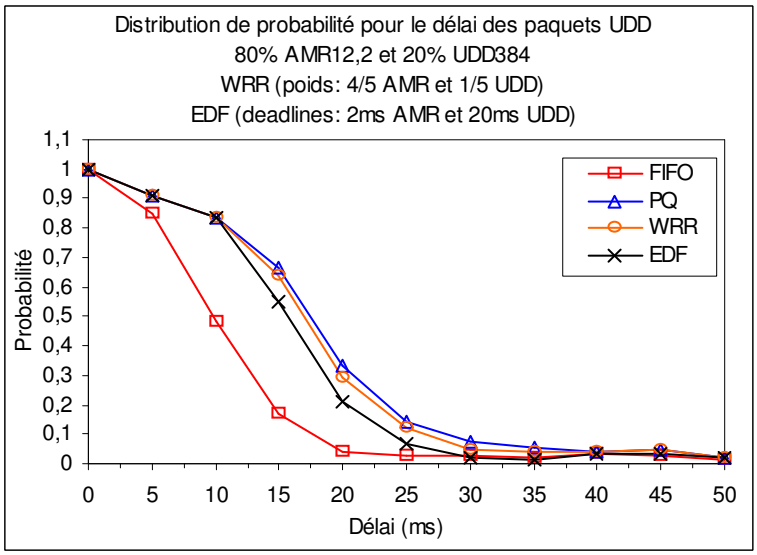


Figure 6.30 Scénario 4 : Distribution de probabilité du délai des paquets UDD

6.4.3 Synthèse des résultats des mécanismes d'ordonnancement

Dans le cas d'un VC mono-service homogène, la conclusion essentielle est qu'un simple algorithme FIFO peut être performant si le trafic est bien contrôlé à l'entrée du réseau. Pour que tous les flux aient le même profil au niveau de l'ordonnanceur et par suite aucun flux ne perturbe le fonctionnement des autres flux, il faut appliquer les algorithmes de conformité décrits dans le chapitre 4. Les paquets non-conformes seront éliminés. Dans les résultats précédents, tous les flux sont conformes et par suite, aucun flux ne peut perturber les autres flux de la même file d'attente.

Dans le cas d'un VC hétérogène, nous avons considéré le cas des flux AMR agrégés avec des flux UDD. Il est toujours possible de considérer un grand nombre de scénarios où plusieurs types de flux sont agrégés. Mais le cas que nous avons considéré est le plus fréquent dans les réseaux UTRAN. En plus, ce scénario est le plus critique parce qu'on agrège des flux très exigeants en terme de délai, dont les paquets sont de petites tailles, avec des flux très sporadiques dont les paquets sont de grandes tailles. Dans ce cas de VC hétérogène, que ce soit mono-service (plusieurs types de flux mais de même classe) ou multi-service, le mécanisme d'ordonnancement qu'on doit implémenter doit prendre en compte les contraintes temporelles ainsi que le partage de la bande passante. FIFO et PQ sont déjà exclus. Puisque EDF présente un inconvénient dans le cas des paquets non temps-réel de longueurs élevées, alors WRR semble une bonne solution surtout que son implémentation est plus facile que celle de EDF. Mais avec WRR, il faut connaître à l'avance le pourcentage de trafic de chaque classe et prendre en compte les contraintes temporelles de chacune pour choisir les bons paramètres (les poids). En effet, on ne peut pas choisir les poids en se basant uniquement sur le pourcentage de trafic de chaque classe. Il faut prendre en considération la taille moyenne des paquets de chaque classe et leurs besoins en terme de délai de transit. Supposons qu'on a N flux dont on connaît la taille moyenne des paquets de chacun ainsi que le pourcentage de trafic de chaque flux par rapport au trafic global. Soit Li la taille moyenne du flux "i". Soit "Lmax" la valeur maximale de toutes les Li. Soit Pi le pourcentage du trafic du flux "i". Le poids Wi du flux "i" est calculé par la formule suivante: Wi = Pi.Lmax /Li. Cette méthode nécessite la connaissance préalable du pourcentage de chaque flux. Le point faible du mécanisme WRR est qu'il ne prend pas en compte les contraintes temporelles de chaque flux. Pour remédier à ce problème, nous proposons un nouvel algorithme de type WRR dont les poids de chaque flux varient dynamiquement en tenant compte des contraintes temporelles ainsi que des changements des pourcentages des différents flux. Cet algorithme est appelé DyWRR (*Dynamic Weighted Round Robin*) et il est une amélioration de l'algorithme WRR.

6.4.4 L'algorithme DyWRR (*Dynamic Weighted Round Robin*)

6.4.4.1 Définition

C'est un algorithme de type WRR dont les poids ne sont pas fixes, mais ils varient avec le temps selon le volume du trafic de chaque flux. Cet algorithme nécessite uniquement la connaissance préalable des contraintes temporelles de chaque flux.

Soit Ti le temps d'attente maximal autorisé pour un paquet du flux "i" dans la file d'attente correspondante. Soit Bi la longueur moyenne en octet de la file d'attente du flux "i" pendant un intervalle de temps Δt.

La bande passante moyenne nécessaire pour que la file d'attente "i" écoulent ses "Bi" octets pendant un temps "Ti" est de: $BW_i = \frac{B_i}{T_i}$.

En fait, il est possible que cette bande passante ne soit pas disponible à un instant donné, mais on peut donner le pourcentage de bande passante nécessaire pour le flux "i" par rapport aux autres flux. Ce pourcentage Pi est donné par la formule suivante:

$$P_i = \frac{BW_i}{\sum_{k=1}^{N} BW_k}$$ où N est le nombre de flux.

Pi prend en compte le volume de trafic et la contrainte temporelle de chaque flux mais ne prend pas en compte la taille des paquets de chaque flux. Pour corriger ce paramètre, on le multiplie par le facteur correcteur $\frac{L_{max}}{L_i}$; Où Li est la taille moyenne des paquets du flux "i". Lmax est la valeur

maximale de toutes les tailles moyennes Li: $L_{max}=MAX(L_i)$. Li est mesurée durant l'intervalle de temps Δt. Le poids Wi (*Weight*) du flux "i" sera alors :

$$W_i=\frac{L_{max}}{L_i}\times\frac{\frac{B_i}{T_i}}{\sum_{k=1}^{N}\frac{B_k}{T_k}}$$

Pendant un intervalle de temps Δt, l'algorithme mesure la taille moyenne de chaque file d'attente ainsi que la taille moyenne des paquets de chaque flux. A la fin de cet intervalle, et en tenant compte des paramètres fixés au préalable (les Ti), l'algorithme DyWRR calcule les nouveaux poids des différentes files d'attente. Ensuite, un mécanisme de type WRR est appliqué en utilisant les nouveaux poids calculés.

L'avantage de cet algorithme est qu'il ne nécessite pas la connaissance préalable de la distribution du trafic des différents flux et il s'adapte rapidement et automatiquement avec tout changement des pourcentages des trafics. Dans l'algorithme WRR normal, on doit choisir les poids au préalable selon notre estimation du trafic, et si la distribution du trafic change à un instant donné, WRR ne sera plus équitable. DyWRR est interactif et répond rapidement au changement de trafic. DyWRR s'adapte aussi aux arrivées groupées. Il peut donner un poids plus élevé à un flux donné, et cela pour une courte durée, même si ce trafic est faible à long terme. Il peut absorber les rafales du flux minoritaire si le flux majoritaire est faible pendant une courte durée. De cette façon, il réduit les délais des paquets en rafale. L'inconvénient de cet algorithme est le temps de traitement. En effet, cet algorithme est implémenté par *Software* et ne peut pas être implémenté par *Hardware* ce qui limite la vitesse de traitement. La figure 6.31 représente le schéma d'un ordonnanceur DyWRR.

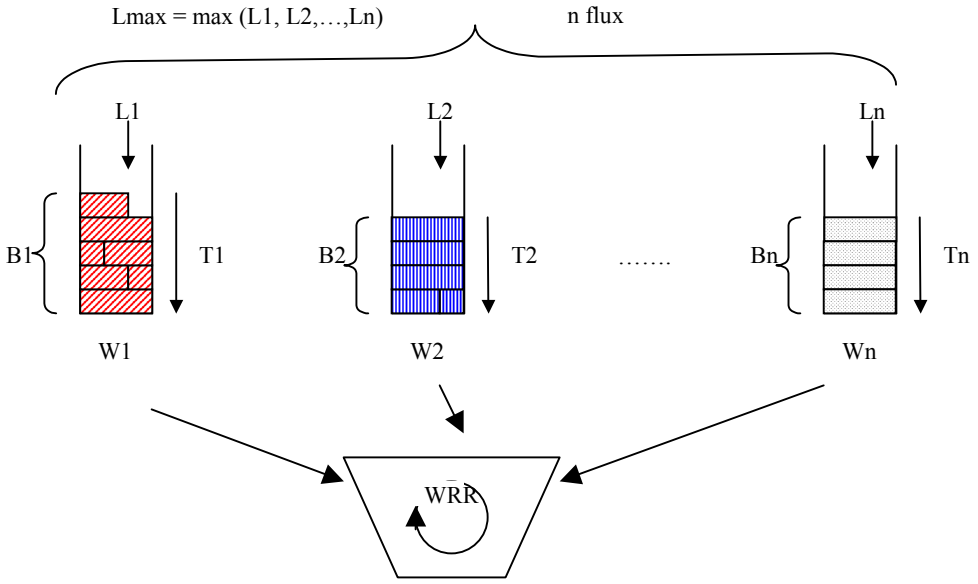


Figure 6.31 Schéma d'un ordonnanceur DyWRR

L'intervalle de temps de mise à jour Δt ne doit pas être trop grand pour que le mécanisme DyWRR réponde le plus vite possible aux changements de trafic. D'un autre côté, il ne doit pas être trop petit pour ne pas entraîner l'instabilité du système.

6.4.4.2 Validation et performances

Pour valider notre algorithme, nous avons procédé par simulation. Nous avons considéré une contrainte temporelle de 2 ms pour les paquets de voix et de 20 ms pour les paquets des données (T_{voix} = 2ms, $T_{données}$ = 20ms). La combinaison du trafic n'est pas la même mais elle varie avec le temps. Par exemple, nous passons d'une combinaison (80% voix et 20% données) à une combinaison (50% voix et 50% données) puis à une combinaison (20% voix et 80% données) pendant la même simulation. Pratiquement, ce scénario peut arriver selon l'activité des utilisateurs. Nous avons considéré deux cas:

- Dans le premier cas, le mécanisme d'ordonnancement implémenté au niveau du multiplexeur CPS est le WRR. Pour choisir les poids, nous avons considéré que le trafic de départ était de 20% voix et 80% données. Dans ce cas, le poids pour la file d'attente de voix est de 2/5 et celui de la file d'attente des données est de 3/5. Ces valeurs sont fixes même si la combinaison de trafic change. En effet, ce sont des paramètres semi-statiques choisis lors de la configuration du réseau.
- Dans le deuxième cas, nous avons considéré le mécanisme DyWRR comme ordonnanceur au niveau du multiplexeur CPS. Dans ce cas, nous n'avons pas besoin de connaître la combinaison du trafic. Même si cette combinaison change avec le temps, le mécanisme DyWRR s'adapte à ce changement et change ses poids en tenant compte des paramètres déjà mentionnés précédemment.

Nous avons considéré un VC de 2 Mbit/s chargé à 60%. Les flux utilisés sont des flux de voix AMR12.2 et de *web browsing* en mode UDD64. La valeur du Timer-CU est fixée à 1 ms.

Nous avons comparé les délais des paquets de voix et des données pour les deux mécanismes utilisés. La figure 6.32 représente le délai des paquets de voix. Pour WRR, on remarque que le délai augmente lorsque le trafic de voix augmente. Ceci est logique parce que le poids de la file d'attente de voix choisi au début, n'est plus adapté à la nouvelle situation. Par contre, le mécanisme DyWRR s'adapte à ce changement de trafic et stabilise le délai des paquets de voix. Ceci va évidemment affecter le délai des paquets des données qui augmente de 1 ou 2 ms. Mais puisque ces paquets sont plus tolérants au délai, alors on préfère privilégier les paquets de voix tout en respectant la borne de délai des paquets des données. Par contre, WRR donne des délais plus faibles pour les paquets des données parce que leur poids est plus grand, mais il affecte les paquets de voix qui sont plus sensibles aux délais. L'amélioration du délai des paquets des données n'est pas utile dans ce cas parce que ces paquets tolèrent un délai plus élevé.

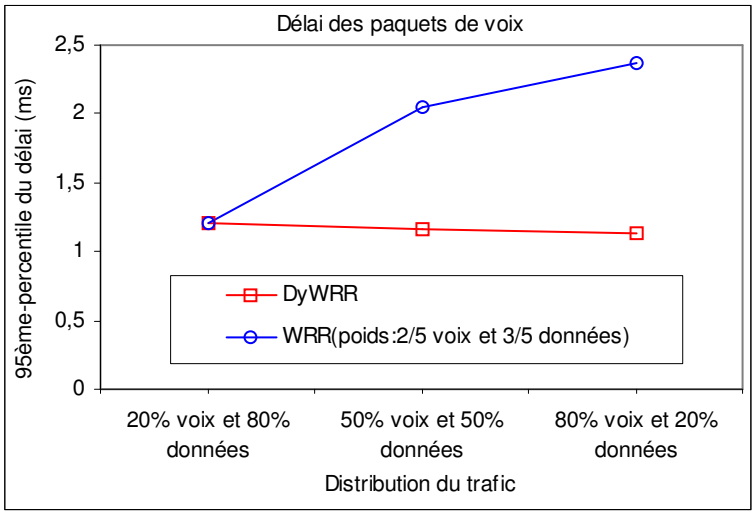


Figure 6.32 DyWRR vs WRR: Délai des paquets de voix

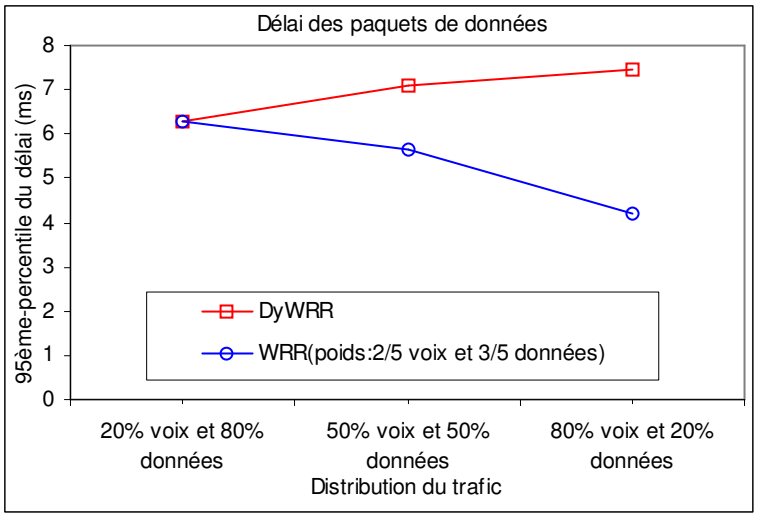


Figure 6.33 DyWRR vs WRR: délai des paquets de données

6.4.4.3 Conclusion

Le mécanisme d'ordonnancement DyWRR est bien adapté aux changements du trafic et donne des bonnes performances dans le cas d'un multiplexeur AAL2. En plus, Il tient compte des contraintes temporelles de chaque flux, c'est pourquoi on peut dire qu'il est un algorithme d'ordonnancement «orienté-QoS».

Cet algorithme peut présenter un problème d'oscillation dans le cas où la combinaison du trafic changerait fréquemment. Pour remédier à ce problème, on choisit une valeur Δt (l'intervalle de mise à jour) élevée tout en respectant un temps de réponse acceptable.

L'algorithme DyWRR est plus efficace que l'algorithme WRR surtout dans le cas de changements fréquents de la combinaison du trafic. En plus, DyWRR tient compte des contraintes temporelles de chaque flux ce qui n'est pas le cas de WRR. Evidemment, DyWRR présente une plus grande compléxité d'implémentation par rapport à WRR.

L'algorithme EDF prend en compte les contraintes temporelles comme DyWRR. Mais, du point de vue complexité d'implémentation, l'algorithme DyWRR nécessite un seul temporisateur pour l'intervalle de mise à jour. Ceci facilite la tâche de gestion des temporisateurs en comparaison avec l'algorithme EDF qui nécessite la gestion d'un nombre élevé de temporisateurs.

6.5 La bande passante équivalente

On a déjà vu dans le chapitre 5 qu'on peut implémenter une fonction de contrôle d'admission CAC en se basant sur la valeur de la bande passante équivalente. Une nouvelle connexion AAL2 est acceptée si la somme des bandes passantes équivalentes de toutes les connexions existantes avec la nouvelle connexion est inférieure ou égale à un certain pourcentage α du PCR du VC DBR.

La bande passante équivalente peut être calculée analytiquement mais cette valeur ne tient pas compte de l'effet du Timer-CU et surtout des octets de bourrage. Alors, on trouve souvent qu'en pratique, la bande passante équivalente mesurée est supérieure à celle calculée. Le pourcentage α doit aussi être connu pour pourvoir établir une fonction CAC basée sur la bande passante équivalente. Ce paramètre peut être mesuré par simulation ou sur un réseau réel. Dans ce paragraphe, nous allons présenter quelques exemples pour mesurer par simulation le cœfficient α ainsi que la bande passante

équivalente de certains types de flux. Les flux utilisés sont de type AMR (voix) ou UDD (*web browsing*).

Nous avons considéré deux VC différents:

- Un VC mono-service homogène transportant des flux de type AMR 12,2 kbit/s.
- Un VC mono-service homogène transportant des flux de type UDD 64 kbit/s.

Pour chaque VC, nous avons mesuré le délai, le taux d'utilisation maximal et le débit ATM moyen pour pouvoir calculer la bande passante équivalente et le cœfficient α. Nous avons considéré plusieurs valeurs de PCR pour analyser l'influence de la capacité du VC sur le cœfficient α. La valeur du Timer-CU utilisée est de 1 ms dans toutes les simulations.

6.5.1 Scénario 1: VC mono-service AMR 12,2 kbit/s

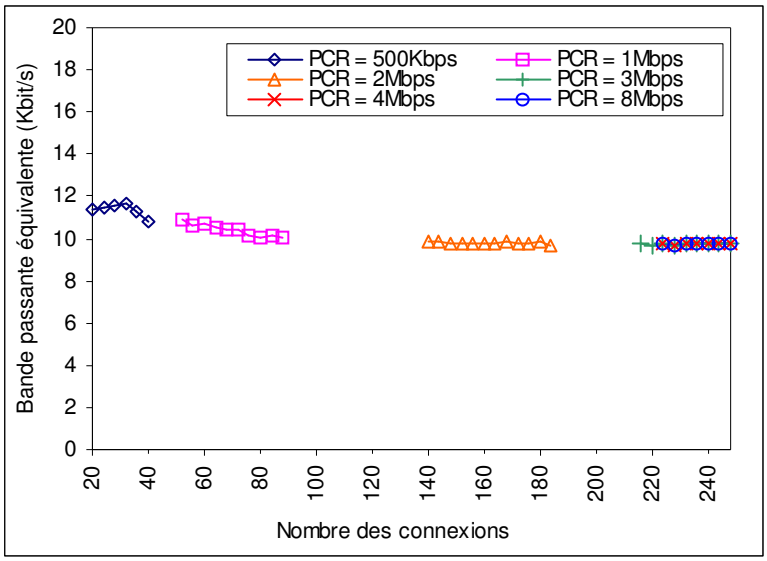


Figure 6.34 Bande passante équivalente des flux AMR 12,2 kbit/s

La figure 6.34 représente la bande passante équivalente d'un flux AMR 12,2 dans un VC mono-service homogène. Cette bande passante équivalente diminue avec le nombre d'utilisateurs. En effet, quand le nombre d'utilisateurs augmente, le taux de remplissage des cellules ATM augmente et par suite, la bande passante perdue en terme d'octets de bourrage diminue. A faible charge, une bande passante additionnelle due au bourrage vient pour s'additionner à la bande passante réellement utilisée par les flux AMR. On remarque aussi que la bande passante équivalente diminue légèrement quand le PCR du VC augmente. En fait, ce n'est pas à cause de la valeur du PCR mais parce que dans le cas d'un gros VC, le nombre d'utilisateurs est plus élevé et par suite, le taux de remplissage est plus élevé ce qui diminue la perte de bande passante en terme de bourrage. Tous ces effets ne seront pas remarqués si le Timer-CU ait une valeur infinie. Dans ce cas, le taux de remplissage est maximal et la bande passante équivalente est minimale. L'équation donnée dans le chapitre 5 pour le calcul de la bande passante équivalente donne un résultat semblable au cas d'un Timer-CU infini. Sur la figure 6.34, la bande passante équivalente dans un VC de 2 Mbit/s est de 9,7 kbit/s. La bande passante équivalente calculée par l'équation est de 9,25 kbit/s. Cette légère différence est due aux 2% de la bande passante perdue sous forme de bourrage.

Dans un VC de faible PCR, on ne peut pas atteindre un grand nombre d'utilisateurs parce que le délai devient très élevé. Dans ce cas, le nombre d'utilisateurs maximal qu'on peut atteindre ne serait pas suffisant pour obtenir un gain statistique maximal et par conséquent une bande passante minimale.

Dans ce cas, la bande passante équivalente est plus élevée dans un VC de faible capacité. La 6.35 représente la bande passante équivalente pour un nombre maximal d'utilisateurs pour chaque VC. A partir d'une valeur de PCR de 1,5 Mbit/s, la bande passante atteint une valeur constante puisque le taux de remplissage se rapproche de 100%. Dans le cas du VC de 0,5 Mbit/s, le taux de remplissage est de 90% et par suite, la bande passante équivalente est plus élevée à cause des 10% de bourrage.

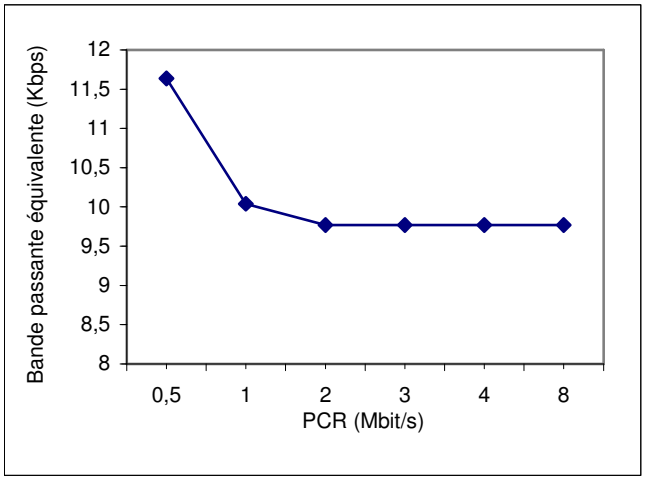


Figure 6.35 Bande passante équivalente en fonction du PCR pour les flux AMR 12,2

La figure 6.36 représente le 95ème percentile du délai des paquets AMR. Elle peut être utilisée pour calculer le facteur α. En effet, α peut être défini comme le rapport entre D et PCR, où D est le débit total maximal qu'on peut atteindre en respectant la limite du délai. En d'autre terme, α est le taux d'utilisation maximal de la bande passante du VC.

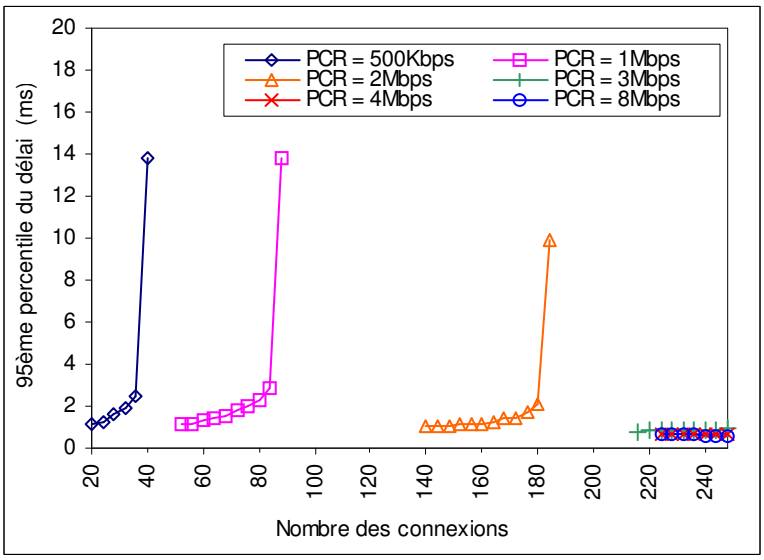


Figure 6.36 95ème percentile du délai des paquets AMR

La figure 6.37 représente l'utilisation maximale de la bande passante du VC pour différentes valeurs de PCR. L'utilisation représente aussi le facteur α. Pour des VC de faible PCR (500 kbit/s), le

facteur α est de l'ordre de 0,74. A partir d'une valeur de 1 Mbit/s de PCR, le facteur α atteint une valeur maximale de 0,8.

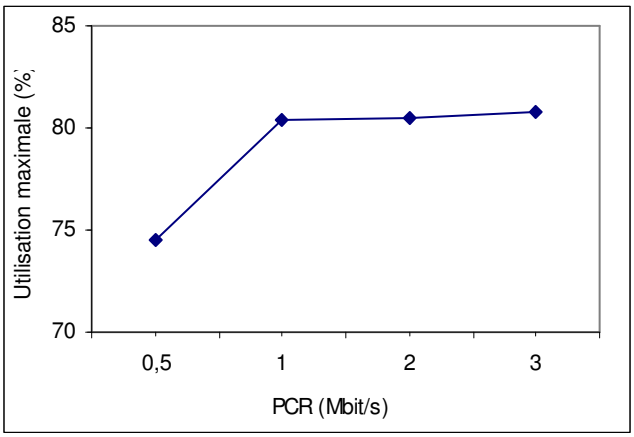


Figure 6.37 Utilisation maximale de la bande passante pour les flux AMR ou facteur α

La bande passante équivalente des flux AMR 12,2 est mesurée (9,7 kbit/s), le facteur α est aussi mesuré (0,8) pour des VC dont le PCR est plus grand que 1 Mbit/s. Il est alors facile d'établir une fonction de contrôle d'admission. Lors de la demande d'une nouvelle connexion AMR 12,2, si $9{,}7\times(N+1) \leq 0{,}8\times PCR$ alors la nouvelle connexion est acceptée, sinon elle est refusée. N est le nombre des connexions déjà établies.

6.5.2 Scénario 2: VC mono-service UDD 64 kbit/s

Nous avons procédé de la même manière pour mesurer la bande passante équivalente des flux UDD 64 kbit/s correspondant au trafic de *web browsing*. La figure 6.38 représente la bande passante équivalente à forte charge. Comme dans le cas précédent, pour des VC de faible PCR, la bande passante équivalente est plus grande. Elle atteint une valeur fixe de 9,3 kbit/s à partir d'un VC de 1,5 Mbit/s

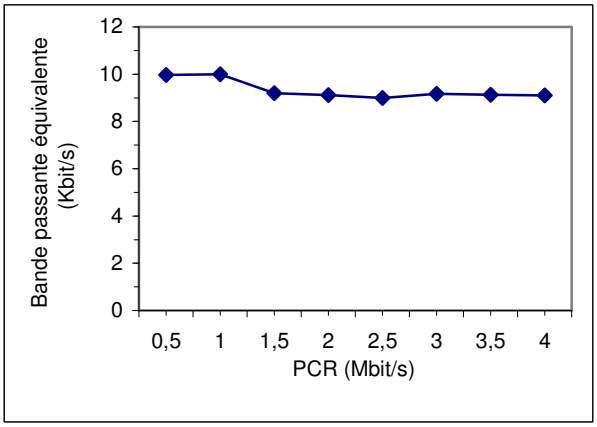


Figure 6.38 Bande passante équivalente pour les flux UDD 64 kbit/s

La figure 6.39 représente l'utilisation maximale de la bande passante qu'on peut atteindre en respectant les contraintes de délai pour les paquets UDD64. Pour des VC de faibles capacités, le taux d'utilisation ne peut pas dépasser 25%. A partir d'un VC de 2,5 Mbit/s, on peut atteindre une valeur fixe de 60%. Donc, le facteur α pour un VC transportant des flux UDD64 est de 0,6. Cette faible

valeur du facteur α est due au comportement très sporadique des flux UDD64. Quand la sporadicité augmente, on doit diminuer la charge d'un VC pour respecter les contraintes temporelles.

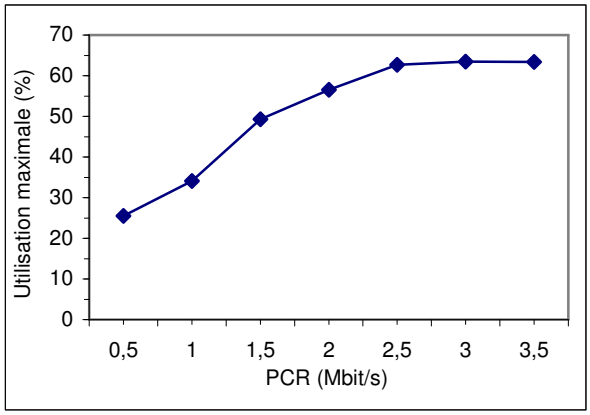


Figure 6.39 Utilisation maximale de la bande passante

6.6 Comparaison entre les capacités de transfert

Dans le chapitre 4, nous avons défini trois capacités de transfert au niveau AAL2 (AAL2-TC). Or, seulement deux AAL2-TC sont utilisables sur les interfaces Iub et Iur à cause des contraintes temporelles strictes de l'UTRAN : AAL2-CBR et AAL2-VBR (chapitre 5).

Au niveau ATM, nous avons proposé deux capacités de transfert (ATC) pour le transport des flux AAL2 dans l'UTRAN : DBR (*Deterministic Bit Rate*) et SBR (*Statistical Bit Rate*). En fait, puisque les flux dans l'UTRAN ont des contraintes temporelles strictes, la capacité de transfert choisie au niveau ATM doit être capable de garantir la qualité de service requise par les flux transportés. L'ATC ABR (*Available Bit Rate*) ne serait pas une bonne solution parce qu'elle ne peut pas garantir le délai de transit. De même pour l'ABT-DT (*ATM Block Transfer - Delayed Transmission*). L'ABT-IT (*ATM Block Transfer - Immediate Transmission*) peut garantir le délai mais il ne garantit pas le taux de perte. Alors il n'y a que le DBR et le SBR qui peuvent satisfaire les contraintes exigeantes de l'UTRAN.

Evidemment, Le DBR est la meilleure solution pour garantir la qualité de service parce qu'il fait une émulation de circuit. Son inconvénient est qu'il réserve une bande passante égale au débit maximal de la connexion et par conséquent, il ne bénéficie pas de l'effet statistique dans le cas des sources de trafic à débits variables. En effet, si la source de trafic est à débit variable, une partie de la bande passante réservée sera perdue et la capacité du lien ne sera pas utilisée d'une manière optimale. Le SBR comble cette faille du DBR en réservant les ressources selon un débit soutenu (*SCR: Sustainable Cell Rate*) et une taille maximale de rafale (*Burst*). Par conséquent, la partie de la bande passante non utilisée par une source à débit variable peut être ré-utilisée par d'autres sources et par suite, l'effet du gain statistique améliore l'utilisation de la bande passante disponible sur le lien. Dans ce paragraphe, nous allons présenter une comparaison entre un VC de type DBR et un autre de type SBR pour évaluer le gain statistique du VC SBR par rapport au VC DBR.

Un VC SBR ne peut pas être utilisé dans le cas où les flux AAL2 seraient transportés sur des connexions de type AAL2-CBR. Dans ce cas, le VC utilisé doit être de type DBR. Dans le cas où les flux AAL2 seraient transportés sur des connexions de type AAL2-VBR, le VC peut être de type SBR ou DBR.

La primitive *MAAL-Send.request* a pour rôle de donner l'autorisation à la couche CPS pour soumettre un CPS-PDU à la couche ATM. En fait, un CPS-PDU ne peut pas être envoyé vers la couche ATM sans cette autorisation même s'il est plein ou si le Timer-CU a expiré (voir le mécanisme

d'insertion des mini-cellules dans le chapitre 4). Cette primitive est envoyée avec une fréquence équivalente au débit PCR dans les deux cas des VC DBR et SBR.

Dans le cas du VC DBR, le contrôle de conformité des cellules ATM suit l'algorithme de conformité relatif au débit PCR décrit dans la recommandation UIT-T I.371 (voir Annexe A). Dans le cas d'un VC SBR, le contrôle de conformité est plus complexe et il doit tenir compte des débits PCR et SCR [66] (voir Annexe A). Pour qu'une cellule ATM soit conforme, elle doit satisfaire les algorithmes de conformité relatifs au VC correspondant.

Dans notre modèle de simulation, nous avons considéré deux configurations de VC :

- **Première configuration** : Des connexions AAL2-VBR sont transportées sur un VC DBR.
- **Deuxième configuration** : Des connexions AAL2-VBR sont transportées sur un VC SBR.

D'après le chapitre 5, le débit crête au niveau ATM (*PCR : Peak Cell Rate*) pour un ensemble de N connexions AAL2 peut être calculé pour les deux types de VC comme suit :

$$PCR = \frac{53}{47} \times \sum_{i=1}^{N} PMBR_i$$

Équation 6.1

où PMBRi est le débit crête au niveau AAL2 du flux "i".

Dans le cas du VC SBR, nous avons varié la valeur du débit SCR afin d'obtenir un taux de cellules non-conformes négligeable (voire nul). En effet, une cellule non-conforme peut être rejetée par les mécanismes de contrôle de flux. Par conséquent, nous avons considéré que le taux de perte des cellules ATM (*CLR: Cell Loss Ratio*) est négligeable. Nous avons considéré des scénarios différents et nous avons mesuré le SCR nécessaire pour chaque scénario pour que le CLR reste négligeable.

- **Scénario 1** : Les deux VC (DBR et SBR) sont mono-services homogènes transportant des flux AMR 12,2 kbit/s. Au niveau AAL2, le débit maximal d'un flux AMR 12,2 est de 16,8 kbit/s. En effet, une trame FP-PDU transportant un paquet AMR dans le sens montant est de 39 octets (32+5+2) (voir chapitre 3). Une mini-cellule AAL2 est constituée d'une trame FP et de 3 octets d'en-tête CPS. Les paquets AMR sont envoyés toutes les 20 ms pendant la période d'activité (ON). Donc, il y a une mini-cellule de 42 octets qui arrive chaque 20 ms. Nous avons considéré 248 connexions AAL2 (le nombre maximal dans un VC). Le débit crête peut être alors calculé d'après l'équation 6.1 : PCR = 248x16,8x53/47 = 4698,28 kbit/s, soit environ 4,7 Mbit/s. C'est le débit PCR considéré pour les deux VC.
- **Scénario 2** : Les deux VC sont mono-services homogènes transportant des flux UDD 64 kbit/s. Quand une source UDD émet à débit crête, la taille d'une trame FP-PDU est de 339 octets (voir chapitre 3). Cette trame est décomposée en 8 mini-cellules dont 7 ont une taille maximale de 48 octets et une de taille 27 octets. Ces mini-cellules sont envoyées durant un TTI de 40 ms d'où le débit maximal au niveau AAL2 est de 72,6 kbit/s. Nous avons considéré aussi 248 connexions d'où le débit PCR est de 20,303 Mbit/s.
- **Scénario 3** : Les deux VC sont de type multi-service transportant des flux AMR 12,2 et des flux UDD 64. Nous avons considéré 3 cas:
    - **Cas (a)** - 80% de trafic AMR et 20% de trafic UDD : Dans ce cas, le débit crête PCR est de 7,24 Mbit/s.
    - **Cas (b)** - 20% de trafic AMR et 80% de trafic UDD : Le PCR est de 16,402 Mbit/s.
    - **Cas (c)** - 50% de trafic AMR et 50% de trafic UDD : Le PCR est de 11,2484 Mbit/s.

La répartition des flux dans les 3 cas dépend du pourcentage du trafic de chaque type de flux en terme de bande passante.

Dans tous les scénarios, le débit SCR est choisi de telle manière que le taux de perte (CLR) soit négligeable et le délai de transfert soit dans les limites autorisées dans le contexte de l'UTRAN (5 ms pour les paquets AMR et 50 ms pour les paquets UDD). Le tableau 6.1 illustre les résultats obtenus pour la valeur du SCR convenable. Le gain statistique est le rapport entre le PCR et le SCR (PCR/SCR).

Scénario	**1**	**2**	**3.a**	**3.b**	**3.c**
PCR (Mbit/s)	4,7	20,303	7,24	16,402	11,2484
SCR (Mbit/s)	2,92	3,505	3,129	3,6356	3,208
Gain statistique	***1,6***	***5,8***	***2,3***	***4,5***	***3,5***

Tableau 6.1 SCR d'un VC SBR

Dans tous les scénarios, le taux de remplissage des cellules était de l'ordre de 99%. Le nombre d'utilisateurs étant élevé, le multiplexeur AAL2 donne un gain statistique assez important surtout dans le cas où le trafic des données serait majoritaire. Le gain statistique devient intéressant avec les flux sporadiques comme les sources UDD. Dans le tableau ci-dessus, nous pouvons remarquer l'avantage de la deuxième configuration où nous utilisons des connexions AAL2-VBR sur un VC de type SBR. Par rapport à la première configuration, la deuxième configuration peut fournir un gain qui peut atteindre 5,8 pour des sources très sporadiques.

On a examiné aussi le cas d'un faible nombre d'utilisateurs. On a considéré un VC mono-service supportant 20 utilisateurs de voix. Pour un taux de perte négligeable, le gain statistique était de 1,1. Le SBR présente donc peu d'intérêt dans ce cas.

Un autre VC mono-service pour les données est considéré avec une faible charge (20 utilisateurs UDD64 kbit/s). Le gain statistique est alors proche de 1. Le SBR ne présente aucun d'intérêt dans ce cas.

En conclusion, le mode SBR est approprié dans le cas où le nombre d'utilisateurs serait élevé, c'est à dire lorsque l'effet statistique est important au niveau du multiplexeur CPS. Dans ce cas, on peut optimiser l'utilisation de la bande passante disponible en faisant une réservation suivant le SCR tout en gardant un taux de perte négligeable. L'effet statistique est très clair surtout dans le cas où le trafic des données serait majoritaire, à cause de la sporadicité marquée des sources des données.

6.7 La commutation AAL2

Dans la technique ATM, les mécanismes de brassage ou de commutation n'étaient présents qu'au niveau de la cellule ATM : brassage de VP, commutation de VC. L'analyse de l'entête ATM et en particulier des champs VPI et VCI est suffisante pour aiguiller une cellule dans le réseau. La couche d'adaptation AAL2 introduit un niveau supplémentaire de commutation : les mini-cellules AAL2 contenues dans une cellule ATM possèdent chacune un identifiant CID (*Channel IDentifier*) qui distingue une connexion AAL2 d'une autre. Il est donc nécessaire de remonter une couche pour réaliser la commutation des mini-cellules, plus précisément la sous-couche CPS de l'AAL2.

6.7.1 Rappels sur la commutation ATM

Les connexions virtuelles établies dans un commutateur ATM sont de deux types : les connexions permanentes ou semi-permanentes (mode PVC et *Soft* PVC) et les connexions commutées (mode SVC). Les premières sont établies par des fonctions de gestion ou d'administration et sont mises en œuvre dans des équipements appelés brasseurs. Les secondes sont établies appel par appel par des fonctions de commande pilotées par l'organe de traitement des protocoles de signalisation et sont mises en œuvre dans les commutateurs de VC. En mode SVC, les deux sens de la connexion sont établis simultanément, les paramètres de trafic et de QoS peuvent être différents par sens. Ces attributs sont contenus dans les messages échangés au cours de la procédure d'appel. En mode PVC, les deux sens sont également établis simultanément par l'administrateur, avec des caractéristiques propres. Cette différence possible dans les procédures de marquage est illustrée sur les deux schémas de la figure 6.40.

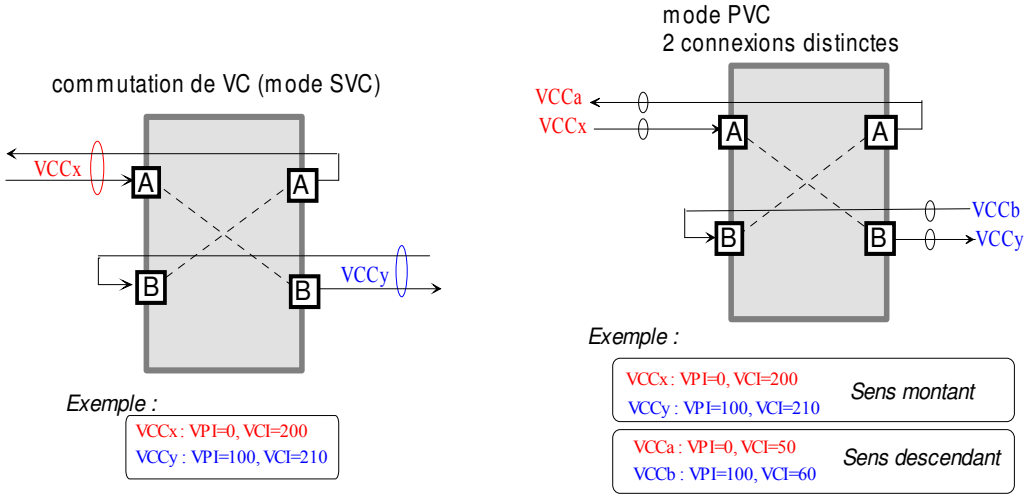


Figure 6.40 Principe de la commutation ATM

6.7.2 Le principe de la commutation AAL2

Le principe de la commutation AAL2 est illustré sur la figure 6.41. Chaque canal AAL2 est identifié par un triplet (VPI, VCI, CID). Une commutation AAL2 consiste à faire une association entre un triplet d'entrée est un triplet de sortie du commutateur:

$$f : (\mathrm{VPI}x, \mathrm{VCI}a, \mathrm{CID}i) \rightarrow (\mathrm{VPI}y, \mathrm{VCI}e, \mathrm{CID}p)$$

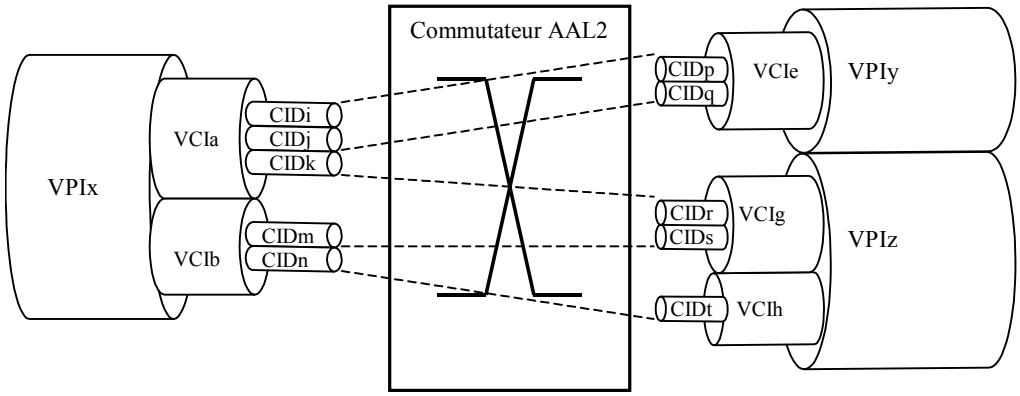


Figure 6.41 Principe de la commutation AAL2

Un nœud de commutation AAL2 a pour rôle de commuter des paquets CPS (des mini-cellules) puisque la notion de canal n'existe qu'au niveau CPS. Par conséquent, dans un commutateur AAL2, il suffit d'implémenter la sous-couche CPS et on n'a pas besoin de la sous-couche SSCS.

Les connexions AAL2 transportent des services dont les flux peuvent être quelconques en terme de symétrie :

- Symétrie en débit : les flux montants et descendants sont les mêmes (c'est le cas des communications conversationnelles).
- Asymétrie des débits : c'est par exemple le cas de la consultation Internet.
- Un seul sens : c'est le cas de la diffusion d'information.

Afin d'utiliser les ressources du réseau de façon optimale, on peut être tenté de tirer parti de la possibilité de gérer les connexions ATM et AAL2 d'une façon comparable à celle mise en œuvre pour le mode PVC, c'est à dire par demi-connexions au niveau VCC et au niveau AAL2. Des VCC de transport unidirectionnels seraient dédiés aux flux AAL2 lorsqu'ils ne sont que montants ou descendants. Des VCC bidirectionnels (symétriques ou asymétriques en débits) seraient dédiés aux flux des communications bidirectionnelles. Les VCC de transport des connexions AAL2 peuvent être établis en mode PVC ou en mode SVC.

6.7.3 Architecture d'un commutateur AAL2

La commutation de mini-cellules peut être fonctionnellement découpée en trois opérations : extraction des mini-cellules (réception), commutation proprement dite et insertion des mini-cellules (émission). Le schéma de la figure 6.42 représente l'architecture fonctionnelle d'un commutateur AAL2. Le module d'extraction a pour rôle d'extraire les mini-cellules à partir des cellules ATM et de reconstruire les mini-cellules qui chevauchent sur deux cellules consécutives. Après cela, il envoie la mini-cellule vers le module de commutation. Ce module assure la translation des valeurs CID selon la table de commutation puis il envoie les mini-cellules commutées vers la file d'attente du VC correspondant. Le module d'insertion assure l'insertion des mini-cellules dans les cellules ATM du VC et établit toutes les fonctions relatives au mécanisme de multiplexage au niveau CPS (Timer-CU, bourrage, etc.).

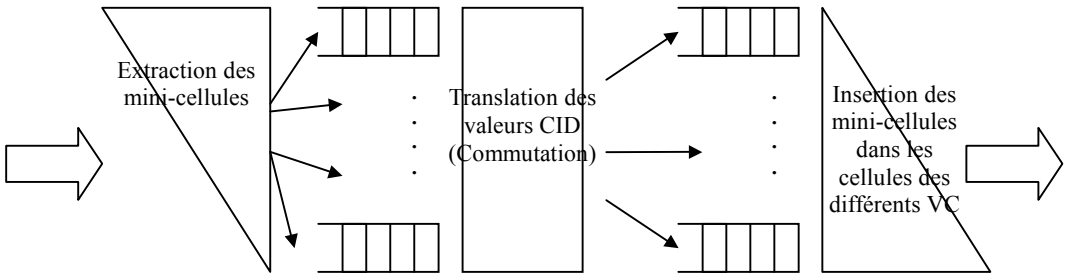


Figure 6.42 Architecture fonctionnelle d'un commutateur AAL2

6.7.4 Les besoins de la commutation AAL2 dans l'UTRAN

La commutation AAL2 présente des inconvénients importants comme les délais supplémentaires dus aux mécanismes d'insertion et d'extraction des mini-cellules. Cet inconvénient est pénalisant dans le cas de l'UTRAN surtout que les délais de transfert sont très stricts. En revanche, si on peut limiter ce délai par des équipements puissants où le temps de traitement serait faible, la commutation AAL2 présente un avantage important qui est l'utilisation efficace des liens du réseau.

En effet, la commutation AAL2 permet de concentrer davantage les flux AAL2 dans les VC ATM. Dans l'UTRAN, cet effet de concentration est important dans le cas où le trafic sortant des Node B serait faible. En fait, si la distante entre un Node B et un RNC est assez élevée, le coût du lien devient assez cher. Si le VC sortant du Node B est faiblement chargé, l'utilisation du lien sera faible ce qui entraîne un gaspillage des ressources physiques. Dans ce cas, on peut agréger les flux de plusieurs Node B dans un concentrateur proche pour sortir sur un seul lien vers le RNC éloigné et par conséquent, on économise la bande passante qui est une ressource chère. Ce concentrateur peut être un commutateur AAL2. Ce cas se présente quand le lien entre le Node B et le RNC (l'interface Iub) n'est pas direct, mais il passe à travers un réseau commuté comme le montre la figure 6.43.

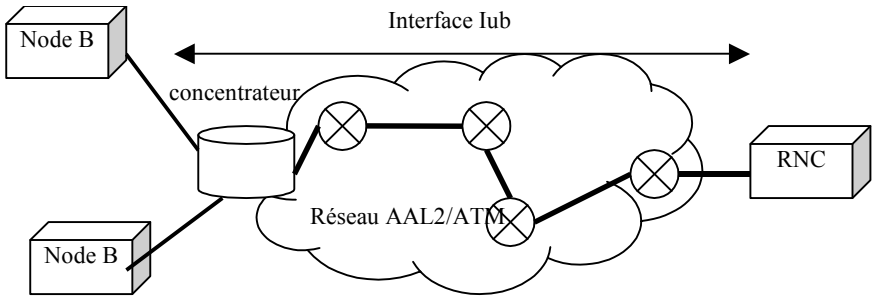


Figure 6.43 Concentration des flux sur l'interface Iub

De toute façon, la commutation AAL2 est indispensable dans certains cas. Considérons par exemple un utilisateur dans un état de *soft handover* (figure 6.44). Le VC reliant le Node B au D-RNC contient des flux ayant des destinations différentes. Le flux de l'utilisateur en *soft handover* doit être acheminé vers le S-RNC pour réaliser les mécanismes relatifs à la macro-diversité. Dans ce cas, il est nécessaire de séparer les flux AAL2 du VC entrant et commuter les mini-cellules vers le S-RNC. Le D-RNC doit disposer d'un commutateur AAL2. Dans le cas où tous les flux AAL2 d'un même VC devraient être acheminés vers une même destination, un commutateur ATM est suffisant.

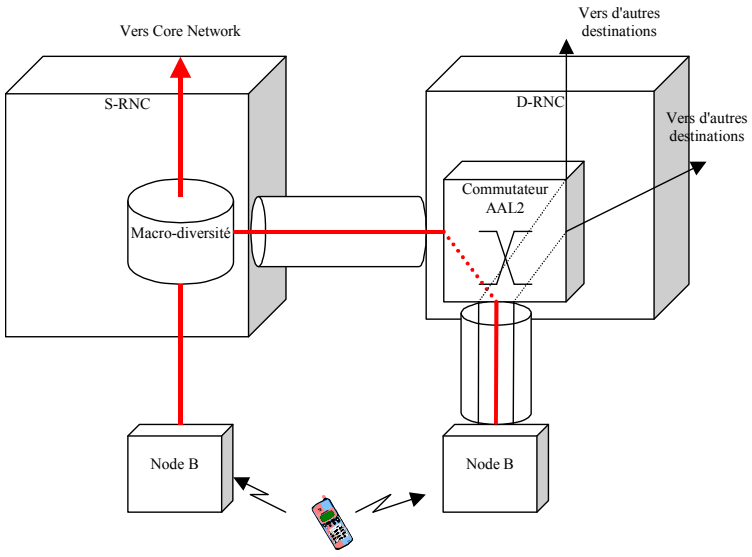


Figure 6.44 Besoin d'un commutateur AAL2 dans le cas de *soft handover*

6.7.5 Schémas d'agrégation des flux dans l'UTRAN

Dans le cas d'un concentrateur entre les Node B et le RNC, plusieurs flux peuvent être agrégés dans un même tuyau de sortie. Cela est intéressant dans le cas où les tuyaux entre les Node B et le

concentrateur sont faiblement chargés. La figure 6.45 représente les différentes approches pour la gestion des flux dans l'UTRAN.

Dans le schéma A, aucune agrégation n'est effectuée, ni au niveau ATM ni au niveau AAL2. Au niveau ATM, un brassage est effectué pour les VP entrant dans le nœud ATM qui est un brasseur ATM. Chaque VP entrant est brassé vers un VP sortant correspondant. Ce schéma peut se présenter dans le cas d'un passage par plusieurs nœuds avant d'atteindre le RNC. Dans ce cas, toutes les connexions AAL2 ont la même source (même Node B de départ) et la même destination (même RNC d'arrivée). Dans ce schéma, si les VP entrant sont faiblement chargés, ils garderont la même charge tout au long de leur chemin et aucune optimisation de la bande passante ne peut être réalisée. Ce schéma ne permet pas de profiter de l'effet de multiplexage et par suite, il ne sera pas étudié.

Le schéma B représente le cas où une commutation de VC est effectuée dans le nœud de concentration (commutateur ATM). Dans ce cas, les VC entrants sont extraits de leurs VP puis commutés vers un seul VP de sortie. Si les VC/VP entrants sont faiblement chargés, on peut profiter du gain statistique lors du multiplexage de plusieurs VC dans un même VP. L'avantage de cette solution est qu'elle ne nécessite pas de remonter jusqu'au niveau AAL2 et une simple commutation ATM est suffisante. L'inconvénient de cette solution est qu'elle ne peut pas éliminer les octets de bourrage dans les cellules ATM transportant des flux AAL2. En effet, si les VC entrants sont faiblement chargés, alors leurs cellules peuvent être partiellement remplies si le Timer-CU dans le Node B a une faible valeur. Puisqu'on commute au niveau ATM, alors il est impossible de regagner la bande passante perdue en terme de bourrage. De toute façon, le gain de multiplexage qu'on peut gagner en agrégeant plusieurs VC dans un même VP est important.

Le schéma C utilise une commutation AAL2. Les connexions AAL2 entrantes sont extraites de leurs VC/VP puis elles sont multiplexées de nouveau dans un seul VC/VP de sortie. L'avantage essentiel de cette solution est l'effet important du gain statistique. En effet, puisque les mini-cellules AAL2 sont extraites des cellules ATM puis ré-insérées de nouveau dans des nouvelles cellules, le nombre d'octets de bourrage peut être réduit surtout que tous les flux entrants sont multiplexés dans un même VC. Ceci peut augmenter le taux de remplissage et optimiser l'utilisation de la bande passante sur le lien entre le commutateur et le RNC. Ce schéma est plus avantageux que le schéma B en terme d'utilisation optimale de la bande passante mais il présente l'inconvénient des délais supplémentaires dus à la commutation AAL2.

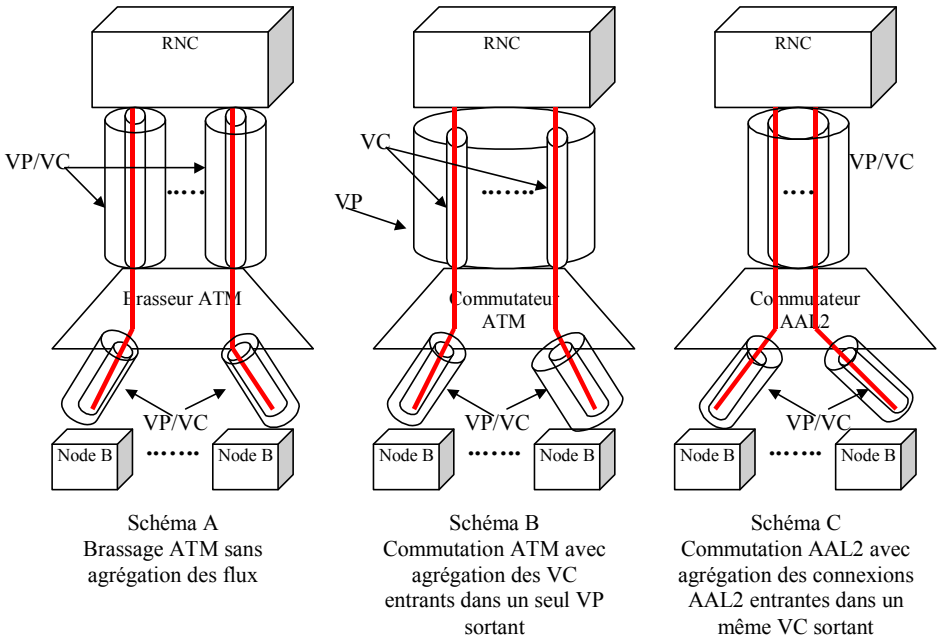


Figure 6.45 Les différentes approches d'agrégation des flux dans l'UTRAN

6.7.6 Comparaison entre un commutateur AAL2 et un autre ATM

Dans ce paragraphe, nous allons présenter une comparaison entre la commutation AAL2 et la commutation ATM dans les cas des schémas B et C. Les critères de comparaison sont le délai et l'utilisation de la bande passante entre le concentrateur et le RNC.

Dans le modèle de simulation, nous avons considéré des Node B reliés à un concentrateur par des VP/VC dont le PCR est de 2 Mbit/s. Le concentrateur peut être un commutateur AAL2 ou ATM dans lequel tous les flux venants des Node B sont agrégés dans un même VP/VC de sortie de PCR 2 Mbit/s. Evidemment, les VP/VC entrants sont faiblement chargés. Dans le cas d'un commutateur ATM, tous les VC entrants sont multiplexés dans le même VP de sortie, le multiplexage se fait au niveau des cellules ATM. Dans le cas d'un commutateur AAL2, les mini-cellules AAL2 sont extraites des cellules puis multiplexées de nouveau dans les cellules de sortie. Les connexions AAL2 sont alors multiplexées dans le VC/VP de sortie. Le commutateur a une vitesse de commutation de 20 Mbit/s. Le nombre de Node B agrégés est un paramètre variable. Concernant les flux transportés, nous avons considéré deux scénarios:

- **Scénario 1** : Tous les VC entrants sont de type mono-service homogène et transportent des flux AMR 12,2 kbit/s. Ils ont tous la même charge de 10% environ.
- **Scénario 2** : Tous les VC entrants sont de type mono-service homogène et transportent des flux UDD 64 kbit/s. Ils ont tous la même charge de 10% environ.

Nous avons mesuré le délai des paquets (ASTD) ainsi que l'utilisation de la capacité du VP sortant et le taux de remplissage en fonction du nombre des Node B agrégés.

6.7.6.1 Résultats du scénario 1

La figure 6.46 représente le 95ème percentile du délai des paquets AMR en fonction du nombre des Node B. Pour une valeur de Timer-CU de 1 ms dans tous les multiplexeurs CPS et pour 5 Node B par exemple, le délai des paquets est de 2,27 ms dans le cas d'un commutateur AAL2 et de 1,23 ms dans le cas d'un commutateur ATM. La différence est de 1,04 ms. Dans le cas d'un Timer-CU nul dans tous les multiplexeurs CPS et pour 5 Node B, le délai est de 0,48 ms dans le cas d'un commutateur AAL2

et de 0,3 ms dans le cas d'un commutateur ATM. La différence est de 0,18 ms. Quelle que soit la valeur du Timer-CU, le délai est toujours plus élevé dans le cas de la commutation AAL2 à cause du délai supplémentaire introduit par les mécanismes d'extraction et d'insertion des mini-cellules dans le commutateur AAL2. Dans le cas d'une valeur nulle du Timer-CU dans tous les multiplexeurs, la différence entre les délais dans les cas des deux commutateurs est faible. D'ailleurs, dans le cas d'une valeur de 1 ms du Timer-CU dans tous les VC, le délai dû au Timer-CU du multiplexeur de sortie du commutateur vient s'ajouter au délai dû au Timer-CU du multiplexeur du Node B. C'est pourquoi la différence de délai entre les deux commutateurs dans le cas d'un Timer-CU de 1 ms est élevée.

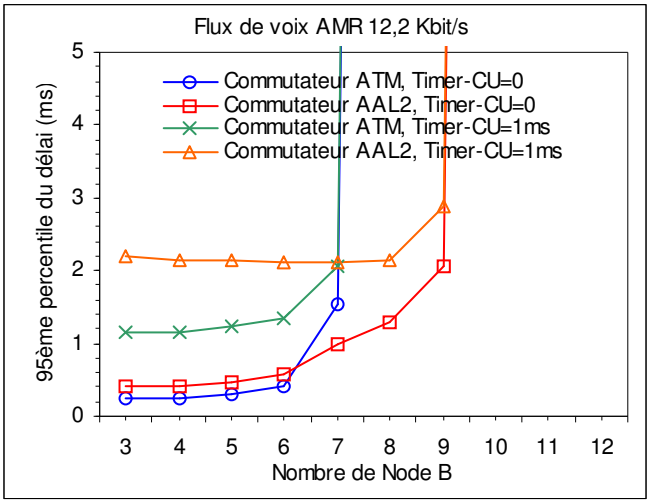


Figure 6.46 Scénario 1: 95ème percentile du délai

En revanche, dans le cas du commutateur AAL2, on peut agréger un nombre plus grand de Node B. Pour le commutateur ATM, le délai dépasse la limite autorisée à partir de 7 Node B. Par contre, dans le cas du commutateur AAL2, on peut agréger jusqu'à 9 Node B tout en respectant un délai acceptable. En fait, les cellules entrantes sont partiellement remplies parce que le trafic est faible sur les VC des Node B. Dans le cas de la commutation ATM, ces cellules sont envoyées sur le VC de sortie avec le même taux de remplissage tandis que dans la commutation AAL2, le taux de remplissage est amélioré grâce au multiplexage de tous les flux AAL2 venants de tous les Node B. Le taux de remplissage est représenté sur la figure 6.47. Il est clair que la commutation AAL2 augmente considérablement le taux de remplissage des cellules ATM sur le VC sortant.

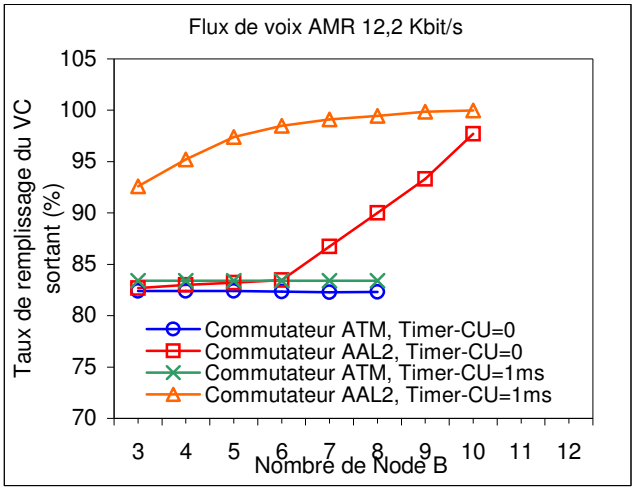


Figure 6.47 Scénario 1 : Taux de remplissage du VC sortant

La charge du VC sortant est représentée sur la figure 6.48. Avec un commutateur AAL2, on a besoin de moins de ressources sur le VC reliant le commutateur au RNC.

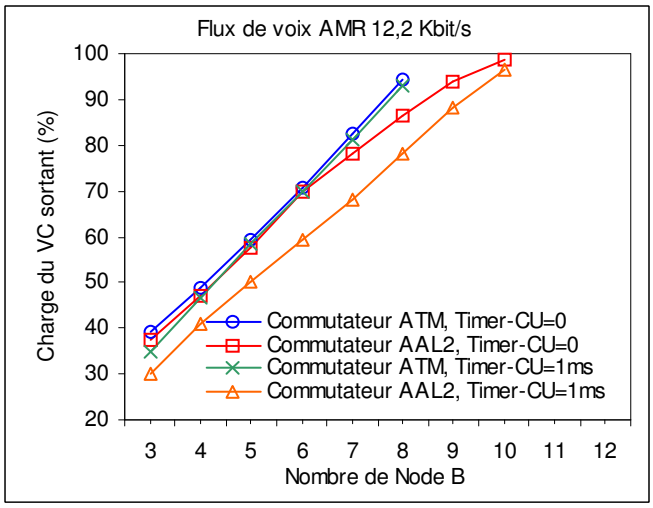


Figure 6.48 Scénario 1: Charge du VC sortant

6.7.6.2 Résultats du scénario 2

La figure 6.49 représente le délai des paquets UDD 64 ainsi que la figure 6.50 représente le taux de remplissage et la charge du VC sortant. L'effet de la commutation AAL2 est moins clair dans le cas de transport des paquets de grandes tailles comme les paquets UDD. En effet, les cellules ATM sortant du Node B sont quasiment pleines même si le trafic est faible. La commutation AAL2 n'apporte pas alors de grands bénéfices en terme d'utilisation de la bande passante du VC sortant.

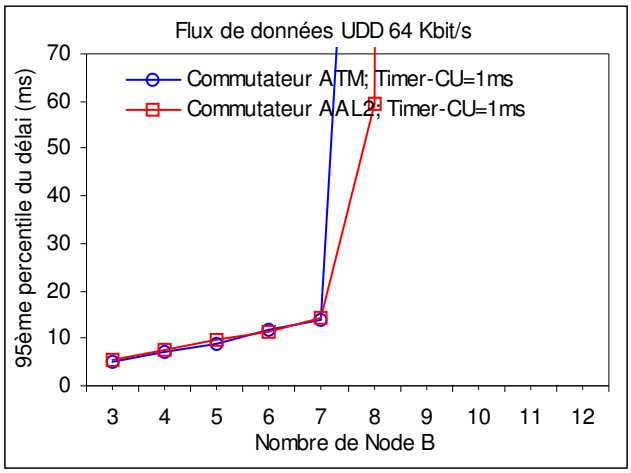


Figure 6.49 Scénario 2: 95ème percentile du délai

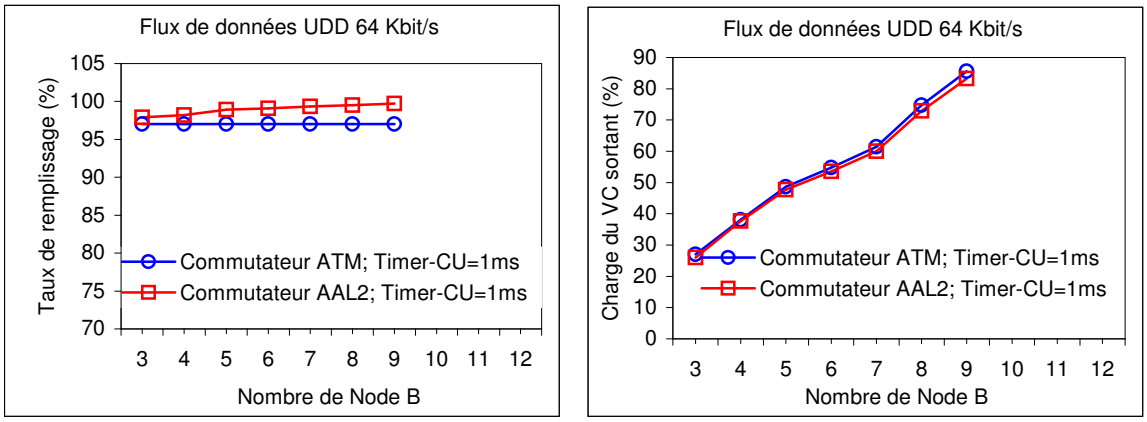


Figure 6.50 Scénario 2: Taux de remplissage et charge du VC sortant

6.7.7 Conclusions sur la commutation AAL2

La commutation AAL2 est indispensable dans le cas où les connexions AAL2 du VC n'ont pas toutes la même destination. Son inconvénient est qu'elle introduit des délais supplémentaires à cause des mécanismes d'extraction et d'insertion des mini-cellules.

Dans le cas d'un nœud de concentration, le choix de la commutation AAL2 peut être avantageux si les liens entrants sont faiblement chargés. Les inconvénients des délais supplémentaires introduits par la commutation AAL2 sont récompensés par les bénéfices de l'augmentation du taux de remplissage et par conséquent, l'optimisation de l'utilisation de la bande passante.

Si les liaisons d'entrée sont fortement chargées ou si le taux de remplissage des VC entrants est assez élevé, la commutation AAL2 n'apporte pas des bénéfices importants. En effet, on ne peut pas bénéficier des avantages de la commutation AAL2 parce que la bande passante offerte à la couche AAL2 est déjà bien utilisée. Dans ce cas, la commutation ATM donne des bonnes performances surtout qu'elle est moins complexe et évite l'introduction des délais supplémentaires. Elle est alors recommandée dans ce cas.

6.8 Qualité des résultats de simulation

Le problème des simulations réside dans les générateurs des nombres aléatoires. En effet, si on demande par exemple à un générateur d'entiers uniformément réparties dans l'intervalle [0, 999999] de nous produire un nombre et qu'il nous sort 999999 comme résultat, quelqu'un peut penser que le résultat n'est pas tellement aléatoire. Et pourtant, ce résultat est aussi bon que 210279, ou n'importe quel autre entier. Ceci montre qu'on ne peut pas parler de nombre aléatoire pour une valeur obtenue à partir d'une procédure où d'une table. Par contre, cela a un sens de parler d'une séquence de nombres aléatoires. En effet, si notre générateur nous sort 10 fois de suite la même valeur, on devrait commencer à se poser des questions. Malheureusement, il est clair qu'un ordinateur pourra seulement produire une séquence finie de nombres qui sera répétée. Une telle séquence n'est évidemment pas aléatoire. Les séquences produites par ces ordinateurs se comportent dans un certain sens comme étant aléatoires. C'est pourquoi, on les appelle des séquences "pseudo-aléatoires".

Une simulation consiste à construire une ou plusieurs trajectoires du modèle et à analyser statistiquement les données ainsi obtenues. Pour un paramètre à mesurer, on utilise un estimateur qui peut être par exemple la valeur moyenne de toutes les valeurs générées le long de la trajectoire. Evidemment, si on construit une autre trajectoire, la valeur obtenue de l'estimateur sera différente, un estimateur étant en soi une variable aléatoire. C'est pour cette raison qu'en général, on ne se contente pas d'avoir uniquement un estimateur, mais on cherche aussi un intervalle de confiance pour cet estimateur. L'intervalle de confiance diminue quand le nombre d'échantillons augmente. Mais on ne peut pas générer un nombre infini d'échantillons vue le temps de calcul qui devient inadmissible. Néanmoins, le nombre d'échantillons générés doit être suffisant pour que l'intervalle de confiance soit le plus petit possible et par suite, les résultats de simulation soient les plus proches des valeurs exactes.

Pour nos simulations, le paramètre critique est le délai de transfert. Nous avons calculé la valeur moyenne des délais moyens pour les différents utilisateurs. Puisque les générateurs aléatoires implémentés dans les différentes sources de trafic sont indépendants, on peut calculer un intervalle de confiance en utilisant la formule déduite du théorème central limite [3]:

$$\Pr(d \in \left[\overline{D(n)} - t_{n-1,0.975}\sqrt{\frac{S^2(n)}{n}}, \overline{D(n)} + t_{n-1,0.975}\sqrt{\frac{S^2(n)}{n}} \right] = 0.95$$

$\overline{D(n)}$ est la valeur moyenne des D_i qui représentent les délais moyens des différents utilisateurs (n est le nombre de populations qui est ici le nombre d'utilisateurs) tandis que $S^2(n)$ est sa variance. t est le point critique calculé d'après le tableau présenté dans [3] pour une distribution normale.

Pour démontrer la précision de nos résultats de simulation, nous prenons un exemple critique qui est la figure 6.5 où nous comparons les délais de transfert des paquets AMR pour différentes valeurs du Timer-CU. Dans le cas de 10 utilisateurs AMR, le délai est de 0,19 ms pour un Timer-CU nul et de 1,9 ms pour un Timer-CU infini. Le problème qui se pose est si la différence est significative vu la précision des mesures. Pour un Timer-CU nul, la moyenne obtenue est de 0,19 tandis que la variance est de 0,0025. L'intervalle de confiance est donc : $\pm 1{,}83 \times 0.016 = 0.029$. Dans le cas du Timer-CU infini, l'intervalle de confiance est de 0,1. L'intervalle de confiance est beaucoup plus petit que la différence mesurée.

Ce raisonnement est fait sur les différents cas de simulation pour montrer que l'intervalle de confiance est toujours acceptable. Notre stratégie de simulation est la suivante : puisque le nombre de sondages indépendants est égal au nombre d'utilisateurs (minimum de 10 dans toutes nos simulations), la longueur d'un sondage augmente quand le nombre de sondage diminue de façon à avoir une valeur minime de l'intervalle de confiance. Les durées de simulation choisies sont assez longues afin

d'obtenir un nombre d'échantillons suffisant (grande population) pour que l'intervalle de confiance soit le plus petit possible et par suite, les résultats de simulation soient fiables et de bonne qualité.

6.9 Conclusion

Dans ce chapitre, nous avons étudié plusieurs aspects liés aux performances du protocole AAL2 pour le transport des flux dans l'UTRAN. Nous avons étudié le Timer-CU et nous concluons qu'une valeur comprise entre 1 ms et 2 ms est une valeur optimale dans le contexte de l'UTRAN. Le mécanisme d'ordonnancement WRR semble une bonne solution, mais son problème est qu'on doit connaître en avance la répartition des trafics pour choisir les poids. Pour cela, nous avons proposé un nouvel algorithme de type WRR dont les poids changent dynamiquement en fonction de la charge de chaque flux. En plus, ce nouvel algorithme appelé DyWRR prend en compte les contraintes temporelles de chaque flux lors du calcul des poids.

L'implémentation d'un ATC SBR présente des profits à faible charge surtout lorsque le trafic est très sporadique. La commutation AAL2 est intéressante dans le cas d'un concentrateur où les liens entrants sont faiblement chargés. Cependant, elle introduit des délais supplémentaires mais ces délais sont récompensés par le gain statistique important que peut apporter la commutation AAL2 par rapport à la commutation ATM.

Les résultats de l'étude faite dans ce chapitre pourront être utiles pour les opérateurs des réseaux mobiles. En fait, ces résultats pourront aider les opérateurs dans le cadre de dimensionnement de leurs réseaux d'accès et surtout dans la première phase de déploiement des réseaux UMTS où les réseaux d'accès UTRAN sont introduits par îlots. Dans ce cas, les ressources de l'infrastructure seront chères et une utilisation efficace de ces ressources sera nécessaire tout en respectant les contraintes temporelles de l'UTRAN. Ceci ne peut pas être réalisé par des méthodes de surdimensionnement et par suite, un dimensionnement rigoureux de l'infrastructure pourra assurer le bon fonctionnement de ces réseaux.

Chapitre 7

7 Le transport en IP dans l'UTRAN*

7.1 Introduction

Dans la Release 5 du 3GPP, le protocole IP [84] a été introduit comme technique de transport dans l'UTRAN en parallèle avec la technologie AAL2/ATM [92]. Le choix de l'IP dans l'UTRAN vient dans le but de déploiement d'un réseau tout-IP. En effet, l'IP est une technologie de transport qui arrive jusqu'au terminal et la plupart des applications seront basées sur cette technologie. L'IP assure le support d'une grande variété de types de trafic dans l'Internet dont la popularité rend le coût des équipements assez faible. Les opérations de maintenance du réseau seront supportées par une technologie IP et par suite le déploiement des réseaux avec une technologie homogène peut réduire le coût des opérations de gestion.

Le trafic IP va dominer dans les réseaux mobiles à cause des applications et services multimédia proposés par l'UMTS. Le choix de l'opérateur entre une technologie IP et une autre AAL2/ATM dans le réseau d'accès constitue une des problématiques essentielles.

Il est préférable d'utiliser les protocoles déjà normalisés pour l'IP (par l'IETF) quand c'est possible, pour éviter des nouveaux travaux et pour avoir une large compatibilité avec les autres réseaux existants. L'introduction de l'IPv6 est envisagée à coté de l'IPv4. Les protocoles liés à la gestion de la mobilité dans un domaine IP sont aussi envisagés comme *Mobile IP*, *Cellular IP*, etc.

L'introduction de l'IP va affecter seulement la couche de transport (TNL) et ne doit pas introduire des modifications majeures dans les couches radio (RNL) parce que les deux couches sont indépendantes. En fait, si la couche radio dépend fortement de la technologie de transport, la compatibilité entre des réseaux de technologies de transport différentes sera très difficile à réaliser.

En tenant compte des contraintes strictes de l'UTRAN, le protocole IP doit garantir la qualité de service nécessaire pour le transport des canaux radio sur les interfaces Iub et Iur. Pour accomplir une telle tâche, une différenciation des services est nécessaire dans le réseau de transport IP.

Un autre aspect important est l'efficacité de l'utilisation des liens dans l'UTRAN. En effet, les Node B sont connectés aux RNC à travers des réseaux dorsaux (*Backbone*) basés sur la technologie IP. Chaque Node B est connecté à ce réseau à travers un lien appelé *Last Mile Link* qui connecte le Node B au plus proche nœud du réseau appelé *Edge Router*. C'est sur cette partie du réseau où la bande passante est chère et son utilisation doit être efficace. Les opérateurs des réseaux des télécommunications s'intéressent au dimensionnement de cette partie du réseau pour profiter au maximum des ressources disponibles.

La problématique de l'introduction de l'IP dans l'UTRAN est un sujet très vaste et les études ne sont pas encore finalisées. Nous avons alors focalisé notre étude sur plusieurs solutions pour le transport en IP et l'efficacité de chacune d'elles sur le *Last Mile Link* de l'interface Iub.

---

* Ce chapitre fait partie d'une étude réalisée dans le cadre d'un projet bilatéral entre l'ENST et France Télécom R&D. En tenant compte de l'aspect confidentiel de ce projet, les résultats ne seront pas publiés dans cette thèse suite à la demande du coordinateur du projet. Seulement les études préliminaires seront publiées avec quelques informations concernant le modèle de simulation utilisé.

7.2 Rappel sur les protocoles de transport

Dans ce paragraphe, nous allons décrire brièvement les protocoles de transport qui sont utilisés dans les différentes architectures de l'interface Iub dans un UTRAN basé sur la technologie de transport IP.

7.2.1 Le protocole UDP (*User Datagram Protocol*) [RFC 768]

C'est un ancien protocole standardisé en 1980 pour permettre le transport des données en mode datagramme dans les réseaux à commutation de paquets [82]. Ce protocole suppose que le protocole utilisé au-dessous est le protocole IP.

UDP fournit la possibilité d'envoyer des données entre les applications avec un minimum de traitement. Il ne garantit pas le transfert des données. Le format d'un datagramme UDP est présenté sur la figure 7.1.

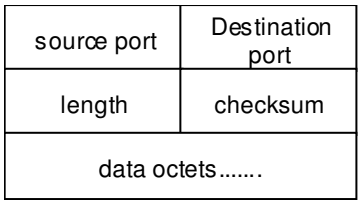


Figure 7.1 Format d'un datagramme UDP

Le champ "*Source Port*" de 2 octets de longueur est un champ optionnel. Il indique le numéro de port du processus source. Le champ "*Destination Port*" de 2 octets désigne l'adresse du port du processus destination. Le champ "*Length*" de 2 octets désigne la longueur en octets du datagramme y compris l'en-tête et les données. Le champ "*Checksum*" de 2 octets est utilisé pour la détection d'erreur sur tous les bits du datagramme UDP.

7.2.2 Le protocole IP (*Internet Protocol*)

Le protocole IP est conçu pour interconnecter les machines dans un réseau à commutation de paquets. Il traite chaque datagramme comme une entité indépendante de tout autre datagramme. Il n'y a pas de notion de connexion ni de circuit logique. Il existe deux versions du protocole IP: IPv4 et IPv6.

7.2.2.1 IP Version 4 [RFC 791]

La figure 7.2 représente le format de l'en-tête d'un paquet IP version 4 [84]. Le champ "*version*" de 4 bits indique la version du protocole IP. Ce champ a une valeur de 4. Le champ "IHL (*Internet Header length*)" de 4 bits indique la longueur de l'en-tête IP. Cette longueur est mesurée en mots de 32 bits (4 octets). Cette valeur indique le début du champ des données. La taille minimale d'un en-tête correct est de 5 mots (20 octets). Le champ "ToS (*Type of Service*)" de 8 bits indique la qualité de service du paquet IP. Cette valeur est utilisée dans les nœuds du réseau pour appliquer des traitements particuliers aux paquets prioritaires. Le champ "*Total length*" indique la longueur en octets du paquet IP y compris l'en-tête. Ce champ est composé de 16 bits et autorise une longueur maximale de paquet de 65535 octets. Le champ "*Identification*" de 16 bits est un identificateur du paquet IP. Il permet le ré-assemblage du paquet en cas de fragmentation dans le réseau. Le champ "*Flag*" de 3 bits est utilisé pour le contrôle de la fragmentation. Le champ "TTL (*Time To Live*)" de 8 bits indique la durée maximale de séjour du paquet IP dans le réseau. Cette valeur est décrémentée à chaque passage par un

nœud jusqu'à ce qu'elle devienne nulle où le paquet sera rejeté. Le champ "*Protocol*" de 8 bits identifie le protocole utilisé pour les données transportées dans le paquet IP, c'est à dire le protocole du niveau suivant (UDP, TCP, etc.). Le champ "*Header Cheksum*" de 16 bits assure la détection d'erreurs sur l'en-tête IP uniquement. Les données ne sont pas protégées par ce champ. Les deux champs "*Source address*" et "*Destination address*" chacun de 32 bits indiquent respectivement les adresses source et destination du paquet IP. Enfin, le champ des options est variable en longueur et peut être nul.

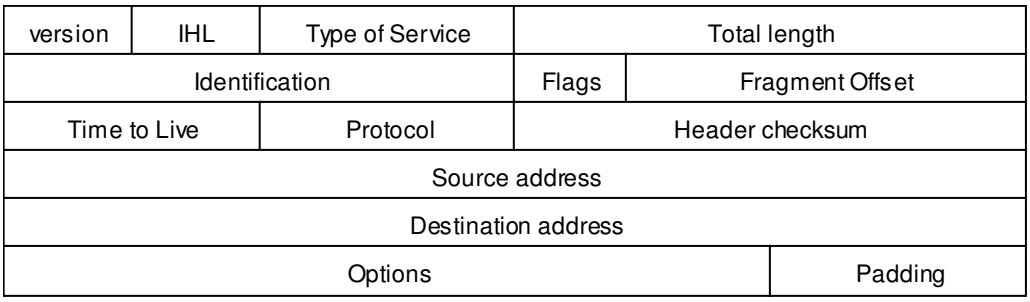


Figure 7.2 Format de l'en-tête IPv4

7.2.2.2 IP Version 6 [RFC 2460]

C'est une nouvelle version du protocole IP [75]. Les différences avec IPv4 peuvent être résumées en ce qui suit:

- Espace d'adressage plus large grâce à la longueur plus élevée des champs d'adresse.
- Format simplifié de l'en-tête.
- Possibilité d'identification des flux à l'aide d'étiquette (*Label*).

Le format d'un en-tête IPv6 est donné sur la figure 7.3. Le champ "*Traffic Class*" de 8 bits indique la classe à laquelle appartient ce paquet. C'est un champ semblable au champ ToS dans l'en-tête IPv4. Le champ "*Flow Label*" de 20 bits indique l'identificateur d'un flot de paquets. Le champ "*Payload length*" de 16 bits donne la longueur en octets de la partie des données. Le champ "*Next header*" de 8 bits indique le type de protocole de l'en-tête suivant. Le champ "*Hop limit*" de 8 bits est semblable au champ TTL de l'en-tête IPv4. Les champs "*Source address*" et "*Destination address*" de 128 bits chacun autorisent un espace d'adressage beaucoup plus large que celui de l'IPv4. La taille minimale d'un en-tête IPv6 est de 40 octets. Des extensions peuvent être ajoutées à cet en-tête mais elles sont optionnelles.

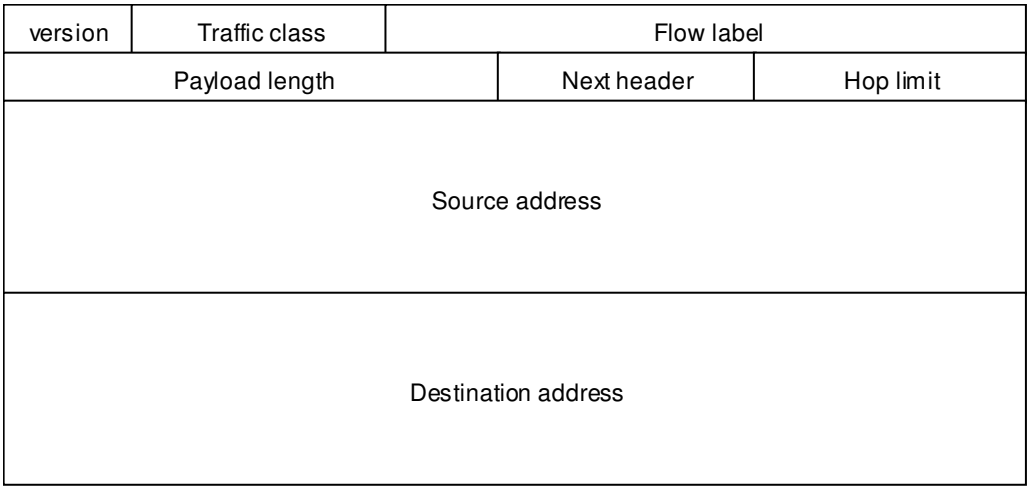


Figure 7.3 Format de l'en-tête IPv6

7.2.2.3 Compression d'en-tête (Header Compression)

La taille relativement large des en-têtes IP peut affecter l'efficacité de la bande passante qui constitue une problématique critique sur les liens à bande étroite. Une solution pour ce problème est la compression d'en-tête.

La compression d'en-tête profite du fait de redondance des informations dans les en-têtes des paquets d'un même flux. En fait, les valeurs de quelques champs comme les adresses source et destination sont les mêmes pour tous les paquets d'un même flux. L'idée de la compression consiste à envoyer initialement l'en-tête complet dans le premier paquet du flux, puis à envoyer dans les en-têtes des autres paquets uniquement les informations qui changent d'un paquet à autre comme le champ TTL (ou *Hop Limit*). Ce mécanisme réduit d'une manière considérable la longueur de l'en-tête IP. Les en-têtes complets sont envoyés de temps en temps pour mettre à jour les informations relatives au flux des paquets et pour restituer les erreurs dues aux pertes des paquets. La fréquence d'envoie des en-têtes complets varie selon la méthode utilisée. Des mécanismes de compression d'en-tête sont déjà normalisés comme par exemple le protocole "*IP Header Compression Protocol*" [76] et le protocole ROHC (*RObust Header Compression Protocol*) [80]. La compression d'en-tête peut réduire la taille d'un en-tête jusqu'à quelques octets (2-7 octets).

7.2.3 Le protocole HDLC (*High Level Data Link*) [ISO 3309]

C'est le premier protocole normalisé de niveau liaison. Ce protocole utilise les services fournis par le niveau physique et fournit un transfert de données fiable ou de type *best effort* entre deux nœuds du réseau. Le type de service fourni dépend du mode HDLC utilisé.

Chaque bloc de données est encapsulé dans une trame HDLC en ajoutant un en-tête et un en-queue (*trailer*). L'en-tête contient une adresse HDLC et un champ de contrôle. Le *trailer* contient un champ CRC (*Cyclic Redundancy Check*) pour la détection d'erreurs. Les trames sont séparées par des drapeaux. Le format d'une trame HDLC est présenté sur la figure 7.4.

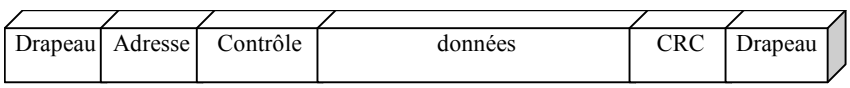


Figure 7.4 Format d'une trame HDLC

Trois types de trames existent:

- Les trames des données I : *Information frame.*
- Les trames de supervision S : *Supervision frame.*
- Les trames U : *Un-numbered frame.*

Les trames I sont utilisées pour le transport des données, les trames S pour assurer le contrôle et la correction d'erreur et les trames U pour l'établissement et la libération des liaisons virtuelles entre deux nœuds. La différence entre ces trois types de trames réside dans le format du champ de contrôle.

7.2.4 Le protocole PPP (*Point to Point Protocol*) [RFC 1661]

Le protocole point-à-point est utilisé pour le transport des paquets sur des liaisons entre deux pairs [72]. Ce protocole de niveau 2 assure le transport de différents protocoles de niveaux supérieurs. Il est inspiré du protocole HDLC et son but est de pourvoir indiquer le type des informations transportées dans le champ des données de la trame. Dans un réseau multi-protocole, il est important de savoir détecter, par un champ spécifique de niveau trame, l'application qui est transportée pour pouvoir l'envoyer vers la bonne porte de sortie. La trame du protocole PPP ressemble à celle d'HDLC. Un champ déterminant le protocole du niveau supérieur vient s'ajouter juste derrière le champ de supervision. La figure 7.5 représente le format d'une trame PPP simple et d'une trame PPP au format HDLC (*HDLC-like framing*). En effet, lorsque PPP est utilisé avec un autre protocole comme L2TP ou ATM, on n'a pas besoin d'encapsuler les trames PPP dans des trames de format HDLC. Les trames PPP sont utilisées dans leur format simple.

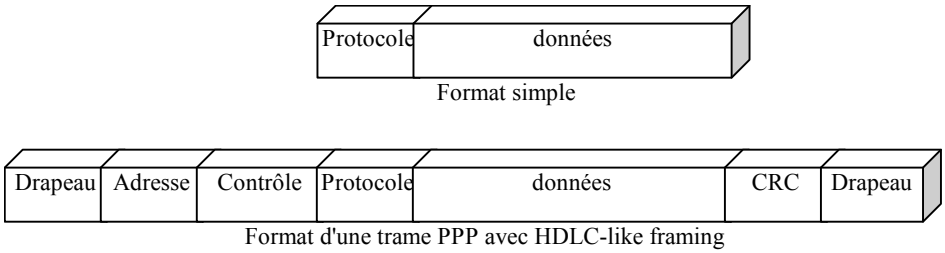


Figure 7.5 Format d'une trame PPP

7.2.4.1 PPP-mux [RFC 3153]

Le protocole PPP-mux est une variante du protocole PPP dans laquelle plusieurs paquets sont multiplexés dans une même trame pour amortir l'*overhead* PPP sur plusieurs paquets [81]. L'idée est de concaténer plusieurs trames PPP dans une seule trame en insérant un délimiteur au début de chaque sous-trame. Le format d'une trame PPP-mux est présenté sur la figure 7.6. Le délimiteur contient deux champs principaux : le champ "*PPP protocol ID*" de deux octets qui indique le type du protocole utilisé pour les paquets encapsulés et le champ "*Length*" (1 ou 2 octets). Le champ "*PPP Protocol ID*" est utilisé uniquement pour la première trame si toutes les trames multiplexées ont le même identificateur de protocole. Le champ "*Length*" contient trois sous-champs : le sous-champ PFF (*Protocol Field Flag*) d'un bit indique si le champ "*PPP Protocol ID*" est utilisé ou non, le sous-champ LXT (*Length EXtention*) d'un bit indique si le champ "*Length*" est composé de 1 ou de 2 octets et le sous-champ LEN (*sub-frame LENgth*) qui indique la longueur en octet de la sous-trame. Si le champ "*Length*" est d'un seul octet, alors le champ LEN est de 6 bits et par suite, la longueur de la sous-trame doit être inférieure à 64 octets. Pour supporter des sous-trames de longueurs plus grandes, l'extension du champ "*Length*" doit être utilisée. Dans ce cas, le champ LEN est de 14 bits et par suite, des sous-trames de 16383 octets peuvent être transportées. Un seul en-tête PPP est utilisé pour l'ensemble de la

trame PPP-mux. Dans le cas de transport sur HDLC, l'en-tête HDLC et le CRC sont utilisés pour toute la trame PPP-mux. De cette manière, le protocole PPP-mux réduit l'*overhead* par trame.

Figure 7.6 Format d'une trame PPP-mux avec HDLC-like framing

7.2.4.2 Multi-Link Protocol (MP) ou PPP-ML [RFC 1990]

Ce protocole est une variante du protocole PPP [74]. Le but de ce protocole est de regrouper plusieurs liaisons indépendantes entre deux nœuds afin d'établir une liaison virtuelle avec une bande passante plus élevée. Les paquets sont envoyés sur des supports physiques différents et sont regroupés à la réception. Les grands paquets peuvent être segmentés en des petits paquets puis envoyés simultanément sur les différents liens physiques. Les segments sont encapsulés en utilisant le format du paquet de la figure 7.7.

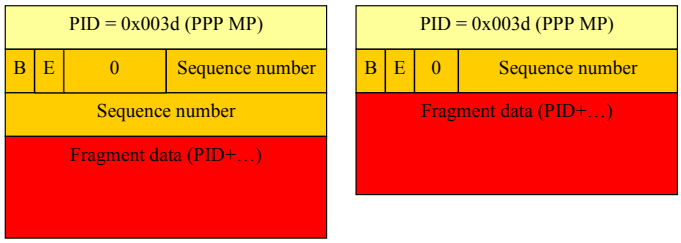


Figure 7.7 Format d'un segment PPP-ML

Le champ PID indique l'identificateur du protocole. Le bit "B" indique le premier segment d'une trame PPP tandis que le bit "E" indique le dernier segment. Le numéro de séquence "*Sequence Number*" peut être de 12 bits (*short sequence number*) ou de 24 bits (*long sequence number*). Il est incrémenté chaque fois qu'un segment est transmis. Entre le champ "E" et le champ "*sequence number*", il existe des bits non utilisés. Ils sont au nombre de 2 dans le cas de "*short sequence number*" et de 4 dans le cas de "*long sequence number*".

Il est possible d'utiliser PPP-ML avec PPP-mux. Dans ce cas, le multiplexage est réalisé avant la segmentation. L'en-tête ML doit être à l'extérieur de l'en-tête PPP-mux.

7.2.4.3 MP Multi-Class (MP-MC) ou PPP-ML-MC [RFC 2686]

C'est une extension du protocole PPP-ML (MP) [79]. En fait, les bits non utilisés dans l'en-tête MP sont utilisés par PPP-ML-MC pour définir des classes de service au niveau PPP. Dans ce cas, les segments appartenant à différentes classes peuvent subir des traitements différents dans les nœuds du réseau. Dans le cas de "*short sequence number*", 2 bits sont disponibles, on peut alors définir jusqu'à 4 classes différentes. Dans le cas de "*long sequence number*", 4 bits sont disponibles et par suite, on peut définir jusqu'à 16 classes différentes.

7.2.5 Le protocole L2TP (*Layer 2 Tunneling Protocol*) [RFC 2661]

Le protocole L2TP permet de créer un lien point-à-point en utilisant le protocole PPP entre deux points distants et en créant un tunnel entre ces deux points [78]. Ce tunnel traverse le réseau de transport qui est transparent par rapport aux deux extrémités du tunnel (figure 7.8).

Un tunnel L2TP peut transporter plusieurs sessions L2TP (sessions PPP) où chacune correspond à un couple d'adresses IP de source et de destination.

Le protocole L2TP utilise deux types de messages : les messages de contrôle et les messages des données. Les premiers sont utilisés pour l'établissement, la maintenance et la résiliation des tunnels et des sessions. Les messages des données sont utilisés pour encapsuler les trames PPP transportées sur le tunnel.

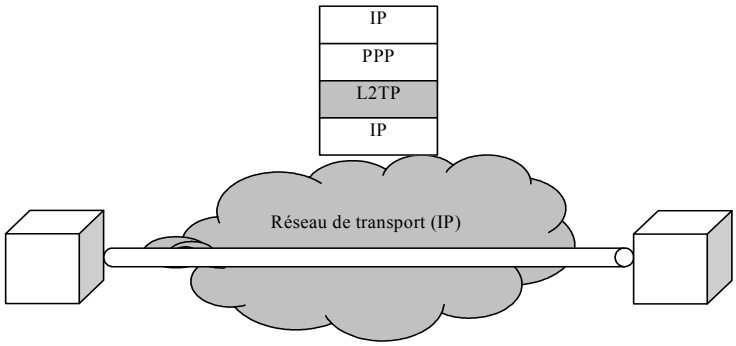


Figure 7.8 Tunnel L2TP sur un réseau IP

Chaque trame PPP est encapsulée dans un message L2TP avec un en-tête dont le format est présenté sur la figure 7.9. Le bit "T" indique le type du message (*control* ou *data*). Le bit "L" indique si le champ "*Length*" est présent ou non. Les bits "x" sont laissés pour des extensions futures. Le bit "S" indique si les champs *Ns* et *Nr* sont présents ou non. Le bit "O" indique si le champ "*Offset*" est présent ou non. Quand le bit de priorité "P" est égal à "1", le message reçoit un traitement prioritaire dans les nœuds du réseau. Le champ "*Ver*" indique la version du protocole L2TP. Le champ "*Length*" de 2 octets est optionnel et il indique la longueur totale du message en octets. "*Tunnel ID*" est un champ obligatoire de 2 octets qui contient l'identificateur du tunnel. Le champ "*Session ID*" est aussi un champ obligatoire de 2 octets qui indique l'identificateur de la session L2TP dans le tunnel. Le champ *Ns* de 2 octets est optionnel et il indique le numéro de séquence du message. Le champ *Nr* de deux octets est aussi optionnel et il est utilisé pour indiquer le numéro de séquence du message attendu afin de détecter les erreurs ou les pertes de messages. Ce champ est utilisé dans les messages de contrôle uniquement pour leur assurer un transfert fiable. Le champ "*Offset size*" de 2 octets est optionnel et il indique le début du champ des données.

T L x x S x O P x x x x	Ver	Length (opt) (2 bytes)
Tunnel ID (2 bytes)		Session ID (2 bytes)
Ns (opt) (2 bytes)		Nr (opt) (2 bytes)
Offset size (opt) (2 bytes)		Offset pad ... (opt)

Figure 7.9 Format d'un en-tête L2TP

7.2.6 La couche d'adaptation de l'ATM AAL-5 [ITU-T I.363.5]

La couche d'adaptation de l'ATM AAL5 (*ATM Adaptation Layer- type 5*) est utilisée pour transporter les paquets des couches supérieures sur un support ATM [65]. Cette couche est divisée en

deux sous-couches : SAR (*Segmentation And Reassembly*) et CPCS (*Common Part Convergence Sublayer*). Les paquets des couches supérieures sont encapsulés dans les unités des données AAL5 qui sont à leur tour segmentées en des petites cellules ATM de 53 octets. Le format de l'unité des données AAL5 est décrit sur la figure 7.10. Le champ CRC (*Cyclic Redundancy Check*) permet la détection d'erreurs de l'unité des données. Le champ "*Length*" de 2 octets indique la longueur en octets du champ des données. Le champ CPI (*Common Part Indicator*) n'est pas utilisé. Le champ UU (*User-to-User indication*) est utilisé par les couches supérieures comme la couche SSCS. Le champ PAD (*PADding*) de taille variable entre 0 et 47 octets est utilisé pour ajuster la taille totale de l'unité des données à un multiple de 48. Le champ des données (*payload*) a une longueur variable entre 1 et 65535 octets, mais la taille totale de l'unité doit être toujours un multiple de 48 pour que la sous-couche SAR puisse la segmenter en des blocs de 48 octets. Ces blocs sont insérés dans le champ d'information des cellules ATM.

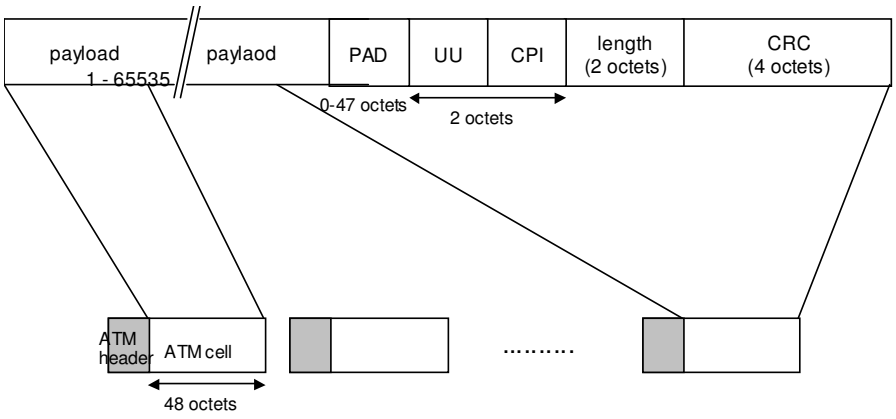


Figure 7.10 Format d'une unité de données AAL5

7.3 Les solutions de transport dans un UTRAN-IP

Dans les spécifications du 3GPP [108], l'utilisation de la pile UDP/IP au niveau 3 de l'architecture OSI est obligatoire dans tous les Node B de l'UTRAN. Il est aussi obligatoire d'utiliser le protocole PPP/HDLC au niveau 2 dans tous les Node B supportant le protocole IP. Tous les Node B de l'UTRAN dans la Release 5 du 3GPP doivent supporter la version 6 du protocole IP. La version 4 est une alternative qui peut coexister avec la version 6. D'autres protocoles de transport et surtout de niveau 2 peuvent être utilisés pour compléter la panoplie protocolaire sur l'interface Iub.

Afin d'optimiser l'utilisation de la bande passante sur le *Last Mile Link* entre le Node B et *l'Edge Router* du *Backbone* IP, la technique de multiplexage des flux est envisageable surtout pour les flux dont les paquets ont des petites tailles comme la voix compressée par exemple. Le multiplexage est effectué au niveau 2 en utilisant le protocole PPP-mux (*Point-to-Point Protocol - multiplexing*).

Les flux transportés sur l'interface Iub ont des contraintes particulières à cause de l'aspect temps-réel imposé par les mécanismes de synchronisation de l'UTRAN. D'ailleurs, certains types d'applications sont plus tolérants que d'autres malgré l'aspect temps-réel de tous les flux transportés. Cette différence permet de distinguer entre plusieurs classes de service et par suite elle permet un traitement différent pour chaque classe. Dans tous les cas, le protocole utilisé dans le réseau de transport doit être capable de gérer des flux avec des contraintes temporelles strictes.

En tenant compte de ce qui précède, une architecture de qualité de service est alors nécessaire sur l'interface Iub dans le cas de transport en IP pour pouvoir garantir les besoins des flux transportés. Trois architectures sont proposées pour la qualité de service sur l'interface Iub : A, B et C. Or, à cause

de l'aspect confidentiel de la troisième architecture (C) qui est une propriété de France Télécom R&D, elle ne sera pas présentée dans cette thèse et seulement les architectures A et B seront évoquées.

7.3.1 Architecture A: QoS de bout en bout et multiplexage de bout en bout

La figure 7.11 représente cette première architecture. Le protocole PPP est utilisé pour créer des connexions point-à-point entre le Node B et le RNC à travers le réseau de transport. Ceci n'est possible que si on utilise un tunnel entre le Node B et le RNC. Ce tunnel est créé par le protocole L2TP qui assure une connexion directe entre le Node B et le RNC. Le réseau *backbone* est alors transparent par rapport à cette connexion. L'usager de la couche IP voit une connexion point-à-point entre le Node B et le RNC. Tous les flux sont alors canalisés dans un tunnel de bout en bout, c'est à dire du Node B jusqu'au RNC. Dans cette architecture, la qualité de service est définie de bout en bout.

Pour optimiser les ressources des liens, des fonctions de multiplexage sont assurées par le protocole PPP-mux et elles sont appliquées sur les paquets de petites tailles. Par contre, les paquets de grandes tailles sont segmentés à l'aide de l'extension ML (*Multi-Link*) du protocole PPP (PPP-ML ou MP). L'extension *Multi-Class* du protocole PPP-ML (PPP-ML-MC) est utilisée pour assurer la différenciation des services. Cette différenciation des services est faite à ce niveau et plusieurs classes de service peuvent être utilisées pour fournir à chaque type d'application la qualité de service nécessaire. Des mécanismes d'ordonnancement sont nécessaires au niveau PPP-ML-MC pour faire la différenciation entre les différentes classes de service.

Tous les flux sont ensuite regroupés dans un même tunnel et transportés dans le réseau de transport (*Backbone*) sur une connexion UDP/IP. Deux niveaux IP sont alors utilisés : le niveau supérieur avant le multiplexage et le niveau inférieur après le multiplexage. Puisque la différenciation des services est faite au niveau supérieur, une seule qualité de service est utilisée au niveau inférieur et toutes les classes supérieures sont transportées sur cette classe. Pour garantir la qualité de service nécessaire pour les flux les plus exigeants en terme de contraintes temporelles, il est alors nécessaire d'utiliser la classe de service de plus haute priorité dans la couche IP du niveau inférieur. Les protocoles PPP/HDLC sont utilisés au niveau 2.

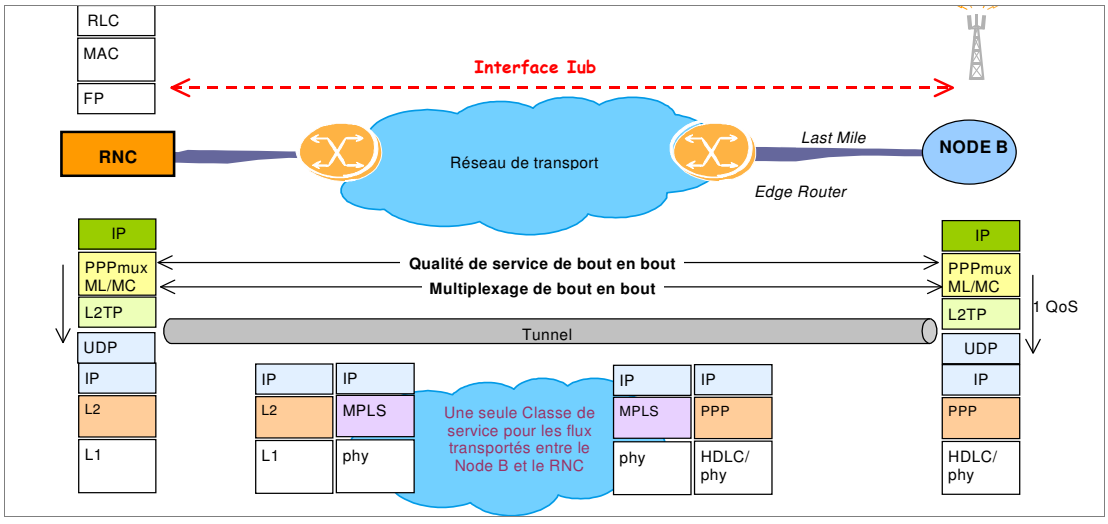


Figure 7.11 Architecture A : QoS de bout en bout et multiplexage de bout en bout

Dans le réseau de transport, une seule classe de service est utilisée pour le transport des flux de l'interface Iub. Par conséquent, le trafic n'a pas besoin d'un traitement particulier dans les nœuds intermédiaires. L'avantage de cette solution est la simplicité du traitement dans les nœuds

intermédiaires. L'inconvénient est que la bande passante réservée peut être plus grande que la bande passante réellement utilisée à cause de la réservation des ressources sur la base de la classe de service de plus haute priorité.

Sur le *Last Mile Link*, les différentes applications sont transportées sur la pile protocolaire suivante:

- UDP/IP/PPPmux-ML-MC/L2TP/UDP/IP/PPP/HDLC

Un paquet est alors encapsulé dans tous les en-têtes de ces protocoles.

7.3.2 Architecture B: QoS point-à-point et multiplexage sur le *Last Mile Link*

Cette architecture diffère de la première par deux aspects essentiels. Tous d'abord, le multiplexage (s'il est utilisé) est effectué sur le *Last Mile Link* uniquement et non pas de bout en bout. Les flux transportés dans le réseau de transport ne sont pas multiplexés. La deuxième différence est que la qualité de service n'est pas définie de bout en bout mais avec une approche point-à-point (*Hop by Hop*). Dans le réseau de transport, la différenciation des services est réalisée en utilisant le champ ToS (*Type of Service*) de l'en-tête IP. Le *Edge Router* doit être capable de réaliser les fonctions de multiplexage au niveau de la couche 2.

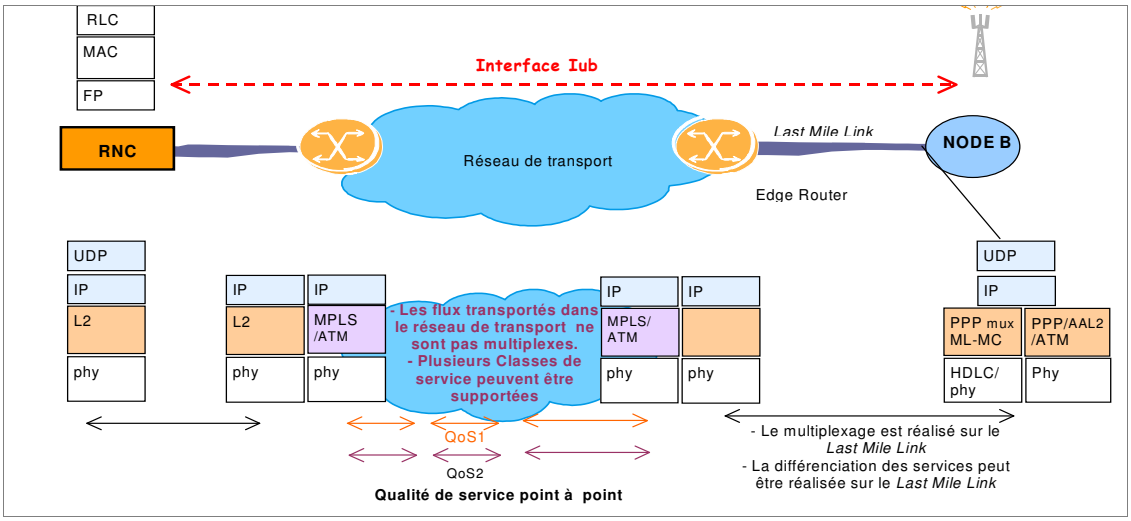


Figure 7.12 Architecture B : QoS point-à-point et multiplexage sur le *Last Mile Link*

Plusieurs solutions sont envisageables pour cette architecture. La solution la plus simple est de transporter les flux UDP/IP directement sur PPP/HDLC sans aucun multiplexage. La différenciation des services est effectuée au niveau 3 (couche IP). L'avantage de cette solution est la simplicité tandis que l'inconvénient est l'efficacité réduite de la bande passante. En effet, la charge supplémentaire due aux en-têtes ne peut pas être amortie dans le cas des petits paquets parce qu'aucun mécanisme de multiplexage n'est utilisé. Une deuxième solution proche de la première consiste à utiliser les extensions ML-MC du protocole PPP. Dans cette solution, il n'y a pas de multiplexage et la différenciation des services peut être réalisée au niveau 2 (PPP-ML-MC). Les piles protocolaires correspondantes à ces deux solutions sont les suivantes:

- UDP/IP/PPP/HDLC
- UDP/IP/PPP-ML-MC/HDLC

Une troisième solution est envisageable. Elle consiste à utiliser le protocole PPP-mux avec ses extensions ML-MC (PPP-mux-ML-MC). Dans ce cas, on profite de l'effet de multiplexage pour

optimiser l'utilisation de la bande passante sur le *Last Mile Link*. Les extensions ML-MC donnent la possibilité de différencier les services au niveau 2. La pile protocolaire de cette solution est :

- UDP/IP/PPPmux-ML-MC/HDLC

La quatrième solution utilise PPP-mux pour le multiplexage des paquets mais utilise la couche d'adaptation AAL5 pour le transport des paquets sur le protocole ATM. Cette solution est envisageable dans le cas où le réseau de transport serait basé sur la technologie ATM. Dans cette solution, le réseau IP de l'UTRAN peut utiliser un *backbone* de type ATM pour transporter ses flux sur l'interface Iub sans problème d'interopérabilité entre les différents réseaux déployant des technologies différentes. La pile protocolaire de cette solution est:

- UDP/IP/PPPmux/AAL5/ATM

La dernière solution utilise le protocole AAL2/ATM au-dessous du protocole de niveau 2 PPP. Dans ce cas, l'interface Iub de l'UTRAN de la Release 5 (IP) peut traverser un *backbone* de la Release 99 basé sur la technologie AAL2/ATM sans aucun problème d'interopérabilité entre les deux réseaux. La pile protocolaire correspondante à cette solution est:

- UDP/IP/PPP/AAL2/ATM

7.3.3 Architecture C: QoS point-à-point et multiplexage de bout en bout

Cette architecture est une propriété de France Télécom R&D et ne sera pas présentée dans cette thèse.

7.3.4 Le protocole MPLS dans le réseau de transport

Nous avons vu dans les paragraphes précédents que le protocole MPLS peut être utilisé dans le réseau de transport comme protocole de transport au-dessous de l'IP pour garantir la qualité de service requise par les flux transportés sur l'interface Iub. Dans ce paragraphe, nous allons présenter le protocole MPLS ainsi que son introduction dans le réseau de transport.

7.3.4.1 Description générale

MPLS (*Multi-Protocol Label Switching*) est un protocole de niveau 2.5 qui complète et améliore le protocole IP en offrant de nouvelles méthodes de transfert de paquet IP tout en utilisant les protocoles de routage IP déjà existants (OSPF, BGP). MPLS peut fonctionner sur plusieurs technologies de niveau 2 (PPP/Sonet, Ethernet, ATM, FR, WDM, etc). Il transmet les paquets IP en utilisant un Label de 20 bits. A l'entrée du domaine MPLS, le routeur d'entrée (*Ingress Router*) appelé aussi *Label Edge Router* décide de la correspondance entre le groupe de paquets entrant et le conduit LSP (*Label-Switched Path*) dans lequel les paquets seront acheminés dans le domaine MPLS et ensuite ajoute le *Label* correspondant pour chaque paquet entrant. Le groupe de paquets transférés sur le même LSP est appelé FEC (*Forwarding Equivalence Class*). Les paquets sont alors transférés à travers le domaine MPLS par les LSR (*Label Swtiched Routers*) en se basant sur le Label. Au nœud de sortie (*Egress Node*) du domaine MPLS, le LSR enlève le Label MPLS de chaque paquet IP et ensuite les paquets IP sont transférés par les mécanismes traditionnels de l'IP. Chaque paire de LSR sur un LSP doit négocier le Label qui sera utilisé sur ce segment du LSP. Cette négociation est effectuée en utilisant un ensemble de procédures appelé *Label Distribution Protocol* (LDP). Le LDP associe un FEC pour chaque LSP. Le FEC associé avec un LSP détermine quels paquets seront associés à ce LSP.

7.3.4.2 Routage avec MPLS

MPLS, comme une technique complémentaire de l'IP pour le transfert des paquets, offre les avantages suivants:

- **Coexistence avec IP Hop-By-Hop Routing.** Un LSR est capable de transférer des paquets IP ainsi que des trames MPLS.
- **Capacité d'ingénierie de trafic.** MPLS utilise le *Label* du paquet IP pour déterminer le trajet que le paquet doit prendre dans le réseau, sans regarder l'adresse IP. Les trajets à travers le réseau peuvent être dimensionnés de façon à satisfaire les besoins des opérateurs (QoS par exemple). Par exemple, dans le nœud de bordure du domaine MPLS, le trafic peut être séparé selon la classe de qualité de service et les paquets peuvent être acheminés sur le trajet MPLS qui correspond à leur besoin en qualité de service.
- **Flexibilité due à la sémantique du Label.** La signification du *Label* peut être ajustée aux besoins demandés dans le réseau. Par exemple, les *Labels* peuvent être utilisés pour spécifier un certain traitement de QoS, multiplexage, *multicast*, compression d'en-tête, etc.
- **Flexibilité due à l'empilement des Labels.** MPLS supporte la possibilité d'empiler plusieurs *Label* au début d'un paquet IP. Ceci permet différents adressages dans différents sous-réseaux ainsi qu'un support efficace des tunnels dans tunnels (*Tunnel-in-Tunnel*) pour la mobilité IP par exemple.
- **Routage transparent.** Les paquets compressés passent d'une façon transparente à travers les LSR intermédiaires. Cela est en contradiction avec les schémas basés par exemple sur PPP où la compression/décompression des en-têtes doit se faire dans chaque nœud traversé, ou les paquets compressés doivent être transportés dans d'autres paquets non compressés d'un tunnel. MPLS rend les nœuds du réseau plus simples.
- **Routage rapide.** Les mécanismes de protection de la commutation MPLS permettent une restitution rapide après la panne d'un nœud. La protection locale et la protection de bout en bout doivent être utilisées ensemble pour achever une restitution rapide d'un tunnel.
- **Adaptation avec n'importe quel niveau 2.** MPLS peut être déployé au-dessus de plusieurs technologies de niveau 2 (Layer 2 : L2). Quand MPLS est utilisé sur ATM ou sur le relais de trames (*Frame Relay*), la correspondance pourra être faite entre les LSP et les connexions de niveau 2 comme les VCC ou les PVC.

7.3.4.3 Support de la qualité de service

MPLS supporte plusieurs mécanismes de différenciation de QoS pour les flux IP. Les flux avec différentes caractéristiques de QoS peuvent être séparés dans plusieurs LSP. Les LSP peuvent être dimensionnés afin de fournir les besoins de QoS pour chaque classe de trafic transportée dans le réseau. Le trafic peut être séparé à la sortie du domaine MPLS selon la classe de QoS.

Prenons l'exemple des liaisons à bas débit, les flux temps-réel passent par les LSP garantissant une bonne qualité de service tandis que les paquets des données passent par des LSP séparés. De cette façon, on évite le risque des longs paquets qui peuvent bloquer le chemin des petits paquets sensibles au délai.

DiffServ (*Differentiated Services*) fournit un mécanisme qui définit le traitement que doit subir un paquet quand il passe dans un réseau IP. Bien qu'il n'y pas de garantie des performances avec *DiffServ*, il peut être utilisé pour améliorer les performances de bout en bout sur des grands réseaux. MPLS peut supporter *DiffServ* en utilisant le marquage *DiffServ* dans chaque paquet pour déterminer:

- Le chemin que doit prendre le paquet. Les chemins peuvent être dimensionnés de telle façon à fournir des garanties de performance meilleures que dans le cas d'un réseau routé avec *DiffServ* pur.
- Le traitement que les paquets doivent subir sur un chemin donné. Dans ce modèle, qui ressemble au modèle *DiffServ* basique, les paquets de QoS différentes peuvent être supportés par le même chemin MPLS (le *LSP*). Dans ce LSP, le marquage *DiffServ* peut être utilisé

pour donner la priorité à certains paquets en respectant les autres paquets supportés par le même LSP.

Enfin, les paquets prennent le même chemin déterminé par le LSP et par suite ils sont récupérés à la réception dans leur ordre d'émission.

L'optimisation de la bande passante est réalisée sur le lien à bande étroite "*Last Mile Link*". La figure 7.13 représente les piles protocolaires dans les nœuds d'un réseau basé sur une solution MPLS.

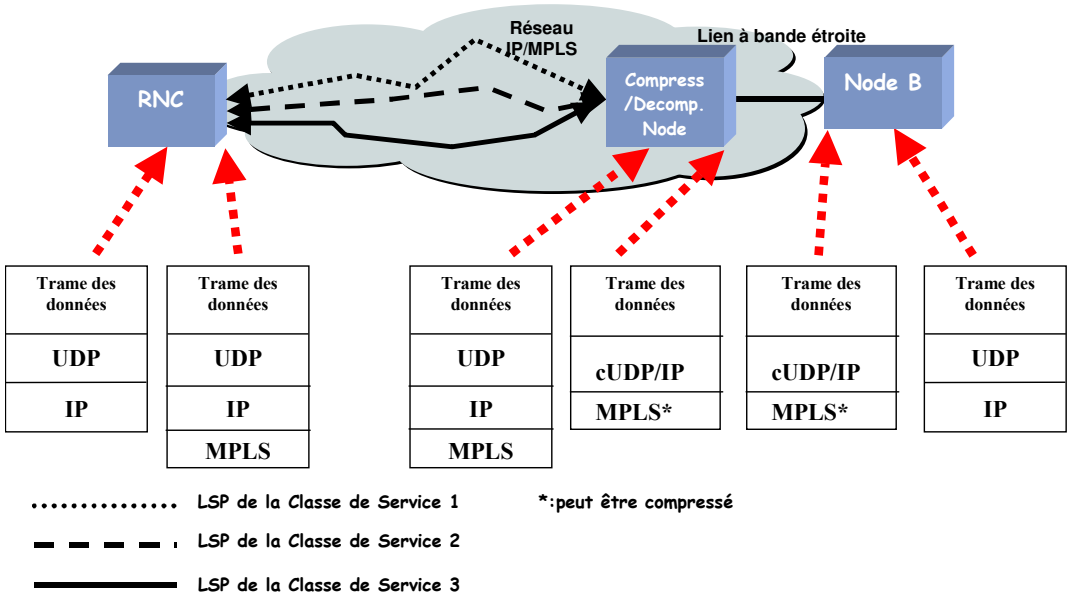


Figure 7.13 Pile protocolaire dans les nœuds d'un UTRAN basé sur une solution MPLS

Sur le lien descendant, les paquets UDP/IP sont associés aux LSP dans le RNC et sont envoyés sans compression à travers le réseau vers le nœud de compression/décompression CDN (*Compression/Decompression Node*). Ces paquets sont alors compressés et envoyés sur le lien à bande étroite. Au niveau du Node B, les paquets UDP/IP sont récupérés et décompressés. Sur le lien montant, les paquets sont compressés dans le Node B et envoyés sur le lien à bande étroite vers le CDN qui les décompresse puis les envoie dans les LSP vers le RNC.

Dans une solution de transport basée sur MPLS avec *DiffServ* :

- Les LSP sont établis entre un RNC et un Node B dans les deux directions. Chaque LSP peut supporter une ou plusieurs classes de service. En donnant le même traitement pour les paquets appartenant à la même classe de service dans chaque nœud du LSP, la qualité de service totale dans ce LSP sera établie.
- L'opérateur décide du nombre des classes de service qui seront supportées dans l'UTRAN et combien de classes seront transportées dans un LSP.
- Un paquet IP est transporté sur un LSP avec la classe de service appropriée en se basant sur deux choses: le code DS dans l'en-tête IP et le FEC au quel le paquet appartient (l'adresse IP de destination).
- Les paquets IP sont transportés sur les LSP convenables dans les nœuds frontières de l'UTRAN, les RNC et les Node B.

Enfin, dans le cas de transport des longs paquets sur une liaison à bas débit, ces paquets peuvent générer des grands délais pour les paquets temps-réel et par suite une dégradation de la qualité de service des flux temps-réel. Une solution pour résoudre ce problème consiste à fragmenter les grands paquets en des petits paquets et à implémenter des mécanismes d'ordonnancement pour différencier entre les différents paquets selon leurs besoins de QoS. Quand MPLS est utilisé dans l'UTRAN, la fragmentation peut être réalisée au niveau 2: AAL5/ATM, *Multi-link* PPP, *Frame Relay*, etc.

7.3.4.4 Fragmentation

La fragmentation est nécessaire pour ajuster la taille des paquets à la taille maximale MTU (*Maximum Transmission Unit*) et pour éviter le retard des paquets temps-réel à cause des grands paquets surtout sur des liaisons à bas débit. Par exemple, pour un débit de 384 kbit/s et un TTI de 80 ms, la taille d'un paquet est de 3840 octets. La couche RLC va segmenter ce paquet mais tous les TB seront multiplexés dans un même paquet (TB set). Alors au niveau du réseau de transport, il est parfois nécessaire de fragmenter les gros paquets pour pouvoir garantir la qualité de service demandée.

7.3.4.4.1 Fragmentation IP

Le protocole IP est capable de fragmenter un paquet en plusieurs segments en se basant sur la taille de la MTU correspondante au trajet qui sera traversé par le paquet. La MTU d'un trajet peut être découverte à travers le protocole "*Path MTU discovery*" en envoyant un message ICMP sur le trajet et en recevant la plus petite MTU découverte sur le chemin traversé. Si le paquet est plus grand que la MTU, il sera fragmenté. La MTU est déterminé par les routeurs en se basant sur les caractéristiques du lien. Pour le protocole PPP, la taille du MTU est flexible. Pour les liens *Ethernet*, la taille maximale par défaut est de 1500 octets. Pour le *Gigabit Ethernet*, la taille du MTU est de 9000 octets (*Jumbo Frame*).

La fragmentation IPv4 présente quelques inconvénients:

1. L'efficacité de la bande passante grâce aux grands paquets ne sera pas réalisée sur les liaisons à grande capacité car les paquets fragmentés ne seront rassemblés qu'au point de terminaison.
2. Pour IPv4, la compression d'en-tête ne peut pas être utilisée ce qui n'est pas le cas pour IPv6.
3. Pour IPv4, l'*overhead* est grand quand on utilise la fragmentation IP. En plus, la fragmentation peut être faite à n'importe quel point sur le chemin. Cela peut introduire des traitements très lourds dans les routeurs. La fragmentation IPv6 ne peut se faire que sur les terminaisons.

La fragmentation, qu'elle soit au niveau IP ou au niveau application, peut être utilisée pour ajuster la taille des paquets au MTU du chemin mais n'est pas convenable pour résoudre le problème sur un lien à bande étroite. En effet, IPv6 autorise une taille minimale de MTU de 1280 octets ce qui n'est pas assez petits pour les liens à bandes étroites.

Puisque les inconvénients de la fragmentation IP ne sont pas considérables dans le cas où elle serait faite sur les extrémités, la fragmentation IP sera supportée dans les nœuds de l'UTRAN pour ajuster les paquets à la taille du MTU du chemin. Cette fragmentation doit être faite uniquement sur les extrémités pour IPv4 et IPv6. Le réseau doit être dimensionné en prenant en compte que la taille du MTU ne soit pas très petite pour que les en-têtes IP ne consomment pas une grande partie de la bande passante. La fragmentation IP ne doit pas être réalisée pour les flux temps-réel sur des liens à bas débit. Les mécanismes de niveau 2 seront utilisés pour ce but comme indiqué pour IPv6 [RFC 2460]. IPv6 exige une taille de MTU de 1280 octets ou plus sur tous les liens. Sur un lien qui ne peut pas supporter cette taille, une fragmentation et un ré-assemblage doivent être réalisés dans une couche au-dessous de l'IPv6.

7.3.4.4.2 Fragmentation au niveau 2

En général, il est plus convenable de considérer les problèmes des liens à bas débits seulement sur la partie à bande étroite et non pas sur tout le chemin. Une solution est de faire la segmentation aux niveaux plus bas. Par exemple, pour IPv6, sur les liens qui ne peuvent pas transporter des paquets de 1280 octets, une fragmentation et un ré-assemblage doivent être réalisés dans une couche au-dessous de IPv6. La fragmentation au niveau 2 fournit une flexibilité car elle n'est pas de bout en bout. Elle peut être réalisée par plusieurs sauts ce qui est plus flexible que la fragmentation au niveau application ou au niveau IP.

7.3.4.4.3 Fragmentation au niveau application

La fragmentation au niveau application peut aider à éviter la fragmentation IP mais ne résout pas le problème de l'efficacité pour les liens à faible vitesse. La méthode "*Path MTU discovery*" peut être utilisée pour déterminer la taille des paquets nécessaire pour éviter la fragmentation IP. En revanche, elle ne peut pas fournir les informations nécessaires pour déterminer la taille des paquets pour une utilisation efficace du lien.

7.4 La qualité de service dans l'UTRAN

Sur l'interface Iub de l'UTRAN, nous pouvons faire la distinction entre les différents types de flux selon leurs exigences en terme de délai de transfert. Même si sur l'interface Iub, tous les flux deviennent sensibles au délai de transfert à cause des mécanismes de synchronisation des couches radio, nous pouvons toujours faire une différenciation entre des flux très exigeants et d'autre plus tolérants comme dans le cas de l'AAL2. En effet, le réseau RNL est indépendant du réseau TNL et par suite, les exigences du réseau radio RLN sont les mêmes quelle que soit la technologie de transport utilisée dans le TNL. En partant de cette hypothèse, nous pouvons diviser les flux de l'UTRAN en plusieurs classes selon leurs exigences de qualité de service.

Au niveau IP, la solution *DiffServ* (*Differentiated Service*) [83, 86] est utilisée pour distinguer entre les différentes classes. Cette solution consiste à utiliser le champ "ToS" dans IPv4 ou le champ "*Traffic Class*" dans IPv6 pour assurer la différenciation entre les classes de service. Dans *DiffServ*, trois grandes classes sont définies :

- La classe EF (*Expedited Forwarding*) pour les applications temps-réel très exigeants en terme de délai de transfert. Les paquets de cette classe ont la priorité la plus élevée.
- La classe AF (*Assured Forwarding*) pour les applications moins exigeantes que celles de la classe EF. Elle a une priorité inférieure par rapport à EF. Dans cette classe, plusieurs sous-classes sont définies pour faire la différenciation entre plusieurs niveaux de priorité dans la même classe AF.
- La classe BE (*Best Effort*) est la classe par défaut dans les réseaux Internet actuels. Elle est la moins prioritaire et elle ne garantit aucune qualité de service pour les flux transportés. Cette classe ne peut pas être utilisée dans le cas de l'UTRAN à cause des contraintes strictes imposées par le transport des canaux radio.

Dans le cas de l'UTRAN, la classe EF est utilisée pour transporter les flux dont les contraintes temporelles sont très strictes comme la classe *Conversational* de l'UMTS. La classe *Streaming* peut être aussi transportée sur la classe EF. Puisque les applications de cette classe tolèrent la variation du délai (la gigue) ainsi qu'un délai plus élevé que les applications conversationnelles, cette classe peut être traitée avec une priorité inférieure à celle des applications conversationnelles. Nous pouvons alors transporter les applications *Streaming* sur la classe AF. Les applications de la classe *Interactive* sont tolérantes au délai et peuvent être transportées sur la classe AF. Pour les applications de la classe *Background*, les contraintes temporelles ne sont pas strictes, mais puisque nous sommes dans un

contexte particulier qui est celui de l'UTRAN, les flux de cette classe deviennent sensibles au délai. Or, les paquets de cette classe sont moins prioritaires que les paquets des autres applications de la classe AF. Deux niveaux de priorité sont alors définis dans la classe AF : la sous-classe AF1 et la sous-classe AF2. La première est plus prioritaire que la deuxième. En résumé, nous avons trois niveaux de priorité pour les flux transportés sur l'interface Iub de l'UTRAN : EF, AF1 et AF2. La classe EF est très sensible au délai mais plus tolérante au taux de perte. Les classes AF1 et AF2 sont plus tolérantes au délai mais sensibles au taux de perte.

La figure 7.14 présente le schéma d'association entre les classes de l'UMTS et les classes *DiffServ* du réseau de transport.

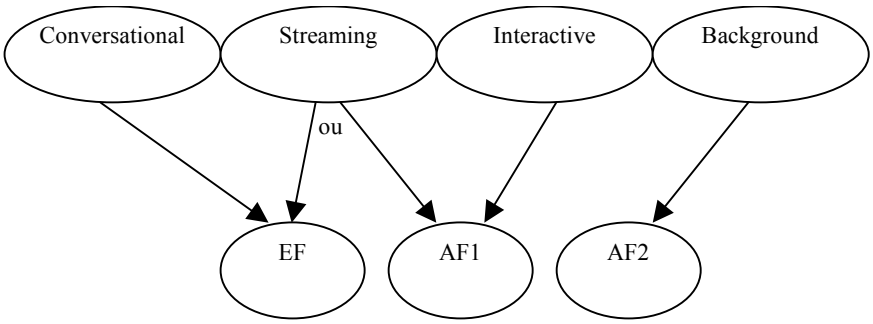


Figure 7.14 Association entre les classes de l'UMTS et les classes DiffServ

7.4.1 Les niveaux de différenciation des services

La différenciation des services dans le réseau de transport peut se faire à différents niveaux selon l'architecture utilisée. Pour l'architecture A, la différenciation des services est réalisée au niveau de la couche IP supérieure, c'est à dire dans le Node B et dans le RNC. Dans ce cas, la différenciation des services est réalisée de bout en bout. Au sein du réseau de transport, tous les flux sont transportés sur une même classe de service qui est la classe la plus prioritaire EF. Un deuxième niveau de différenciation des services de bout en bout peut être réalisé grâce au protocole PPP-ML-MC.

Ce deuxième niveau de différenciation des services peut être utile dans le cas où les fonctions de segmentation du protocole PPP-ML seraient utilisées. En effet, les gros paquets de basse priorité peuvent retarder les paquets temps-réel de petites tailles lors de la transmission sur le lien point-à-point. En segmentant les gros paquets en des petits segments et en donnant une priorité supérieure aux paquets temps-réel, on peut intercaler ces derniers entre les segments de priorité inférieure en utilisant des mécanismes d'ordonnancement au niveau PPP-ML-MC. Ces mécanismes d'ordonnancement sont basés sur plusieurs niveaux de priorité au niveau de la couche PPP-ML-MC. Ces mécanismes peuvent améliorer la qualité de service fournie aux paquets temps-réel.

Trois niveaux de priorité (1,2 et 3) sont définis au niveau de la couche PPP-ML-MC sous forme de classes de service. Cette priorité est indiquée dans le champ "*Class*" de l'en-tête PPP-ML-MC. Une association entre les classes de l'IP et les classes de PPP-ML-MC est définie. La classe EF est associée à la classe de priorité 1 du niveau PPP-ML-MC, la classe AF1 est associée à la classe de priorité 2 et la classe AF2 est associée à la classe de priorité 3.

Une fois que l'ordonnancement est fait au niveau PPP-ML-MC, le flux sortant est canalisé dans un tunnel L2TP sur UDP/IP/PPP/HDLC. Au niveau de la couche IP inférieure, une seule classe de service est utilisée (EF). Tous les paquets de ce flux subissent le même traitement prioritaire dans les différents nœuds du réseau de transport.

Dans le cas de l'architecture B, il existe un seul niveau IP. La différenciation des services est alors réalisée au niveau IP en utilisant les classes *DiffServ*. D'ailleurs, lorsque le protocole PPP-ML-MC est

utilisé au niveau de la couche 2, il est possible de réaliser un deuxième niveau de différenciation des services comme dans le cas de l'architecture A. Dans le cas d'une solution PPPmux sur AAL5/ATM au niveau 2, un VC est établi pour chaque classe *DiffServ*. Trois VC sont alors utilisés et des mécanismes d'ordonnancement sont utilisés au niveau ATM pour différencier entre les cellules des différents VC et pour donner la priorité aux cellules plus exigeantes en terme de délai de transfert. Dans le cas d'une solution PPP sur AAL2/ATM au niveau 2, une connexion AAL2 est établie pour chaque classe du niveau IP. Par conséquent, chaque classe peut être identifiée au niveau AAL2 par un numéro de CID. Des mécanismes d'ordonnancement sont utilisés au niveau AAL2 pour donner la priorité aux paquets de la classe de plus haute priorité.

Quelle que soit la solution utilisée pour l'architecture B, les différents services peuvent être différenciés dans tous les nœuds du réseau de transport dans une approche de QoS point-à-point. Dans ce cas, les paquets conservent leurs niveaux de priorité tout au long du chemin traversé et ils subissent des traitements différents dans les nœuds du réseau selon leurs classes de service.

Nous n'allons pas donner des détails sur le cas de l'architecture C à cause de son aspect confidentiel.

7.4.2 La stratégie d'ordonnancement

Quelle que soit l'architecture utilisée (A, B ou C) et quelle que soit la solution de transport adoptée, trois classes de services sont toujours définies pour transporter les différents flux de l'UTRAN. Comme nous l'avons déjà mentionné plus haut, des mécanismes d'ordonnancement sont nécessaires pour différencier entre les flux des différentes classes. Quel que soit le niveau auquel nous implémentons ces mécanismes d'ordonnancement (IP, PPP-ML-MC, AAL2 ou ATM), nous adoptons la même stratégie pour différencier entre les classes de service. Le schéma de la figure 7.15 représente la stratégie d'ordonnancement utilisée.

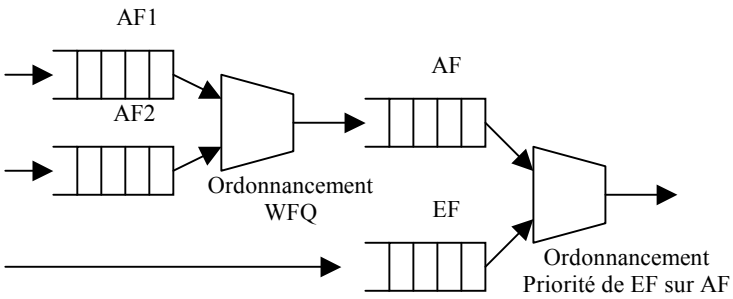


Figure 7.15 Stratégie d'ordonnancement

Tous les flux d'une même classe sont agrégés dans une file d'attente selon le mécanisme classique FIFO. Un algorithme d'ordonnancement WFQ (*Weighted Fair Queueing*) est utilisé pour différencier entre les classes AF1 et AF2. Ce mécanisme, expliqué ci-dessous, partage la bande passante disponible entre les deux classes et selon leurs besoins. Les paquets sortant de l'ordonnanceur WFQ sont mémorisés dans une file d'attente de type FIFO. L'ordre des paquets dans cette file d'attente FIFO est déterminé par le mécanisme WFQ. Ensuite, un autre ordonnanceur est implémenté pour différencier entre les paquets AF et les paquets EF. Cet ordonnanceur est de type PQ (*Priority Queueing*) dans lequel les paquets EF sont prioritaires.

Le mécanisme d'ordonnancement WFQ:

Nous avons mentionné dans le chapitre 6 que le mécanisme d'ordonnancement WRR ne peut assurer un partage juste de la bande passante que si tous les paquets ont la même taille. Le mécanisme

WFQ n'a pas cet inconvénient. Il peut partager la bande passante disponible entre les différents flux de manière juste quelle que soit la taille de leurs paquets.

L'algorithme WFQ peut être modéliser par un système GPS (*Generalized Processor Sharing*) dont le comportement est similaire à un algorithme WRR appliqué sur la base des bits et non pas des paquets. Chaque flux a un poids correspondant à son pourcentage demandé de bande passante. Dans chaque cycle du mécanisme, un certain nombre de bits de chaque flux est transmis. Ce nombre dépend du poids du flux ou de la file d'attente. Or, le système GPS est un modèle théorique qui ne peut pas être implémenté en pratique. L'algorithme WFQ est une approximation du système GPS qui fonctionne sur la base de paquet. Lors de l'arrivée d'un paquet dans la file d'attente, une date virtuelle lui est associée. Cette date appelée "*finish-time*" correspond à l'instant de transmission du dernier bit du paquet dans un système GPS. Tous les paquets dans les files d'attente ont des valeurs différentes de *finish-time*. L'algorithme choisit alors le paquet ayant la plus petite valeur de *finish-time* pour le transmettre sur le lien de sortie. Il est important de noter que le *finish-time* ne représente pas le vrai temps d'émission du paquet. Le calcul de la valeur du *finish-time* prend en considération la date d'arrivée du paquet, la longueur du paquet, le poids correspondant au flux de ce paquet et le débit de sortie. Le *finish-time* d'un paquet "k" du flux "i" arrivant à l'instant "t" est calculé selon la formule suivante:

$$F_i(k, t) = \max\{ F_i(k-1, t), R(t) \} + \frac{P_i(k, t)}{\phi(i)}$$

$P_i(k, t)$ est la longueur en bit du paquet "k" du flux "i" arrivant à l'instant "t".

$\phi(i)$ est le poids du flux "i".

R(t) est la valeur du "*Round number*" à l'instant "t". Le *Round number* désigne le numéro du cycle et il est incrémenté en fonction du débit de sortie et du nombre des flux actifs.

7.5 Les différentes catégories des piles protocolaires sur le *Last Mile Link*

D'après ce qui précède, nous avons vu que différentes piles protocolaires peuvent être utilisées pour le transport des flux sur le *Last Mile Link* de l'interface Iub. Elles dépendent de l'architecture utilisée pour la qualité de service et le multiplexage. Ces piles protocolaires (*Protocol Stack*) peuvent être divisées en deux grands groupes selon la version du protocole IP utilisée. Pour la version 4 du protocole IP, la compression d'en-tête peut être utilisée ou non. Pour IPv6, la compression d'en-tête est indispensable à cause de la longueur élevée de l'en-tête IPv6. D'après cette classification, nous pouvons diviser les solutions possibles en trois grandes catégories de piles protocolaires. Dans chaque catégorie, sept piles protocolaires sont envisagées selon l'architecture utilisée. Une comparaison entre les piles de chaque catégorie est nécessaire pour évaluer les avantages et les inconvénients de chaque solution.

7.5.1 Catégorie 1 : IPv4 avec compression d'en-tête

1. c(UDP/IPv4)/PPP/HDLC
2. c(UDP/IPv4)/PPP ML-MC/HDLC
3. c(UDP/IPv4)/PPP mux-ML-MC/HDLC
4. c(UDP/IPv4)/PPPmux-ML-MC/L2TP/c(UDP/IPv4)/PPP/HDLC
5. Propriété de France Télécom R&D (Architecture C)
6. c(UDP/IPv4)/PPPmux/AAL5/ATM

7. c(UDP/IPv4)/PPP/AAL2/ATM.

Le sigle "c" désigne que la compression d'en-tête est utilisée pour ce niveau.

7.5.2 Catégorie 2 : IPv4 sans compression d'en-tête

1. UDP/IPv4/PPP/HDLC
2. UDP/IPv4/PPP ML-MC/HDLC
3. UDP/IPv4/PPP mux-ML-MC/HDLC
4. c(UDP/IPv4)/PPPmux-ML-MC/L2TP/UDP/IPv4/PPP/HDLC
5. Propriété de France Télécom R&D (Architecture C)
6. UDP/IPv4/PPPmux/AAL5/ATM
7. UDP/IPv4/PPP/AAL2/ATM.

7.5.3 Catégorie 3 : IPv6 avec compression d'en-tête

1. c(UDP/IPv6)/PPP/HDLC
2. c(UDP/IPv6)/PPP ML-MC/HDLC
3. c(UDP/IPv6)/PPP mux-ML-MC/HDLC
4. c(UDP/IPv6)/PPPmux-ML-MC/L2TP/c(UDP/IPv6)/PPP/HDLC
5. Propriété de France Télécom R&D (Architecture C)
6. c(UDP/IPv6)/PPPmux/AAL5/ATM
7. c(UDP/IPv6)/PPP/AAL2/ATM.

7.6 Performances des piles protocolaires

Dans le paragraphe précédent, nous avons cité plusieurs solutions possibles pour le transport des flux dans l'UTRAN en utilisant le protocole IP. Le but de ce paragraphe est de faire une comparaison entre les performances des différentes solutions. Le critère essentiel de comparaison est l'efficacité de la bande passante. En d'autre terme, c'est la capacité en nombre d'utilisateurs que peut supporter un lien tout en respectant les limites des délais de chaque classe de service. D'ailleurs, les paramètres des protocoles utilisés constituent un sujet d'étude important. En fait, le protocole PPP-mux a 3 paramètres configurables [RFC 3153] :

- *MAX-SF-LEN* : C'est la taille maximale d'une sous-trame pouvant être multiplexée dans une trame PPP-mux. Si un paquet a une taille plus grande que MAX-SF-LEN, il ne peut pas être multiplexé avec d'autres paquets et il sera transmis dans sa propre trame PPP.
- *Maximum Receive Unit* (MRU): C'est la taille maximale d'une trame PPP-mux. Le processus de multiplexage s'arrête si la taille de la trame atteint cette valeur. Dans ce cas, la trame PPP-mux est envoyée.
- *Timer* : C'est un temporisateur pour limiter le temps de multiplexage lorsque le trafic est faible. Son rôle est semblable à celui du *Timer-CU* du protocole AAL2 mais dans le cas de PPP-mux, il n'y a pas d'octets de bourrage. Les trames PPP-mux ont des tailles variables.

Les valeurs de ces trois paramètres ont une influence sur le délai de transfert (à cause du temps de multiplexage) et sur l'efficacité de la bande passante. En effet, quand le nombre des sous-trames

multiplexées augmente, la charge due aux en-têtes sera amortie sur plusieurs paquets et par suite, l'utilisation de la bande passante devient plus efficace. C'est pourquoi, il est intéressant de choisir une valeur élevée du paramètre MRU. En revanche, si le trafic est faible et les paquets ont des petites tailles, le temps de multiplexage devient inacceptable pour des trames PPP-mux de grande taille. Il faut alors utiliser une valeur faible du *Timer* pour limiter le temps d'attente. Un compromis pour les différents paramètres est à prendre en considération selon la pile protocolaire utilisée.

La taille des segments PPP-MP est aussi un paramètre qui a une incidence sur le délai de transfert des paquets prioritaires. En effet, si un gros paquet AF est transmis sur le lien avant l'arrivée d'un paquet EF, ce dernier est obligé d'attendre la fin de transmission du gros paquet ce qui peut augmenter son délai de transfert. Si le gros paquet AF est segmenté en des petits paquets, le paquet EF peut être alors transmis juste après la transmission du premier segment du paquet AF. La taille du segment ne doit pas être très faible pour ne pas augmenter la taille des *overheads* et par conséquent réduire l'efficacité de la bande passante. Le choix de la taille du segment est alors un sujet d'étude selon l'architecture protocolaire utilisée.

7.6.1 Etude analytique de l'efficacité de la bande passante

Dans ce paragraphe, nous allons présenter les équations analytiques des débits moyens à l'entrée et à la sortie de chaque pile protocolaire afin de pouvoir comparer l'efficacité dans différentes solutions. L'efficacité en terme de bande passante est représentée par la quantité d'*overheads* introduits par la pile protocolaire. Nous la définissions par le rapport entre le débit entrant dans la pile protocolaire, c'est à dire le débit au niveau de la couche FP avec tous les en-têtes radio et FP, et le débit de sortie sur le lien physique avec tous les en-têtes de la pile protocolaire.

Tout d'abord, nous définissions quelques notations (voir chapitre 3 pour plus de détails sur les valeurs de ces paramètres):

- FP_PDU_size: taille moyenne d'une trame FP avec tous les en-têtes FP (en octet).
- ON et OFF sont respectivement les durées moyennes des périodes d'activité et de silence (en seconde).
- TTI : C'est le *Transmission Time Interval* du *Bearer Service* transportant le flux (en ms) (pour les *Bearer Service*, voir Annexe B).
- *ATM_payload*: c'est la taille de la charge utile d'une cellule ATM (48 octets).
- AAL2_max_payload: c'est la taille maximale de la charge d'une mini-cellule AAL2 (45 octets).
- Les paramètres suivants désignent les tailles des en-têtes (en octet) des protocoles correspondants (voir Annexe B pour les valeurs de ces paramètres): UDP, IP, PPP, PPP_mux, PPP_ML_MC, L2TP, AAL2, AAL5, ATM, HDLC.

  Remarque : Le paramètre IP peut désigner la taille d'un en-tête IPv4 ou IPv4, avec ou sans compression. De même pour "UDP" qui peut désigner la taille de l'en-tête UDP avec ou sans compression.
- CRC: C'est la taille du champ CRC dans le protocole PPP.
- *Segment_max_size* : C'est la taille maximale d'un segment PPP-MP (octet). Si un paquet a une taille plus grande que cette valeur, il sera segmenté en des petits paquets de taille maximale *Segment_max_size.*
- *PPPmux_max_size* : C'est la taille maximale d'un paquet PPP-mux (en octet). Ce paramètre représente le MRU du protocole PPP-mux.
- *Av_input* : C'est le débit moyen à la sortie de la couche FP (en kbit/s).

- *Av_output* : C'est le débit moyen à la sortie sur le lien physique (en kbit/s).
- $\lceil x \rceil$: C'est l'entier immédiatement plus grand ou égal à x.
- $\lfloor x \rfloor$: C'est l'entier immédiatement plus petit ou égal à x.

7.6.1.1 Débit moyen entrant

Le débit moyen (*Av_input*) par utilisateur à l'entrée de la pile protocolaire (à la sortie de la couche FP) est donné par:

a - Pour un flux AMR: $Av\_input = \frac{8 \times FP\_PDU\_size}{TTI} \times \frac{ON}{ON+OFF}$

b - Pour un flux vidéo: $Av\_input = \frac{8 \times FP\_PDU\_size}{TTI}$

c - Pour les autres types de flux: $Av\_input = \frac{8 \times FP\_PDU\_size}{TTI} \times \frac{ON}{ON+OFF}$

avec $ON = \frac{8 \times 1000 \times Packet\_call\_size}{1000 \times D}$, *Packet_call_size* est la taille moyenne d'un *packet-call* (voir chapitre 3); *D* est le débit maximal du *Radio Bearer Service* qui transporte ce flux (voir Annexe B).

7.6.1.2 Débit moyen sortant

Le débit moyen (*Av_output*) à la sortie sur le lien physique est calculé selon la pile protocolaire utilisée et la taille des paquets.

7.6.1.2.1 Pile 1 : UDP/IP/PPP/HDLC

Pour tous les paquets:

$$Av\_ouput = (FP\_PDU\_size + UDP + IP + PPP + HDLC) \times \frac{8}{TTI} \times \frac{ON}{ON+OFF}$$

7.6.1.2.2 Pile 2 : UDP/IP/PPP-ML-MC/HDLC

A - Pour les petits paquets (sans segmentation au niveau PPP-ML-MC):

$$Av\_output = (FP\_PDU\_size + UDP + IP + PPP\_ML\_MC + PPP + HDLC) \times \frac{8}{TTI} \times \frac{ON}{ON+OFF}$$

B - Pour les grands paquets (avec segmentation au niveau PPP-ML-MC):

$$Av\_output = \left(\left\lceil \frac{P}{Segment\_\max\_size} \right\rceil \times (PPP\_ML\_MC + PPP + HDLC) + P \right) \times \frac{8}{TTI} \times \frac{ON}{ON+OFF}$$

avec P = FP_PDU_size + UDP + IP

7.6.1.2.3 Pile 3 : UDP/IP/PPP-mux-ML-MC/HDLC

A - Pour les petits paquets (avec multiplexage au niveau PPP-mux):

$$Av\_ouput = (N \times P + PPP\_ML\_MC + PPP + HDLC) \times \frac{8}{N \times TTI} \times \frac{ON}{ON+OFF}$$

avec : $P=FP\_PDU\_size+UDP+IP+PPP\_mux$ et $N=\left\lfloor \frac{PPPmux\_\max\_size}{P} \right\rfloor$

B - Pour les grands paquets (avec segmentation au niveau PPP-ML-MC):

$$Av\_output=\left(N\times(PPP\_ML\_MC+PPP+HDLC)+P\right)\times\frac{8}{TTI}\times\frac{ON}{ON+OFF}$$

avec : $P=FP\_PDU\_size+UDP+IP$ et $N=\left\lceil \frac{P}{Segment\_\max\_size} \right\rceil$

*7.6.1.2.4 **Pile 4 : UDP/IP/PPP-mux-ML-MC/L2TP/UDP/IP/PPP/HDLC***

A - Pour les petits paquets (avec multiplexage):

$$Av\_output=\left(N\times P+PPP\_ML\_MC+PPP+L2TP+UDP+IP+PPP+HDLC\right)\times\frac{8}{N\times TTI}\times\frac{ON}{ON+OFF}$$

avec : $P=FP\_PDU\_size+UDP+IP+PPP\_mux$ et $N=\left\lfloor \frac{PPPmux\_\max\_size}{P} \right\rfloor$

B - Pour les grands paquets (avec segmentation):

$$Av\_ouput=\left(N\times\left(PPP\_ML\_MC+PPP+L2TP+UDP+IP+PPP+HDLC\right)+P\right)\times\frac{8}{TTI}\times\frac{ON}{ON+OFF}$$

avec : $P=FP\_PDU\_size+UDP+IP$ et $N=\left\lceil \frac{P}{Segment\_\max\_size} \right\rceil$

*7.6.1.2.5 **Pile 5 : (Propriété de France Télécom R&D)***

*7.6.1.2.6 **Pile 6 : UDP/IP/PPPmux/AAL5/ATM***

A - Pour les petits paquets (avec multiplexage au niveau PPP-mux et segmentation au niveau AAL5):

$$Av\_output=n\times\left(ATM\_payload+ATM\right)\times\frac{8}{N\times TTI}\times\frac{ON}{ON+OFF}$$

avec : $P=FP\_PDU\_size+UDP+IP+PPP\_mux$, $N=\left\lfloor \frac{PPPmux\_\max\_size}{P} \right\rfloor$

et $n=\left\lceil \frac{N\times P+PPP+AAL5}{ATM\_payload} \right\rceil$

B - Pour les grands paquets (sans multiplexage au niveau PPP-mux et avec segmentation au niveau AAL5):

$$Av\_output=\left\lceil \frac{P}{ATM\_payload} \right\rceil\times\left(ATM\_payload+ATM\right)\times\frac{8}{TTI}\times\frac{ON}{ON+OFF}$$

avec : $P=FP\_PDU\_size+UDP+IP+PPP+AAL5$

7.6.1.2.7 Pile 7 : UDP/IP/PPP/AAL2/ATM

Pour tous les paquets:

$$Av\_output=(N\times AAL2+P)\times\frac{53}{47}\times\frac{8}{TTI}\times\frac{ON}{ON+OFF}$$

avec: $P=FP\_PDU\_size+UDP+IP+PPP+CRC$ et $N=\left\lceil\frac{P}{AAL2\_\max\_payload}\right\rceil$

7.6.1.3 Comparaison entre les différentes piles protocolaires

Les équations du paragraphe §7.6.1.2 donnent le débit moyen par utilisateur à forte charge. Dans le cas de multiplexage (PPP-mux ou AAL2), on considère que le trafic entrant est suffisant pour remplir les paquets avant l'expiration du temporisateur. Pour cela, le temporisateur n'a pas d'influence et il n'est pas considéré dans les équations. Cette hypothèse est justifiable parce qu'en fait, le cas critique de l'efficacité de la bande passante se manifeste à forte charge.

Dans ce qui suit, nous allons utiliser les équations du paragraphe §7.6.1.2 pour comparer l'efficacité en terme de bande passante des différentes piles protocolaires.

7.6.1.3.1 Cas de multiplexage

Nous avons considéré le cas des petits paquets qui ont une taille plus petite que la taille maximale d'un paquet PPP-mux. Dans ce cas, les paquets sont multiplexés au niveau de la couche PPP-mux dans les piles 3, 4, 5 et 6. En fonction de la taille maximale d'un paquet PPP-mux, nous avons comparé l'efficacité des différentes piles.

Dans les trois figures ci-dessous, nous avons considéré le cas des paquets AMR ayant une taille de 39 octets. La pile 7 donne, en général, le plus petit taux d'efficacité qui est de 0,62. Cela est dû au grand nombre d'en-têtes introduits par cette pile. Dans le cas où les mécanismes de compression seraient utilisés, la pile 3 donne le meilleur taux d'efficacité qui peut atteindre 0,8. Pour l'architecture B (piles 1, 2, 3, 6 et 7), la pile 3 donne le meilleur taux d'efficacité. Les piles 4 et 5 (architecture A et C respectivement) se distinguent clairement des autres piles dans le cas de la catégorie 2 où la compression d'en-tête n'est pas réalisée. Pour une taille maximale de paquet PPP-mux, ces deux piles peuvent atteindre un taux d'efficacité proche du cas où la compression d'en-tête est réalisée (0,8 environ).

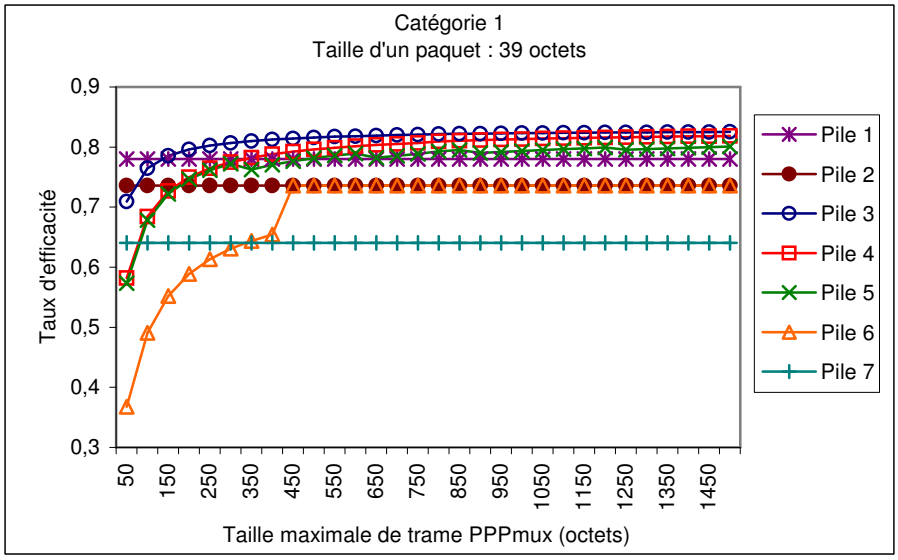


Figure 7.16 Flux AMR, catégorie 1

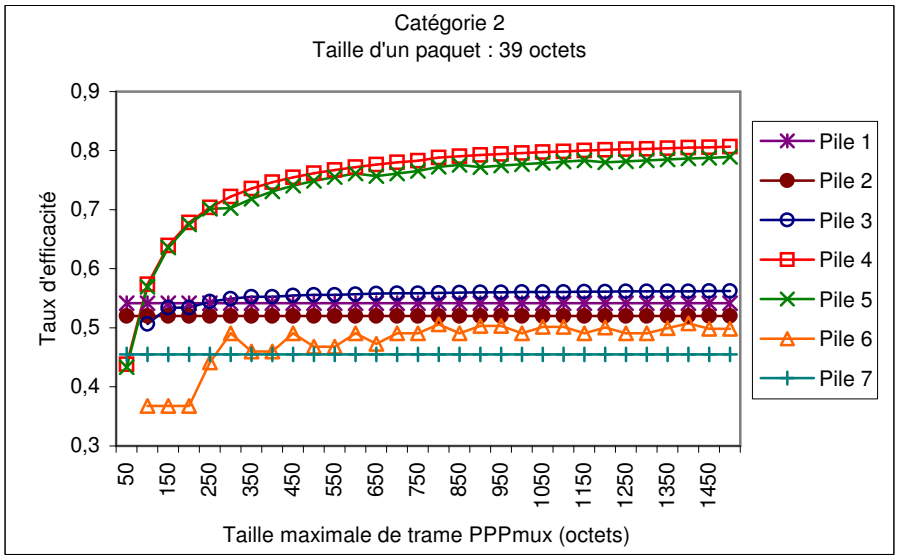


Figure 7.17 Flux AMR, catégorie 2

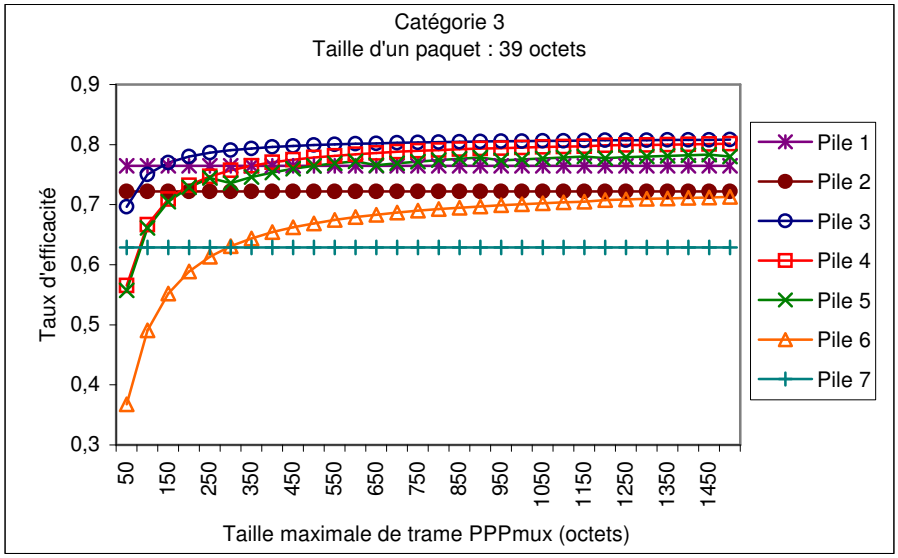


Figure 7.18 Flux AMR, catégorie 3

Dans tous les cas, l'effet de multiplexage devient avantageux à partir d'une valeur de 300 octets de la taille maximale d'une trame PPP-mux. Les très grandes tailles des trames PPP-mux n'apportent pas un avantage remarquable. D'ailleurs, le choix de la taille maximale d'une trame PPP-mux doit tenir compte du temps de remplissage. Il faut alors choisir la plus petite taille d'une trame PPP-mux qui apporte un taux d'efficacité maximal.

Nous avons fixé la taille d'une trame PPP-mux à 300 octets et nous avons calculé le taux d'efficacité en fonction de la taille des paquets. Les figures ci-dessous représentent les résultats obtenus. Les piles 6 et 7 donnent des mauvais taux d'efficacité. Les piles 4 et 5 donnent des meilleures performances dans le cas des très petits paquets.

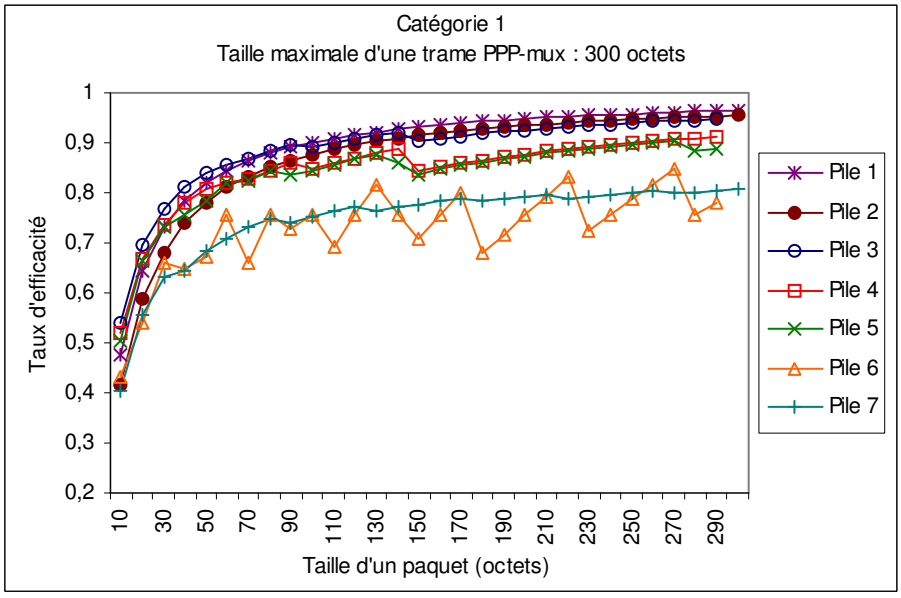


Figure 7.19 Catégorie 1 : Trame PPPmux = 300 octets

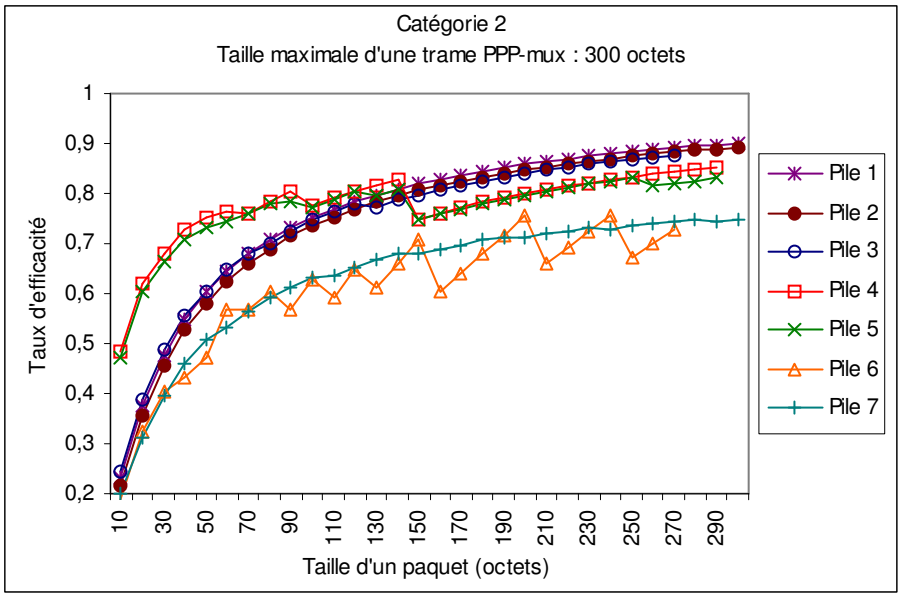


Figure 7.20 Catégorie 2 : Trame PPPmux = 300 octets

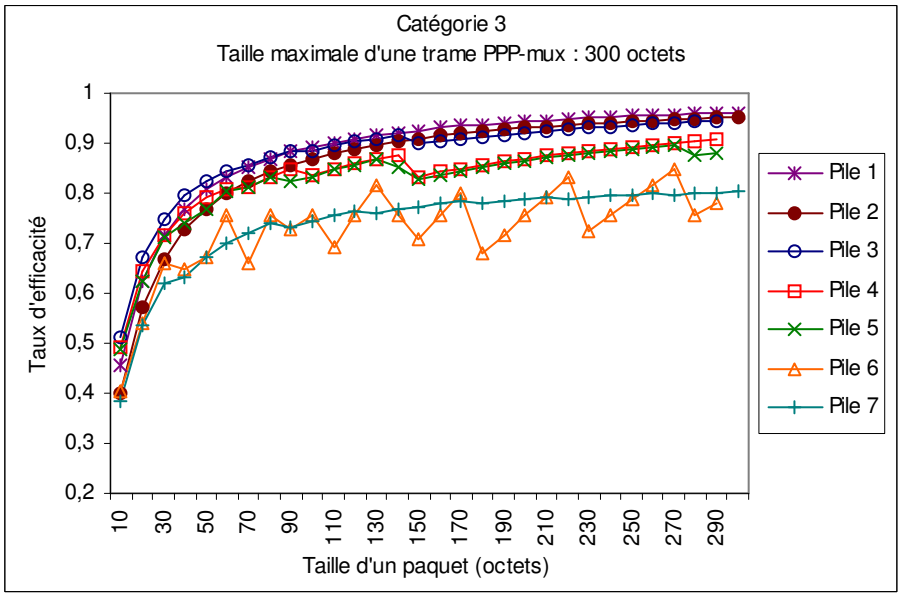


Figure 7.21 Catégorie 3 : Trame PPPmux = 300 octets

7.6.1.3.2 Cas de segmentation

Nous avons considéré le cas où les paquets ont des tailles plus grandes que la taille d'un segment PPP-MP. Dans ce cas, les paquets sont segmentés. Nous avons calculé le taux d'efficacité en fonction de la taille d'un segment PPP-MP et cela pour des trafics de type UDD (modes 64 et 384 kbit/s). La différence entre les deux modes est la taille des paquets FP-PDU (voir chapitre 3). Les figures ci-dessous représentent les résultats obtenus.

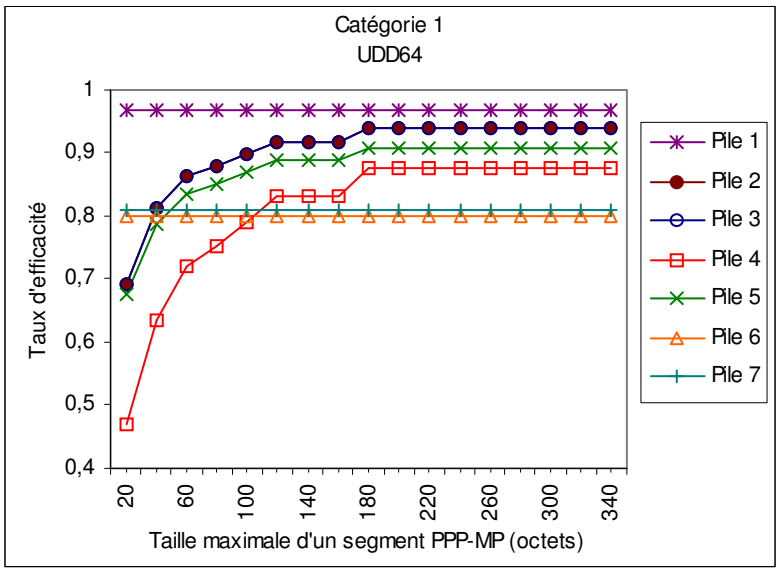


Figure 7.22 Catégorie 1, trafic UDD 64

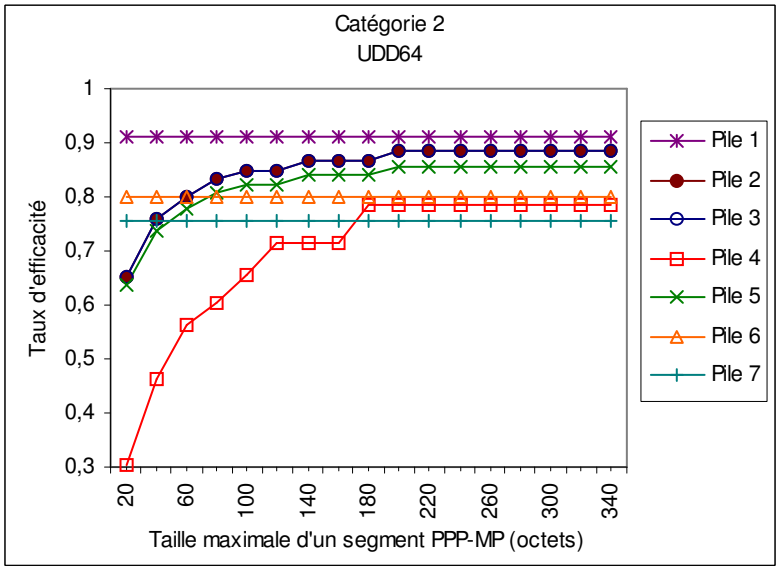


Figure 7.23 Catégorie 2, trafic UDD 64

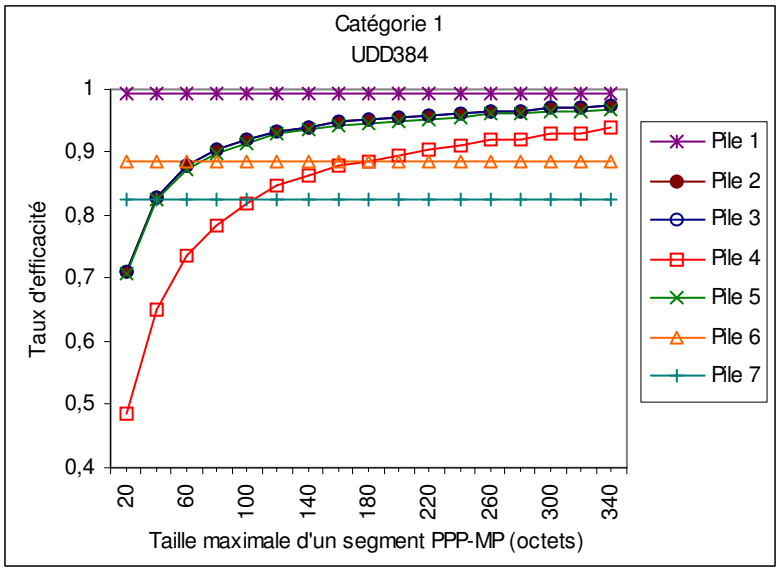


Figure 7.24 Catégorie 1, trafic UDD 384

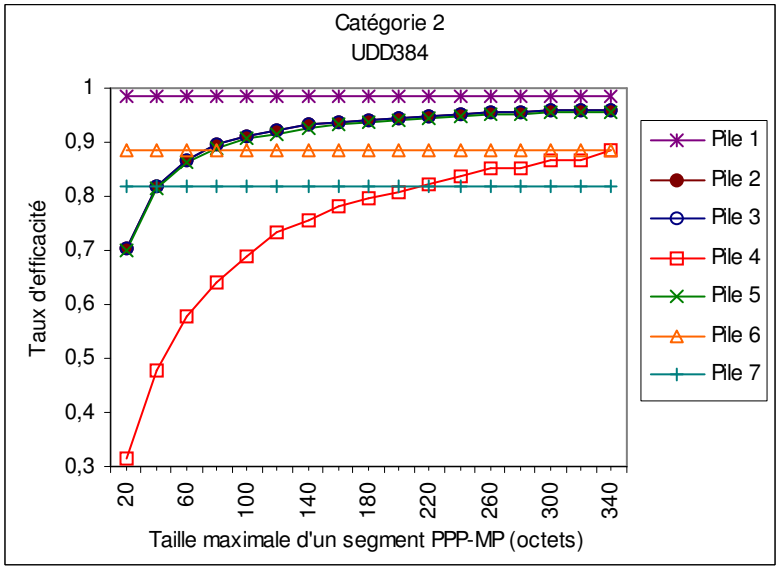


Figure 7.25 Catégorie 2, trafic UDD 384

A partir d'une taille de segment de 300 octets, le taux d'efficacité devient quasi-constant. Pour le trafic UDD 384, le taux d'efficacité est plus élevé que pour le trafic UDD 64. Cela est dû à la grande taille des paquets UDD 384. Les piles 2 et 3 ont les mêmes performances. La pile 4 donne le plus petit taux d'efficacité pour une petite taille de segment surtout dans le cas de l'absence de la compression d'en-tête. Cela est dû aux en-têtes UDP/IP du niveau inférieur qui sont ajoutés pour chaque segment PPP-MP ce qui augmente le nombre d'octets d'*overhead*. La segmentation pénalise l'efficacité de la bande passante mais elle est parfois nécessaire pour garantir un délai acceptable pour les paquets temps-réel lorsque ceux-ci sont agrégés avec des longs paquets sur le même support physique.

7.6.2 Etude par simulation

L'étude analytique faite dans le paragraphe précédent ne prend pas en compte tous les détails du système. Elle donne des résultats approximatifs. En pratique, les paquets ont des tailles variables et les instants exacts d'arrivée des paquets ont une influence sur la taille des paquets multiplexés et par conséquent sur le pourcentage des *overheads*. Le modèle analytique utilise des valeurs moyennes sans tenir compte de la finesse du système réel. Les méthodes analytiques deviennent très compliquées, voir impossibles, si on veut prendre en compte tous les détails du système. Dans ce cas, des modèles de simulation fins et complexes peuvent être développés pour analyser les performances du système. Ces modèles de simulation peuvent prendre en compte tous les détails du système avec une grande précision.

7.6.2.1 Le modèle de simulation

Pour évaluer les performances des différentes piles protocolaires et comparer leur efficacité dans le contexte de l'UTRAN, nous avons développé un modèle de simulation dans lequel un simulateur des couches radio (RLC et MAC) ainsi que de la couche FP est implémenté. Ce simulateur est le même qui est utilisé pour l'étude de l'AAL2 parce que, comme nous l'avons déjà dit, le réseau RNL est indépendant du réseau TNL et par suite, un changement de technologie dans le TNL n'affecte pas le RNL. Pour le TNL, nous avons développé les différents protocoles utilisés dans les architectures protocolaires (UDP, IP, L2TP, HDLC, PPP, PPPmux, PPP-ML-MC, AAL2, AAL5, ATM). Chaque protocole est implémenté dans un module du simulateur. Nous avons aussi implémenté tous les modèles de trafic définis dans le chapitre 3.

Le modèle fonctionnel du simulateur est présenté sur la figure 7.26. Les différentes applications sont transportées sur les supports de service radio (*Bearer service*) décrits dans l'annexe B. Les flux de voix et de vidéo-conférence (*Conversationnal*) sont transportés par la classe EF avec la priorité la plus élevée. Les flux de *web browsing* et de *video-streaming* sont transportés par la classe AF1. Les flux E-mail, SMS et FTP sont transportés par la classe AF2.

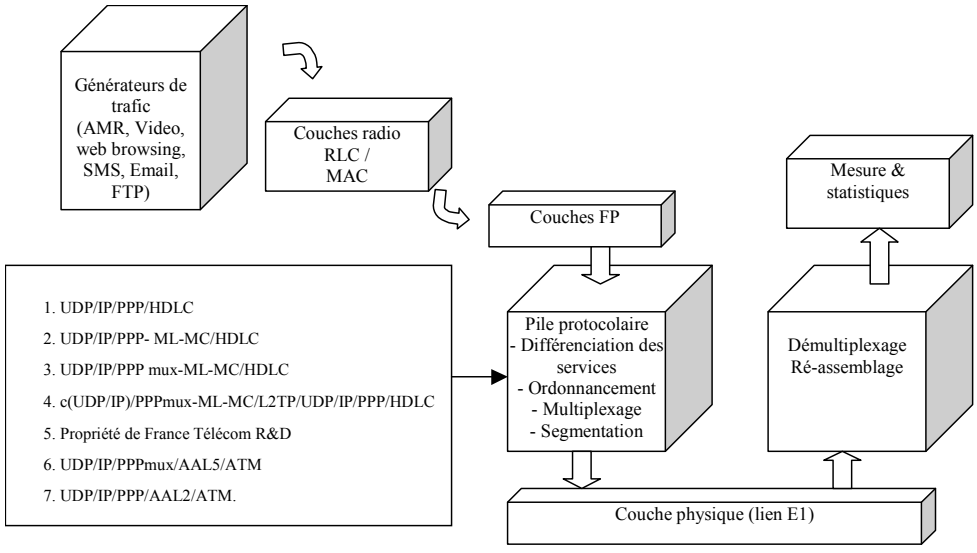


Figure 7.26 Modèle fonctionnel du simulateur

Plusieurs scénarios de trafic sont considérés. En fait, nous avons analysé les différentes piles protocolaires dans des contextes différents afin d'étudier la stabilité de leurs performances dans des situations différentes. Par exemple, le cas d'un trafic de voix majoritaire dans le réseau d'accès est intéressant. D'ailleurs, le trafic des applications Internet, comme le *web browsing* et les E-mail, va

devenir de plus en plus important dans les réseaux de mobiles. Le cas d'un trafic des données important est aussi considéré. Le tableau 7.1 représente les quatre scénarios étudiés avec la répartition des classes en terme de bande passante.

	Classe EF	Classe AF1	Classe AF2
Scénario A	100% AMR	-	-
Scénario B	80%	5%	15%
Scénario C	50%	25%	25%
Scénario D	20%	40%	40%

Tableau 7.1 Les scénarios de trafic

7.6.2.2 Les résultats et les conclusions

Comme nous l'avons déjà mentionné au début de ce chapitre, les résultats quantitatifs de cette étude ne seront pas publiés à cause de la confidentialité du projet dans lequel cette étude a été réalisée. En revanche, nous allons citer les principales conclusions de cette étude d'une manière qualitative. Cette étude porte sur les paramètres de qualité de service (délai et gigue) ainsi que sur l'efficacité de la bande passante pour les différentes piles protocolaires définies dans le paragraphe §7.5.

La catégorie 1 (IPv4 avec compression d'en-tête) et la catégorie 3 (IPv6 avec compression d'en-tête) donnent presque les mêmes résultats. Dans tous les scénarios, les piles 6 et 7 donnent le plus faible taux d'efficacité à cause du grand nombre d'*overhead* introduit par les en-têtes. En fait, la segmentation en des petits paquets aux niveaux AAL2 et AAL5 donne des mauvais résultats en terme d'efficacité de bande passante. La pile 6 donne toujours des mauvaises performances en terme de délai de transfert. En revanche, la pile 7 donne parfois des meilleures performances en terme de délai pour les flux EF surtout dans le cas de la catégorie 2 (IPv4 sans compression d'en-tête). En fait, les gros paquets AF des données sont segmentés en des petites mini-cellules et grâce aux mécanismes d'ordonnancement au niveau AAL2, les paquets EF peuvent passer sans beaucoup d'attente sur le lien physique. Les piles 4 et 5 donnent des taux d'efficacité semblables dans tous les cas. D'ailleurs la pile 4 donne des meilleures performances que la pile 5 en terme de délai de transfert. Dans le cas de la catégorie 2 (sans compression d'en-tête), les piles 4 et 5 se distinguent de toutes les autres piles et améliorent la capacité du lien en terme de nombre d'utilisateurs. Le protocole PPP-mux apporte d'amélioration en terme d'utilisation de la capacité du lien surtout dans le cas des paquets de petites tailles.

Les extensions ML-MC du protocole PPP peuvent être utilisées pour la segmentation des gros paquets et la différenciation des services au niveau PPP. Ces mécanismes améliorent les performances en terme de délai surtout lorsque le trafic EF est agrégé avec les trafics AF1 et AF2 ayant des paquets de grandes tailles et une tolérance pour le délai de transfert. Cet avantage devient intéressant lorsque le trafic AF est majoritaire.

Le trafic vidéo a été transporté sur la classe EF. Or, ce trafic dégrade la qualité de service des autres flux EF à cause de la grande taille de ses paquets. Dans le cas de *vidéo-streaming*, il est recommandé de transporter ce trafic sur la classe AF1. De toute manière, le trafic vidéo ne doit par être transporté sur des liens E1 parce qu'il dégrade considérablement la capacité du lien en terme de nombre d'utilisateurs. Il doit être transporté sur des liens de capacité plus élevée ou on peut profiter de l'extension ML du protocole PPP pour créer des liens virtuels à large bande à partir de plusieurs liens E1 à 1,92 Mbit/s.

7.7 Conclusion

Dans ce chapitre, nous avons parlé de l'introduction du protocole IP dans les réseaux d'accès UTRAN. Plusieurs solutions pour le transport en IP sont possibles selon le contexte et l'architecture du réseau de transport. Nous avons traité une méthode pour garantir la qualité de service pour les flux transportés sur l'interface Iub de l'UTRAN. Une comparaison entre les différentes solutions de transport en IP est réalisée. Cette étude permettra aux opérateurs des réseaux mobiles de choisir la solution convenable selon leurs besoins ainsi que selon la configuration de leurs réseaux.

Ce chapitre n'a pas été traité d'une manière détaillée à cause de l'aspect confidentiel du projet dans lequel cette étude a été menée avec France Télécom R&D. Plusieurs aspects importants et résultats intéressants n'ont pas été présentés dans ce chapitre. Malheureusement, ce fait n'a pas donné à cette étude la valeur souhaitée.

Chapitre 8

8 Conclusion générale et ouvertures

8.1 Conclusion

Dans notre thèse, nous avons traité le sujet de la qualité de service et les performances des protocoles de transport dans les réseaux radio d'accès terrestre de l'UMTS (UTRAN). Le réseau d'accès UTRAN présente des exigences particulières en terme de délai de transfert à cause des contraintes temporelles imposées par les mécanismes de synchronisation des couches radio. En effet, les canaux radio de l'UTRAN sont transportés entre le RNC et le Node B sur les protocoles de transport des interfaces Iub et Iur. Ces protocoles doivent être capables de fournir la qualité de service nécessaire pour les flux transportés.

Dans une partie de notre thèse, nous avons traité le sujet du protocole AAL2. Le protocole AAL2/ATM a été choisi par le 3GPP dans sa Release 99 comme solution de transport sur les interfaces Iub et Iur de l'UTRAN et pour tous les types de flux. Or, dans les spécifications de l'UIT-T, il n'y a pas de définition de la qualité de service au niveau de la couche AAL2. En plus, les études faites dans ce contexte sont rares et aucune étude n'a proposé une architecture complète de la qualité de service au niveau AAL2. Nous avons alors contribué à la définition de quatre classes de services avec des paramètres de QoS à deux niveaux : le niveau SSCS et le niveau CPS. Ces paramètres et ces classes de services servent à la supervision de la qualité de fonctionnement du réseau et des services rendus aux différents utilisateurs. Une classe SRT est définie pour les applications très exigeantes en terme de délai de transfert. Une classe TRT est définie pour les applications qui tolèrent un certain délai et qui ne sont pas sensibles à la gigue mais qui sont plus sensibles au taux de perte. Une classe NRT pour les applications non temps-réel mais qui sont très sensibles au délai. Enfin, la classe BE est définie pour les applications qui n'ont pas des contraintes temporelles ni de taux de perte. Puisque dans l'UTRAN, les flux transportés deviennent tous temps-réel, seulement les classes SRT et TRT sont utilisées pour les applications transportées dans l'UTRAN. Les classes NRT et BE peuvent être utilisées dans d'autres contextes comme celui du *truncking*. Nous avons aussi défini des algorithmes de conformité aux deux niveaux pour le contrôle des flux dans les réseaux de transport. Les algorithmes de conformité du niveau SSCS servent à contrôler les flux à l'entrée d'un sous réseau AAL2. Les algorithmes de conformité au niveau CPS servent à contrôler les flux dans les commutateurs AAL2 au sein d'un sous-réseau AAL2. Trois capacités de transfert sont aussi définies au niveau AAL2 pour permettre le transport des applications de différents profils ainsi que pour pouvoir définir des schémas de réservation des ressources et de contrôle d'admission. La capacité de transfert AAL2-CBR est utilisée pour les applications à débit constant où la réservation des ressources est effectuée selon le débit crête. La classe AAL2-VBR pour les applications à débits variables utilise un débit soutenu avec une taille maximale de rafale pour la réservation des ressources. Une troisième classe appelée AAL2-ABR est définie mais elle n'est pas utilisée dans le contexte de l'UTRAN parce qu'elle ne garantit pas un débit minimal pour les flux transportés. Des relations entre les paramètres des deux niveaux CPS et SSCS sont définies.

Un schéma d'association entre les classes de services de l'UMTS et celles de l'AAL2 ainsi que celles de l'ATM est défini. Un schéma d'association entre les ATC et les AAL2-TC est aussi défini avec une méthode d'allocation des ressources et de contrôle d'admission basée sur les paramètres du

profil du trafic. Un autre schéma d'allocation des ressources et de contrôle d'admission en fonction de la bande passante équivalente est encore défini avec les conditions nécessaires pour son applicabilité.

L'AAL2 est un protocole nouveau et les études menées pour l'évaluation de ses performances ne sont pas nombreuses, surtout dans le contexte sévère de l'UTRAN. Une étude fine et détaillée des performances du protocole AAL2 dans le contexte particulier de l'UTRAN est alors réalisée dans le cadre de notre thèse. Nous avons étudié l'influence du Timer-CU sur les performances du protocole AAL2 pour le transport des applications temps-réel ainsi que sur l'efficacité de la bande passante. Une conclusion sur l'influence de ce paramètre dans le contexte de l'UTRAN permet de dire qu'une valeur comprise entre 1 ms et 2 ms est une valeur optimale qui satisfait les besoins de l'UTRAN en terme d'optimisation de la bande passante et de garantie des bornes des délais.

Plusieurs mécanismes d'ordonnancement sont proposés au niveau AAL2 et une comparaison entre leurs performances est réalisée. Un algorithme de type WRR dynamique appelé DyWRR est aussi proposé pour contourner la problématique du choix des poids du WRR. Cet algorithme a montré une bonne efficacité surtout lorsque le trafic change parce qu'il s'adapte automatiquement aux nouvelles situations en recalculant les poids des files d'attente.

Deux ATC au niveau ATM sont proposés pour transporter les flux AAL2: le DBR et le SBR. L'avantage du DBR est la simplicité de son implémentation. Son inconvénient est la perte de bande passante dans le cas des flux à débits variables. L'avantage du SBR est l'optimisation de l'utilisation de la bande passante dans le cas des flux à débits variables et cela grâce à l'effet du gain statistique. Or, l'implémentation du SBR est assez compliquée, c'est pourquoi nous avons étudié le gain statistique qu'il peut apporter et qui peut récompenser la complexité de son implémentation pour pouvoir justifier le choix de son utilisation.

La commutation AAL2 est indispensable dans certains cas. Elle peut être aussi utilisée pour ajouter un deuxième niveau d'agrégation des flux venants de plusieurs Node B. Or, la commutation AAL2 introduit des délais supplémentaires à cause des mécanismes additionnels. Une comparaison entre la commutation AAL2 et la commutation ATM est réalisée pour étudier les profits que peut apporter la commutation AAL2.

Dans la dernière partie de notre thèse, nous avons traité le sujet de l'introduction du protocole IP dans le réseau de transport de l'UTRAN. Trois architectures sont proposées pour la qualité de service dans un UTRAN IP. Trois classes *DiffServ* sont utilisées pour transporter les différents flux de l'UTRAN. Un deuxième niveau de différenciation des services est introduit au niveau de la couche 2. Un schéma d'association entre les classes de l'UMTS et celles des niveaux inférieurs est présenté. Plusieurs solutions de transport sont étudiées et comparées analytiquement et par simulation pour en tirer une conclusion sur l'efficacité de chaque solution en terme de bande passante.

Dans cette thèse, nous avons étudié plusieurs sujets concernant la qualité de service et les performances des protocoles de transport dans les réseaux mobiles d'accès terrestre de troisième génération. Cette étude sera utile pour les opérateurs des télécommunications mobiles pour le déploiement de leurs premiers réseaux d'accès où la bande passante est une ressource chère même dans les réseaux filaires à cause des grandes distances entre les différents nœuds du réseau de transport.

Enfin, signalons que les résultats des travaux faits dans le cadre de notre thèse sur le protocole AAL2 ont été repris par France Télécom R&D pour une contribution à la normalisation au sein de l'UIT-T. Cette contribution a abouti à une nouvelle recommandation de l'UIT-T traitant la qualité de service et le contrôle de trafic au niveau de la couche de transport AAL2. Cette recommandation a été publiée sous le nom « ITU-T I.378 : *Traffic Control and Congestion Control at the ATM Adaptation Layer type 2* ».

8.2 Ouvertures

Pendant notre thèse, nous n'avons pas eu le temps de traiter tous les sujets concernant les réseaux d'accès. En fait, c'est un sujet vaste et plusieurs axes d'études peuvent être envisagés dans le futur. Le premier axe d'étude concerne la gestion des *handover*. En effet, quand un utilisateur est en *soft handover*, il établit plusieurs connexions avec le réseau d'accès. Ce nombre élevé de connexions pour un même utilisateur peut augmenter le trafic sur les interfaces Iub et Iur et par suite une gestion de ces connexions est nécessaire pour optimiser l'utilisation des ressources disponibles. Une stratégie d'allocation des ressources est nécessaire pour diviser la bande passante disponible dans un Node B en deux parties: une partie pour les nouveaux appels et une autre partie pour les appels en *handover*. Les derniers sont prioritaires par rapport aux premiers, mais il faut aussi considérer les nouveaux appels ayant une classe de service qui ne tolère pas une grande latence pour l'établissement de l'appel. Des modèles sophistiqués de mobilité doivent être proposés puis utilisés pour la validation des nouvelles propositions. Ce sujet peut être traité dans le cas de l'AAL2 et de l'IP. Dans le deuxième cas, des protocoles existants comme *Cellular IP* ou *Mobile IP* peuvent être utilisés et améliorés pour la gestion de la mobilité dans les réseaux d'accès.

Un autre axe d'étude important est le sujet de l'interopérabilité entre des réseaux d'accès basés sur la technologie AAL2/ATM et d'autres basés sur la technologie IP. En effet, les spécifications du 3GPP prévoient la coexistence entre ces deux types de réseaux d'accès. Dans le cas où un flux traverserait plusieurs nœuds appartenant à des réseaux de technologies différentes, ces réseaux doivent garantir à ce flux la qualité de service négociée dans le réseau d'entrée. Une correspondance entre les classes des différents réseaux est nécessaire. En plus, les mécanismes de contrôle de flux des différents réseaux doivent être cohérents. La question de la mobilité se pose encore dans ce contexte. En effet, quand un utilisateur est en soft *handover* avec deux Node B appartenant à deux réseaux de technologies différentes, il est nécessaire de synchroniser les flux sur les deux interfaces pour pouvoir appliquer les mécanismes de recombinaison et de macro-diversité dans le S-RNC de destination. Ce sujet nécessite une étude complémentaire.

Enfin, les mêmes sujets traités dans le contexte de l'UTRAN (AAL2 ou IP) peuvent être étudiés dans le cas d'un réseau d'accès ayant une composante satellite. C'est le réseau d'accès USRAN (*UMTS Satellite Radio Access Network*) dans lequel le temps de transmission des données est un paramètre considérable à cause de la grande distance terre-satellite. Ce délai supplémentaire affecte les performances des protocoles de transport. Pour cela, une étude détaillée sur ce sujet serait intéressante.

Liste des publications

1. R. Makké, S. Tohmé, J-Y. Cochennec, S. Pautonnier. "Performance Evaluation of the AAL2 Protocol within the UTRAN". Annals of Telecommunications Journal, Vol. 58, N° 7-8, juillet-août 2003, pp. 1041-1065.

2. R. Makké, S. Tohmé, J-Y. Cochennec, S. Pautonnier. "Quality of Service and Admission Control within the UMTS Terrestrial Radio Access Network". 18th World Telecommunications Congress WTC'2002, Disneyland-Paris, France, 22-27 septembre 2002.

3. R. Makké, S. Tohmé, J-Y. Cochennec, S. Pautonnier. "Architecture de QoS et Schéma d'Allocation des Ressources dans le Réseau d'Accès de l'UMTS". Colloque Francophone sur l'Ingénierie des Protocoles CFIP 2002, Montréal, Canada, 27-30 mai 2002.

4. R. Makké, S. Tohmé, J-Y. Cochennec, S. Pautonnier. "Performance of the AAL2 Protocol within the UTRAN". IEEE 2nd European Conference on Universal Multiservice Networks ECUMN'2002, Colmar, France, 8-10 avril 2002.

   (choisi comme meilleur article dans la conférence)

5. R. Makké, S. Tohmé, J-Y. Cochennec, S. Pautonnier. "Architecture de QoS et Performances du Protocole AAL2 dans les Réseaux d'Accès de l'UMTS". Journées Doctorales Informatiques et Réseaux JDIR'2002, Toulouse, France, 4-6 mars 2002.

6. R. Makké, S. Tohmé, J-Y. Cochennec, S. Pautonnier. "Some Issues in Performance of the AAL2 Multiplexer in the Context on the UTRAN". IFIP Workshop on IP and ATM Traffic Management WATM'2001 & EUNICE Summer School 2001, Paris, France, 3-5 septembre 2001.

7. R. Makké, S. Tohmé, J-Y. Cochennec, S. Pautonnier. "Impact of the Timer-CU of the AAL2 Protocol on Traffic Performance within the UTRAN". IFIP Personal Wireless Communications PWC'2001 conference, Lappeenranta, Finlande, 8-10 août 2001.

   (Prix du meilleur article : Best Paper Award)

8. J-Y. Cochennec, J-P. Quinquis, R. Makké, S. Tohmé, S. Pautonnier. "L'AAL2 dans les réseaux d'accès UMTS". Digital Communications and Services magazine (Novamedia), numéro 9, mars-avril 2001.

9. J-Y. Cochennec, O. Roussel, R. Makké, S. Tohmé, S. Pautonnier. "L'AAL2 dans l'UTRAN: les résultats du projet RNRT Minicel". Networks Developments conference, Rennes, France, 20-22 mars 2001.

Références bibliographiques

[1] Ali M. Dawood and Mohammed Ghanbari. Content-Based MPEG Video Traffic Modeling. IEEE Transactions on Multimedia, Vol.1, No.1, March 1999

[2] Andras Vargaa. Site web du simulateur Omnet++ : http://whale.hit.bme.hu/omnetpp/

[3] Averill M. Law and W. David Kelton. Simulation, Modeling and Analysis. Mc-Graw-Hill Book Company. 1982.

[4] Guy Pujolle. Les Réseaux. Eyrolles. 3ème édition, 2000

[5] H. Sariowan, Rene L. Cruz, George C. Polyzos. SCED: A Generalized Scheduling Policy for Guaranteeing Quality of Service. IEEE/ACM (Association for Computing Machinery) Transactions on Networking, Vol. 7, No. 5, October 1999.

[6] Hiroshi Saito. Bandwidth Management for AAL2 in UBR VCs. IEEE ATM Workshop, 1999, pp. 77-92

[7] Hiroshi Saito. Bandwidth Management for AAL2 traffic. IEEE Transactions on Vehicular Technology, Vol. 49, No. 4, July 2000, pp. 1364-1377

[8] Hiroshi Saito. Performance Evaluation and Dimensioning for AAL2 CLAD. IEEE INFOCOM, Vol.1, 1999, pp. 153-160

[9] Hiroshi Saito. Performance Evaluation of AAL2 Switching Nodes and a Network using them. IEEE ICC, Vol.3, 1999, pp. 1807-1813

[10] Hiroshi Saito. Performance Evaluation of AAL2 voice multiplexer. Elsevier Performance Evaluation. No. 42, 2000, pp. 57-83

[11] J.H. Baldwin, B.H. Bharucha, B.T. Doshi, S. Dravida and S. Nanda. AAL2- A new ATM adaptation layer for small packet encapsulation and multiplexing. Bell Labs Technical Journal, Vol.2, No. 2, 1997, pp. 111-131

[12] J.H. Baldwin, B.H. Bharucha, B.T. Doshi, S. Dravida and S. Nanda. AAL-CU: An ATM adaptation layer for small packet encapsulation. International Teletraffic Congress ITC 1997, pp. 1445-1456.

[13] Jean-Lien c.Wu, Chi-Hong Huang and Rong-Tsung Sheu. Performance study of AAL2 protocol for low-bit-rate multimedia services. IEEE 15th International conference on information networking, 2001. pages 793-798

[14] K. Sriram and Y.T Wang. Voice over ATM using AAL2 and Bit Dropping: Performance and call admission control. IEEE Journal on Selected Areas in Communications JSAC, Vol.17, No.1, January 1999

[15] K. Sriram, Terry G. Lyons and Yung-Terng Wang. Anomalies Due to Delay and Loss in AAL2 packet Voice Systems : Performance Models and Methods of Mitigation. IEEE JSAC, Vol. 17, No. 1, January 1999.

[16] Kurt Aretz, Martin Haardt, Walter Konhauser, worner mohr. The future of wireless communications beyond the third generation. Elseiver, Computer Networks No. 37 (2001), pp. 83-92.

[17] Marco Boldt, Ulrich Weiss, Rolf Sigle, Ulrich Dropmann. Modeling an ATM-Based Access Network for 3rd Generation Mobile Communication Networks. IEEE VTC'98, pp. 2590-2593.

[18] O. Isnard, J.M. Calmel, A.L. Beylot and G. Pujolle. Handling Traffic Classes at AAL2/ATM layer over the Logical Interfaces of the UMTS Terrestrial Radio Access Network. IEEE PIMRC, September 2000.

[19] O. Isnard. Etude du protocole AAL2 dans le réseau d'accès radio terrestre UMTS. Thèse de doctorat de l'université de Versailles Saint-Quentin en Yvelines. 2000.

[20] Rapports techniques du projet RNRT Minicel. France Télécom R&D, Mitsubishi Electric ITE et ENST-Paris.

[21] Rapports techniques du projet TIPS. France Télécom R&D et ENST-Paris

[22] S. Nananukul, Y. Guo, M. Holma and Sami Kekki, Some issues in performance and design of the ATM/AAL2 transport in the UTRAN. IEEE Wireless Communications and Networking Conference, September 2000.

[23] Site web du 3GPP : http://www.3gpp.org

[24] Tiziana Ferrari. End-to-end performance analysis with traffic aggregation. Elsevier computer networks, Vol. 34, 2000, pp. 905-914.

[25] G. Eneroth, G. Fofor, G. Leijonhufvud, A. Racz and I. Szabo. Applying ATM/AAL2 as a switching technology in third generation mobile access networks. IEEE Communications Magazine. Vol. 37, Issue 6, June 1999. pp. 112-122.

[26] I. Lopez, P. Ameigeiras, J. Wigard, P. Mogensen. Downlink radio resource ma,agement for IP packet services in UMTS. IEEE VTC 2001, Spring, 6-9 May 2001, Vol. 4, pp. 2387-2391.

[27] E.J. Hernandez-Valencia and M.C. Chuah. Transport Delays for UMTS VoIP. IEEE Wireless communications and networking conference WCNC 2000, 23-28 September 2000. Vol. 3, pp. 1552-1556.

[28] K. Venken, D. De Vleeschauwer and J. De Vriendt. Designing a DiffServ-capable IP-Backbone for the UTRAN. IEE 3G Mobile Communication Technologies, 26-28 March 2001, No. 477, pp. 47-52.

[29] Nishith D. Tripathi. Simulation based analysis of the radio interface performance of an IS-2000 system for various data services. IEEE VTC 2001 Fall, 7-11 October 2001, Vol. 4, pp. 2665-2669.

[30] I. Szabo. A flexible signalling protocol for supporting switched AAL type 2 connections in UMTS terrestrial radio access networks. IEEE 2nd international conference on ATM, ICATM'99, 21-23 June 1999, pp. 67-71.

[31] C. Semeria,. Supporting differentiated service classes: queue scheduling disciplines. Site web de Juniper Networks. http://www.juniper.net. Part number: 200020-001 12/01.

[32] J. Manner, A. Lopez Toledo, A. Mihailovoc, H. L. Velayos Munoz, E. Hepworth, Y. Khouaja. Evaluation of mobility and quality of service interaction. Elsevier Computer Networks, Vol. 37, 2002, pp. 137-163.

[33] A.M. Dawood and M. Ghanbari. MPEG video model based on scene content. Electronics letters, 30th April 1998, Vol. 34, No. 9, pp. 868-870.

[34] C. Metz. A pointed look at the point-to-point protocol. IEEE Internet Computing, July-August 1999. http://computer.org/internet, pp. 85-88.

[35] S. Floyd and V. Jacobson. Link-sharing and resource management models for packet networks. IEEE/ACM Transactions on Networking, Vol. 3, No. 4, August 1995.

[36] I. Lesjak and R. Trobec. Voice and data integration in HDLC environment. IEEE CompEuro '89, VLSI and Computer Peripherals. VLSI and Microelectronic Applications in Intelligent Peripherals and their Interconnection Networks, 8-12 May 1989, pp. 4/12-4/16.

[37] G. Fodor, G. Leijonhufvud, S. Malomosky, A. Racz. Comparison of call admission control algorithms in ATM/AAL2 based 3rd generation mobile access networks. IEEE Wireless Communications and Networking Conference, WCNC 1999, 21-24 September 1999 Vol 3, pp. 1508 –1512.

[38] I. Szabo, S. Szekely, I. Moldovan. Performance optimisation of AAL2 signalling for supporting soft handoffs in UMTS terrestrial radio access networks, Fifth IEEE Symposium on Computers and Communication, ISCC 2000, 3-6 July 2000, pp. 46 –51.

[39] D.W. Petr, R.R. Vatte, P. Sampath, Y-Q. Lu. Efficiency of AAL2 for voice transport: simulation comparison with AAL1 and AAL5. IEEE International Conference on Communications, ICC '99, Vol. 2, 6-10 June 1999 , pp. 896 –901.

[40] P. Sen, B. Maglaris, N-E. Rikli and D. Anastassiou. Models for packet switching of variable-bit-rate video sources. IEEE JSAC, Vol. 7, No. 5, June 1989, pp. 865-869.

[41] H. Nyberg, C. Johansson, B. Olin. A streaming traffic model for the mobile access network. Vehicular Technology Conference VTC 2001 Fall, IEEE VTS 54th, 7-11 October 2001, Vol. 1, pp. 423 –427.

[42] B. Subbiah. Transport architecture evolution in UMTS/IMT-2000 cellular networks**.** IEEE Global Telecommunications Conference GLOBECOM 2000, Vol. 1, 27 November-1 December 2000, pp. 231 –235.

[43] B. Subbiah, S. Dixit. ATM adaptation layer 2 (AAL2) for low bit rate speech and data: issues and challenges. IEEE ATM Workshop Proceedings, 1998, 26-29 May 1998, pp. 225 –233.

[44] G. Sfikas,S. Palat, I. Kriaras. Migration to All IP UMTS networks: transport and mobility issues. IEEE Wireless Communications and Networking Conference WCNC 2000, 23-28 September 2000, Vol. 2, pp. 787 –792.

[45] *R.S. Pazhyannur, I. Ali; I.N. Vukovic.* PPPmux-a new protocol for transporting small IP packets.IEEE International Conference on Communications ICC 2001, 11-14 June 2001 Vol. 8, pp. 2472 –2477.

[46] D. Saha, S. Mukherjee and S. K. Tripathi. Carry-over Round Robin: A simple cell scheduling mechanism for ATM Networks. IEEE/ACM Transactions on networking, December 1998, Vol. 6, No. 6, pp. 779-796.

[47] B. Subbiah, Y. Raivio. Transport architecture for UMTS/IMT-2000 cellular networks IEEE International Performance, Computing, and Communications Conference IPCCC 2000, 20-22 February 2000, pp. 208 –214.

[48] N.H. Loukas, C.K. Xenakis, L. Merakos.Transport performance evaluation of an ATM-based UMTS access network**.** IEEE 6th International Conference on Universal Personal Communications Record, 1997, 12-16 October 1997, Vol. 1, pp. 316 –320.

[49] S. Gruhl A. Echihabi, T. Rachidi, M. Link and M. Sollner. A demonstrator for real-time multimedia sessions over 3rd generation wireless networks. IEEE International Conference on Multimedia and Expo, ICME 2000, 30 July-2 August 2000, Vol. 2, pp. 959 –962.

[50] D. Turina. Challenges of real-time IP support in GSM/EDGE radio access network. The 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2000, 18-21 Sept. 2000, Vol. 1, pp. 8 –12.

[51] Y. Xu, R. Guerin. Individual QoS versus aggregate QoS: A loss performance study. INFOCOM 2002, 21st Annual Joint Conference of the IEEE Computer and Communications Societies, 23-27 June 2002, Vol. 3, pp. 1170 –1179.

[52] S.S. Chakraborty. The interworking approach for narrowband access to ATM transport-based multiservices mobile networks. IEEE Personal Communications, Vol. 5, Issue 4, August 1998 , pp. 70 –79.

[53] H. Koraitim, S. Tohmé. Resource allocation and connection admission control in satellite networks. IEEE JSAC, Vol. 17, No. 2, February 1999.

[54] B. Maglaris, D. Anastassiou, P. Sen, G. Karlsson and J. D. Robbins. Performance models of statistical multiplexing in packet video communications. IEEE transactions on communications, Vol. 36, No. 7, July 1988, pp. 834-843.

[55] X. Lagrange, P. Godlewski, S. Tabbane. Réseaux GSM-DCS, des principes à la norme. Edition Hermes, Paris 1995.

[56] M. Schwartz. Telecommunication networks, protocols, modelling and analysis. Addison-Wesley publishing company. May 1987.

[57] F. Kamoun, L. Kleinrock. Analysis of shared finite storage in a computer network node environment under general traffic conditions. IEEE Transactions on communications, Vol. 28, 1980, pp. 992-1003.

[58] L. Kleinrock. Queueing systems Vol 1: Theory. Editor: John Wiley & sons, New York. 1975.

[59] L. Kleinrock. Queueing systems Vol 2 : Computer applications. Editor: John Wiley & sons, New York. 1976.

[60] A. S. Tanenbaum. Computer networks. Editor: Prentice Hall, Upper Saddle River, NJ, 3rd edition, 1996.

Les recommandations de l' ITU-T

[61] ITU-T I.356. B-ISDN ATM layer cell transfer performance

[62] ITU-T I.366.1. Segmentation and Reassembly Service Specific Convergence Sublayer for the AAL type 2.

[63] ITU-T I.366.2. AAL type 2 service specific convergence sublayer for narrow-band services

[64] ITU-T I.363.2. B-ISDN ATM Adaptation Layer specification : Type 2 AAL

[65] ITU-T I.363.5. B-ISDN ATM Adaptation Layer specification : Type 5 AAL

[66] ITU-T I.371. Traffic control and congestion control in B-ISDN

[67] ITU-T Q.2630.1. AAL Type 2 Signalling Protocol (Capability Set 1)

[68] ITU-T Y.1541 Objectifs de qualité de fonctionnement pour les services en mode IP

[69] ITU-T Y.1221 Gestion du trafic et des encombrements dans les réseaux en mode IP

[70] ITU-T I.361 B-ISDN ATM Layer specification

[71] ITU-T Q.2630.2. AAL type 2 signalling protocol - Capability set 2

Les RFC de l'IETF

[72] IETF RFC 1661. The Point-to-Point Protocol (PPP), July 1994

[73] IETF RFC 1662. PPP in HDLC-like Framing, IETF STD 51, July 1994

[74] IETF RFC 1990. The PPP Multilink Protocol (MP), August 1996

[75] IETF RFC 2460. Internet Protocol, Version 6 (IPv6) Specification, December 1998

[76] IETF RFC 2507. IP header compression, February 1999

[77] IETF RFC 2509. IP Header Compression over PPP, February 1999

[78] IETF RFC 2661. Layer 2 Tunneling Protocol, August 1999

[79] IETF RFC 2686. The Multi-Class Extension to Multilink PPP, September 1999

[80] IETF RFC 3095. RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP and uncompressed, July 2001

[81] IETF RFC 3153. PPP Multiplexing, August 2001

[82] IETF RFC 768. User Datagram Protocol, August 1980

[83] IETF RFC 2475. An architecture for Differentiated Services, December 1998

[84] IETF RFC 791. Internet Protocol Specification , September 1981

[85] IETF RFC 2508. Compressing IP/UDP/RTP Headers for Low-Speed Serial Links, February 1999.

[86] IETF RFC 2474. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 header, December 1998

Les spécifications du 3GPP

[87] ETSI TR 101.112. Universal Mobile Telecommunications System (UMTS); Selection procedures for the choice of radio transmission technologies of the UMTS

[88] 3G TR 22.925. Technical Specification Group Services and System Aspects; Service aspects; Quality of Service and Network Performance

[89] 3G TR 23.907. Technical Specification Group Services and System Aspects; QoS Concept

[90] 3GPP TR 23.922. Technical Specification Group Services and System Aspects; Architecture for an All IP network

[91] 3GPP TR 25.853. Technical Specification Group (TSG) RAN; delay Budget within the Access Stratum (Release 1999)

[92] 3GPP TR 25.933. Technical Specification Group (TSG) RAN; IP Transport in UTRAN Work Technical Report

[93] 3G TR 25.934. Technical Specification Group Radio Access Network; QoS optimisation for AAL type 2 connections over Iub and Iur interfaces (Release 4)

[94] ETSI TS 125.401. Universal Mobile Telecommunications System (UMTS); UTRAN Overall Description (Release 1999)

[95] ETSI TS 125.413. Universal Mobile Telecommunications System (UMTS); UTRAN Iu Interface RANAP Signalling (Release 1999)

[96] ETSI TS 125.420. Universal Mobile Telecommunications System (UMTS); UTRAN Iur Interface General Aspects and Principles (Release 1999)

[97] ETSI TS 126.101. Universal Mobile Telecommunications System (UMTS); Mandatory Speech Codec speech processing functions; AMR Speech Codec Frame Structure (R99)

[98] 3GPP TS 21.203. Technical Specification Group Services and System Aspects; 3rd Generation mobile system Release 5 specifications (Release 5)

[99] 3G TS 22.105. Technical Specification Group Services and System Aspects; Service aspects; Services and Service Capabilities (Release 5)

[100] 3G TS 25.331. Technical Specification Group Radio Access Network; Radio Resource Control (RRC) Protocol specification

[101] 3GPP TS 23.107. Technical Specification Group Services and System Aspects; QoS Concept and Architecture (Release 5)

[102] 3GPP TS 23.207. Technical Specification Group Services and System Aspects; End-to-End QoS Concept and Architecture (Release 5)

[103] 3G TS 25.301. Technical Specification Group Radio Access Network; Radio Interface Protocol Architecture (Release 1999)

[104] 3G TS 25.302. Technical Specification Group Radio Access Network; Services provided by the Physical Layer (Release 1999)

[105] 3G TS 25.321. Technical Specification Group Radio Access Network; MAC protocol specification (Release 1999)

[106] 3G TS 25.322. Technical Specification Group Radio Access Network; RLC Protocol Specification (Release 1999)

[107] 3G TS 25.402. Technical Specification Group Radio Access Network; Synchronisation in UTRAN Stage 2 (Release 1999)

[108] 3GPP 25.426. Technical Specification Group Radio Access Network; UTRAN Iur and Iub interface data transport & transport signalling for DCH data streams (Release 5)

[109] 3G TS 25.427. Technical Specification Group Radio Access Network; UTRAN Iub/Iur Interface User Plane Protocol for DCH Data Streams (Release 1999)

[110] 3G TS 26.071. Technical Specification Group Services and System Aspects; Mandatory Speech Codec speech processing functions ; AMR Speech Codec; General Description

[111] 3G TS 26.092. Technical Specification Group Services and System Aspects; Mandatory Speech Codec speech processing functions; AMR Speech Codec; Comfort noise aspects

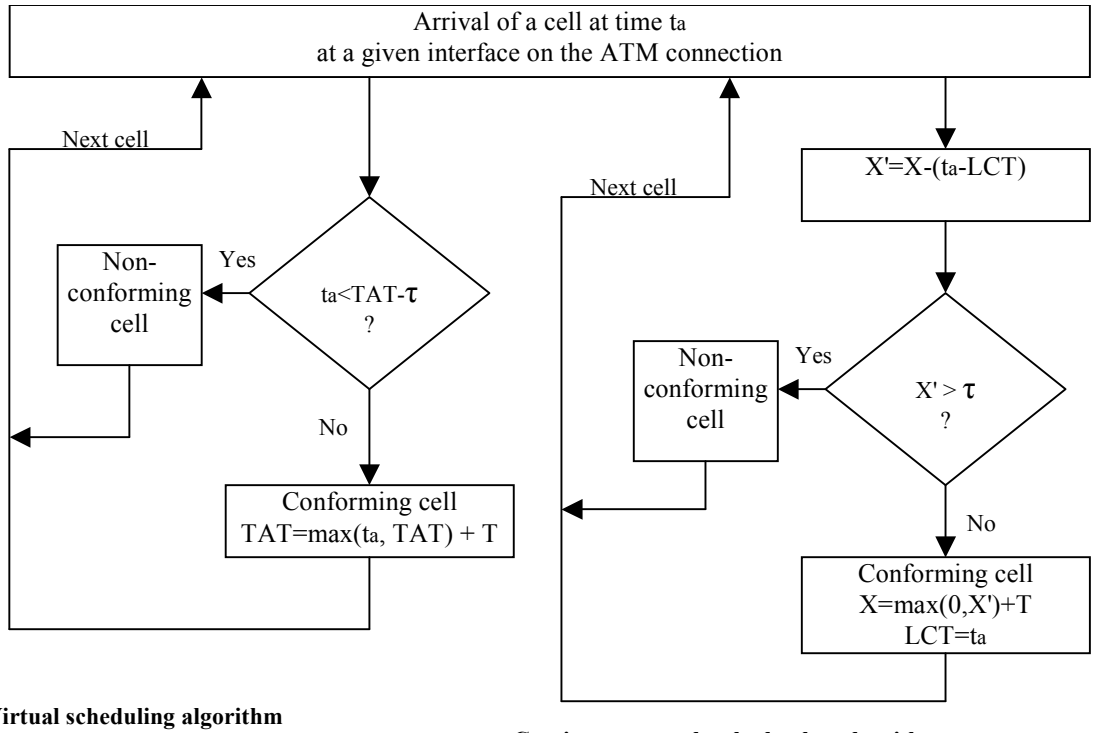
[112] 3GPP TS 26.234. Technical Specification Group Services and System Aspects; Transparent end-to-end packet switched streaming service (PSS); Protocols and codecs (Release 5)

Annexes

Annexe A

A.1 *Generic Cell Rate Algorithm* (GCRA) pour le débit PCR

La figure ci-dessous représente l'algorithme de conformité des cellules ATM utilisé pour le débit crête PCR [ITU-T I.371].



Virtual scheduling algorithm

TAT : Theoretical Arrival Time
ta: Time of arrival of a cell to the given interface

At the time of arrival ta of the first cell of the connection to cross the given interface, TAT=ta

τ : permitted tolerance for the given PCR value

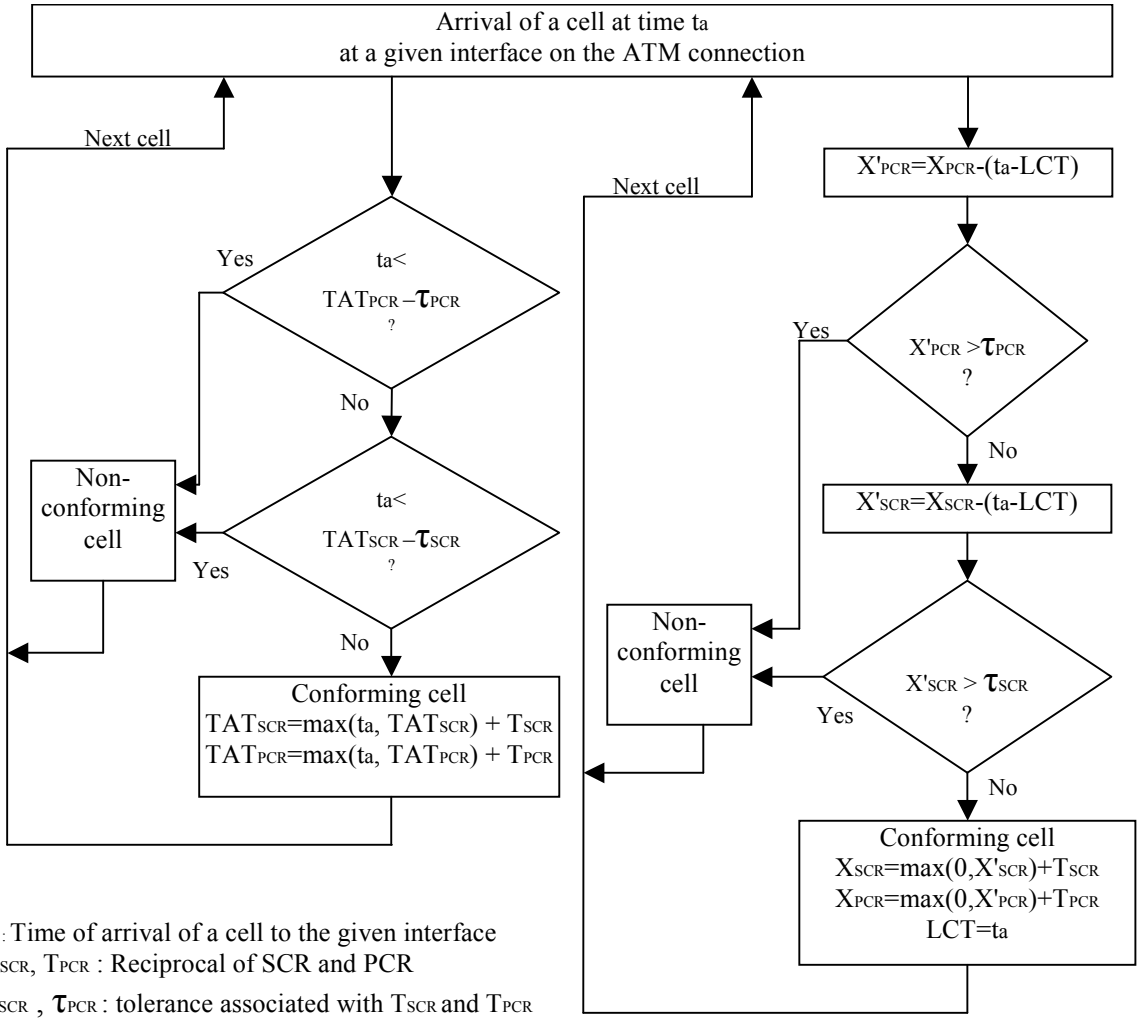
Continuous-state leacky bucket algorithm

X: Value of the leacky Bucket counter
X': Auxiliary variable
LCT: Last Conformance Time

At the time of arrival ta of the first cell of the connection to cross the given interface, X=0 and LCT=ta

A.2 GCRA pour le débit SCR

La figure ci-dessous représente l'algorithme de conformité des cellules ATM utilisé pour le débit soutenu SCR [ITU-T I.371].



ta : Time of arrival of a cell to the given interface
TSCR, TPCR : Reciprocal of SCR and PCR
τSCR , τPCR : tolerance associated with TSCR and TPCR

Virtual scheduling algorithm

TATSCR, TATPCR : Theoretical Arrival Times
ta: Time of arrival of a cell to the given interface

At the time of arrival ta of the first cell of the connection to cross the given interface, TATSCR = TATPCR =ta

Continuous-state leacky bucket algorithm

XSCR, XPCR : Values of the Leacky Bucket counters
X'SCR X'PCR : Auxiliary variables
LCT: Last Conformance Time

At the time of arrival ta of the first cell of the connection to cross the given interface, XSCR = xPCR = 0 and LCT=ta

Annexe B

B.1 Les *Bearer Service*

Les paramètres des services support (*Bearer Service*) utilisés pour transporter les flux sur l'interface radio sont donnés ci-dessous:

RB1:

Higher layer	RAB/Signalling RB		**RAB**
RLC	Logical channel type		DTCH
	RLC mode		UM
	Payload sizes, bit		320
	Max data rate, bps		32000
	AMD PDU header, bit		8
MAC	MAC header, bit		0
	MAC multiplexing		N/A
Layer 1	TrCH type		DCH
	TB sizes, bit		328
	TFS	TF0, bits	0x328
		TF1, bits	1x328
		TF2, bits	2x328 (alt. N/A)
	TTI, ms		20 (alt. 10)
	Coding type		TC (alt. CC 1/3)

RB2:

Higher layer	RAB/Signalling RB		**RAB**
RLC	Logical channel type		DTCH
	RLC mode		AM
	Payload sizes, bit		320
	Max data rate, bps		64000
	AMD PDU header, bit		16
MAC	MAC header, bit		0
	MAC multiplexing		N/A
Layer 1	TrCH type		DCH
	TB sizes, bit		336
	TFS	TF0, bits	0x336
		TF1, bits	1x336
		TF2, bits	2x336
		TF3, bits	3x336
		TF4, bits	4x336
	TTI, ms		20

RB3:

Higher Layer	RAB/Signalling RB		**RAB**
RLC	Logical channel type		DTCH
	RLC mode		AM
	Payload sizes, bit		320
	Max data rate, bps		16000
	AMD PDU header, bit		16
MAC	MAC header, bit		0
	MAC multiplexing		N/A
Layer 1	TrCH type		DCH
	TB sizes, bit		336
	TFS	TF0, bits	0x336
		TF1, bits	1x336
		TF2, bits	2x336
	TTI, ms		40

RB4:

Higher Layer	RAB/Signalling RB		**RAB subflow #1**	**RAB subflow #2**	**RAB subflow #3**
RLC	Logical channel type		DTCH		
	RLC mode		TM	TM	TM
	Payload sizes, bit		39, 81 (alt. 0, 39, 81)	103	60
	Max data rate, bps		12200		
	TrD PDU header, bit		0		
MAC	MAC header, bit		0		
	MAC multiplexing		N/A		
Layer 1	TrCH type		DCH	DCH	DCH
	TB sizes, bit		39, 81 (alt. 0, 39, 81)	103	60
	TFS	TF0, bits	0x81(alt. 1x0) (note)	0x103	0x60
		TF1, bits	1x39	1x103	1x60
		TF2, bits	1x81	N/A	N/A
	TTI, ms		20	20	20

RB5:

Higher Layer	RAB/Signalling RB		**RAB**
RLC	Logical channel type		DTCH
	RLC mode		UM
	Payload sizes, bit		320
	Max data rate, bps		128000
	TrD PDU header, bit		8
MAC	MAC header, bit		0
	MAC multiplexing		N/A
Layer 1	TrCH type		DCH
	TB sizes, bit		328
	TFS	TF0, bits	0x328
		TF1, bits	1x328
		TF2, bits	2x328
		TF3, bits	4x328
		TF4, bits	8x328
		TF5, bits	16x328
	TTI, ms		40

B.2 *Mapping* entre les applications et les BS

Le tableau ci-dessous représente le *mapping* entre les applications que nous avons utilisées et les *Bearer Service*.

Service	Class	RLC mode	Radio bearers DL
Visio conference	Conversational	UM	RB5
WEB browsing	Interactive	AM	RB2
FTP	Background	AM	RB2
VoIP	Conversational	UM	RB1
AMR	Conversational	TM	RB4
Video streaming	Streaming	UM	RB5
SMS	Background	AM	RB3
E-mail	Background	AM	RB3

B.3 Les en-têtes des protocoles

Le tableau ci-dessous représente les tailles des en-têtes des protocoles utilisés dans le modèle de simulation du chapitre 7.

Protocole	*Overhead* (Octet)		
	Minimum	Maximum	Typique
RTP	12	12 +	12
UDP	8	8	8
IP v4	20	60	20
IP v6	40	Nx40 (*[1])	40
cUDP	0	2	2
cRTP (*[3])	2	4	2
c(UDP/IPv4)	2	-	6
c(UDP/IPv6)	2	-	7
c(RTP/UDP/IPv4) (*[4])	2	4	4
cIPv4	2	-	4
cIPv6	2	-	5
PPP (without framing)	1(*[5])	2 (*[5])	1(*[5])
PPPmux	1(*[2])	4(*[2])	2(*[2])
PPP-ML-MC	3	5	3
L2TP	6	14 or plus	6
HDLC	4	8	4
AAL5	8	8+47	8
AAL2 (paquet CPS)	3	3	3
ATM (avec AAL2)	6: (5 ATM + 1 STF)	6	6
ATM	5	5	5

(*[1]) N est le nombre d'options dans l'en-tête IPv6.
(*[2]) C'est l'*overhead* d'une seule sous-trame PPPmux. Toutes les sous-trames sout agrégées dans une seule trame PPP à laquelle un en-tête est ajouté (1-2 octets, typiquement 1 octet).
(*[3]) L'en-tête RTP (*Real-Time Protocol*) est ajouté avec l'en-tête UDP pour les paquest des applications VoIP.
(*[4]) C'est une méthode de compression pour l'ensemble des en-têtes RTP/UDP/IPv4. Elle est plus efficace que la compression de chaque en-tête à part (voir RFC 2508).
(*[5]) Quand PPP est utilisé au dessus de l'AAL2, deux octets de CRC sont ajoutés.
N.B. : Les valeurs données dans la colonne "valeur typique" seront considérées dans nos simulations.

Annexe C

Les fichiers créés dans le simulateur sont de trois types : *fichier.cc* pour le code en *C++*, *fichier.h* pour le fichier header, et *fichier.ned* pour les fichiers contenant la description du modèle de réseau en langage NED. Les fichiers "*fichier.ini*" contiennent les paramètres d'initialisation.

Dans cette annexe, nous n'allons pas présenter tous les fichiers que nous avons développés parce que le code source est trop long (quelques milliers de lignes de code). Nous allons se contenter de quelques exemples de code source pour donner une idée du travail réalisé dans le cadre du développement.

L'annexe C.1 présente un exemple du code source d'un générateur et d'un récepteur de voix AMR. L'annexe C.2 présente le code source du multiplexeur CPS de la couche AAL2.

C.1 Générateur et récepteur AMR

Fichier amr.h

```
#ifndef __AMR_H
#define __AMR_H
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "omnetpp.h"
#include "csig.h"
#include "cdataunit.h"

struct VoiceCall;

class AMRManagement : public cSimpleModule
{
Module_Class_Members(AMRManagement,cSimpleModule,0) // uses handleMessage()
virtual void initialize();
virtual void finish();
virtual void handleMessage(cMessage *msg);
virtual void start_call(cSig *msg);

private:
int number_of_users;
int call_number;
long arrived_calls;
long total_call_number;
cStdDev stat_calls;
cOutVector vector_calls;
double lambda;
double mean_call_ia_time;
double call_ia_time;
};

class AMRGenerator : public cSimpleModule
{
Module_Class_Members(AMRGenerator,cSimpleModule,0) // uses handleMessage()
virtual void initialize();
virtual void handleMessage(cMessage *msg);
virtual void finish();
virtual void CreateCall(cSig *msg);
```

```
virtual void ComputeCall(VoiceCall *CallInfo);
virtual void CreatePeriod(cSig *msg);
virtual void ComputePeriod(VoiceCall *CallInfo);
virtual void ComputeTalkspurt(VoiceCall *CallInfo);
virtual void ComputeSilence(VoiceCall *CallInfo);
virtual void CreatePacket(cSig *msg);
virtual void SendPacket(cSig *msg);

private:
int number_of_users;
int call_index;
double tot_R;
};

struct VoiceCall
{
long call_index;
double call_duration;
double period_duration;
int talk_or_silence;
long max_period_index;
long packet_index;
long sig_packet_index;
};

//================ Sink ================

class AMR_Sink : public cSimpleModule
{
Module_Class_Members(AMR_Sink,cSimpleModule,0)
virtual void initialize();
virtual void handleMessage(cMessage *msg);
virtual void finish();

private:
double PTD;
double PDV;
double all_last_ptd;
double all_pdv;
double last_PTD;
cOutVector vector_PTD;
cOutVector vector_PDV;
double all_ptd;
cStdDev stat_all_ptd;
cOutVector vector_all_ptd;
cOutVector vector_results;
cOutVector vector_distribution;
void statistics(cDataUnit *msg);
void all_pkts(cDataUnit *msg);
double max;
double before_max;
double table[6001];
double distrib[6001];
double counter;
};

#endif
```

Fichier amr.cc

```
#include "amr.h"
Define_Module( AMRManagement )

void AMRManagement::initialize()
{
number_of_users = par("number_of_users");
call_number = 0;
arrived_calls = 0;
total_call_number = 0;
stat_calls.setName("calls");
vector_calls.setName("calls");

if (number_of_users > 0)
      {
      ev << "Nb of AMR users: " << number_of_users << endl;
      ev << "--------------------------------------" << endl;
      call_ia_time = (double)uniform(0,max_call_ia_time);
      cSig *msg = new cSig("NewCall",NEWCALL);
      scheduleAt(simTime()+traffic_generation_start_time+call_ia_time,msg);
      }
}

void AMRManagement::handleMessage(cMessage *msg)
{
      switch ( msg->kind() )
      {
      case NEWCALL:
            start_call((cSig *)msg);
            break;

      default:
            break;
      }
}

void AMRManagement::start_call(cSig *msg)
{
call_number++;
cSig *tCall = new cSig ("Call",CALL);
send(tCall,"out");

if (call_number < number_of_users)
      {
      call_ia_time = (double)uniform(0,max_call_ia_time);
      scheduleAt(simTime()+call_ia_time,msg);
      }
else
      {
      delete msg;
      }
}

Define_Module( AMRGenerator )

void AMRGenerator::initialize()
{
number_of_users = par("number_of_users");
```

```
call_index = 0;
tot_R = 0;
}

void AMRGenerator::handleMessage(cMessage *msg)
{
switch (msg->kind())
      {
      case CALL:
            CreateCall((cSig *)msg);
            break;
      case PERIOD:
            CreatePeriod((cSig *)msg);
            break;
      case PACKET:
            CreatePacket((cSig *)msg);
            break;
      };
}

void AMRGenerator::finish()
{
double interval = 1000*(collect_statistics_stop_time -
collect_statistics_start_time);
double av_br = (double) 8*tot_R/(double)interval;
if (av_br == 0)
      ev << "No AMR traffic" << endl;
else
      {
      ev << "Average AMR BitRate: " << av_br << " Kbps" << endl;
      }
ev << "--------------------------------------" << endl;
}

void AMRGenerator::CreateCall(cSig *tCall)
{
call_index++;
delete tCall;
struct VoiceCall *CallInfo = new VoiceCall;
ComputeCall(CallInfo);
cSig *tPeriod = new cSig("Period",PERIOD);
tPeriod->addPar("info") = (void *) CallInfo;
scheduleAt(simTime(),tPeriod);
}

void AMRGenerator::ComputeCall(VoiceCall *CallInfo)
{
CallInfo->call_index = call_index;
double call_duration = voice_call_duration;
CallInfo->call_duration = call_duration;
CallInfo->talk_or_silence = intrand(2);
CallInfo->packet_index = 1;
CallInfo->max_period_index = 0;
CallInfo->period_duration = 0;
CallInfo->sig_packet_index = 0;
}

void AMRGenerator::CreatePeriod(cSig *tPeriod)
{
```

```
VoiceCall *CallInfo =(VoiceCall *)(void *)tPeriod->par("info");
ComputePeriod(CallInfo);
cSig *tPacket = new cSig("Packet",PACKET);
tPacket->addPar("info") = (void *) CallInfo;
scheduleAt(simTime(),tPacket);
if (simTime() + CallInfo->period_duration < CallInfo->call_duration)
      {
      scheduleAt(simTime() + CallInfo->period_duration,tPeriod);
      }
else
      {
      delete tPeriod;
      }
}

void AMRGenerator::ComputePeriod(VoiceCall* CallInfo)
{
switch (CallInfo->talk_or_silence)
      {
      case SILENCE:
      ComputeTalkspurt(CallInfo);
      break;
      case TALK:
      ComputeSilence(CallInfo);
      break;
      };
}

void AMRGenerator::ComputeTalkspurt(VoiceCall *CallInfo)
{
CallInfo->packet_index = 1;
CallInfo->talk_or_silence = TALK;
double talkspurt_duration = exponential(mean_talkspurt_duration);
if (simTime() + talkspurt_duration > CallInfo->call_duration)
      talkspurt_duration = CallInfo->call_duration - simTime();
long num_talkspurt_packets = (long) ceil(talkspurt_duration /
packet_ia_time);
CallInfo->period_duration = num_talkspurt_packets*packet_ia_time;

CallInfo->max_period_index = num_talkspurt_packets;
}

void AMRGenerator::ComputeSilence(VoiceCall* CallInfo)
{
CallInfo->packet_index = 1;
CallInfo->talk_or_silence = SILENCE;
double silence_duration = exponential(mean_silence_duration);
if (simTime() + silence_duration > CallInfo->call_duration)
      silence_duration = CallInfo->call_duration - simTime();
long num_silence_packets = (long) ceil(silence_duration / SID_ia_time);
CallInfo->period_duration = SID_ia_time*num_silence_packets;
CallInfo->max_period_index = num_silence_packets;
}

void AMRGenerator::CreatePacket(cSig *tPacket)
{
VoiceCall *CallInfo = (VoiceCall *)(void *)tPacket->par("info");
cSig *msg = new cSig("");
msg->addPar("info") = (void *) CallInfo;
```

```
SendPacket(msg);
++CallInfo->packet_index;
++CallInfo->sig_packet_index;
if (CallInfo->sig_packet_index == 15)
            CallInfo->sig_packet_index = 0;
if (CallInfo->packet_index <= CallInfo->max_period_index)
      {
      switch (CallInfo->talk_or_silence)
            {
            case TALK: scheduleAt(simTime() +
packet_ia_time,tPacket);break;
            case SILENCE: scheduleAt(simTime() +
SID_ia_time,tPacket);break;
            }
      }
else
      {
      delete tPacket;
      switch (CallInfo->talk_or_silence)
            {
            case TALK:
                  if (simTime() + packet_ia_time >= CallInfo-
>call_duration)
                        delete CallInfo;
                  break;
            case SILENCE:
                  if (simTime() + SID_ia_time >= CallInfo->call_duration)
                        delete CallInfo;
                  break;
            }
      }
}

void AMRGenerator::SendPacket(cSig *msg)
{
VoiceCall *CallInfo = (VoiceCall *)(void *)msg->par("info");
delete msg;

cDataUnit *pkt = new cDataUnit("AMR",AMR);

switch (CallInfo->talk_or_silence)
      {
      case TALK:
      pkt->setLength(bytes_per_talkspurt_packet + FP_H_V + FP_Tail);
      break;
      case SILENCE:
      pkt->setLength(bytes_per_silence_packet + FP_H_SID + FP_Tail);
      break;
      };
if (CallInfo->sig_packet_index == 0)
      pkt->addLength(sig_msg_length);
PACKETHEADER header;
header.initialize();
header.nb_of_pkts = 0;
header.Type = AMR;
header.TOS = EF;
header.Flow_ID = CallInfo->call_index;
header.Length = pkt->length();
header.MORE = NO;
```

```
header.FP_PDU_size = pkt->length();
header.timestamp = simTime();
pkt->setHeader(header);
pkt->setTimestamp();
ev << "Time : " << simTime() << endl;
ev << "Ev : Emission of packet from AMR generator" << endl;
ev << "pkt length: " << pkt->length() << endl;
ev << "====================================" << endl;
if ( (simTime() >= collect_statistics_start_time) && (simTime() <=
collect_statistics_stop_time) )
      {
      double len = pkt->length();
      tot_R += len;
      }
send(pkt,"out");
}

//=============== Sink ================

Define_Module( AMR_Sink )

void AMR_Sink::initialize()
{
last_PTD = 0;
PTD = 0;
PDV = 0;
vector_PTD.setName("PTD for AMR");
vector_PDV.setName("PDV for AMR");
all_ptd =0;
all_last_ptd = 0;
all_pdv = 0;
stat_all_ptd.setName("PTD for all AMR users");
vector_all_ptd.setName("PTD for all pkts");
vector_distribution.setName("probability distribution for AMR delay");
vector_results.setName("AMR performance");
max = 0;
before_max = 0;
counter = 0;
for (int i=0; i<=6000; i++)
      {
      table[i] = 0;
      distrib[i] = 0;
      }
}

void AMR_Sink::handleMessage(cMessage *msg)
{
cDataUnit *pkt = (cDataUnit *)msg;
int user = pkt->header().Flow_ID;
all_pkts(pkt);
ev << "Time : " << simTime() << endl;
ev << "Ev : Reception of packet in AMR sink" << endl;
ev << "pkt length : " << pkt->length() << endl;
ev << "timestamp : " << pkt->header().timestamp << endl;
ev << "====================================" << endl;
delete msg;
}

void AMR_Sink::all_pkts(cDataUnit *msg)
```

```
{
if ( (msg->header().timestamp >= collect_statistics_start_time) && (msg-
>header().timestamp <= collect_statistics_stop_time) )
            {
            all_ptd = simTime() - msg->header().timestamp;
            all_ptd = 1000*all_ptd;
            stat_all_ptd.collect(all_ptd);

            if (all_last_ptd == 0)
                  all_last_ptd = all_ptd;
            double pdv = all_ptd - all_last_ptd;
            all_last_ptd = all_ptd;
            if (pdv < 0)
                  pdv = - pdv;
            if (pdv > all_pdv)
                  all_pdv = pdv;
            ++counter;
            #ifdef TRACE
            ev << "counter= " << counter << endl;
            ev << "ptd= " << all_ptd << endl;
            #endif
            double d = 100*all_ptd;
            int i = (int) ceil(d);
            #ifdef TRACE
            ev << "d= " << d << endl;
            ev << "i= " << i << endl;
            #endif
            if (i>6001)
                  ++table[6000];
            else
                  {
                  ++table[i-1];
                  #ifdef TRACE
                  ev << "i" << i-1 << "= " << table[i-1] << endl;
                  #endif
                  }
            }
}

void AMR_Sink::statistics(cDataUnit *msg)
{
PTD = simTime() - msg->header().timestamp;
PTD = 1000*PTD;
if (last_PTD == 0)
      last_PTD = PTD;
PDV = PTD - last_PTD;
last_PTD = PTD;
vector_PTD.record(PTD);
vector_PDV.record(PDV);
}

void AMR_Sink::finish()
{
double d999 = 0;
double d95 = 0;
double mean = 0;
double max = 0;
double stddev = 0;
```

```
if (counter != 0)
{
ev << "AMR: " << endl;
int l95 = (int) ceil(0.95*counter);
int l999 = (int) ceil(0.999*counter);
double s = 0;
for (int i=0; i <= 6000; i++)
      {
      s += table[i];
      if (s >= l95)
            {
            d95 = (double) (i+1)/100;
            #ifdef RESULTS
            vector_results.record(d95);
            #endif
            if (d95 > 60)
                  ev << "very high delay !!! >> 60ms" << endl;
            else
                  ev << "95-percentile delay = " << d95 << " ms" << endl;
            break;
            }
      }
s = 0;
for (int i=0; i <= 6000; i++)
      {
      s += table[i];
      if (s >= l999)
            {
            d999 = (double) (i+1)/100;
            #ifdef RESULTS
            vector_results.record(d999);
            #endif
            if (d999 > 60)
                  ev << "very high delay !!! >> 60ms  " << endl;
            else
                  ev << "99.9-percentile delay = " << d999 << " ms" <<
endl;
            break;
            }
      }

#ifdef DISTRIBUTION
distrib[6000] = table[6000];
for (int j=1; j<=6000; j++)
      {
      int i = 6000-j;
      distrib[i] = table[i] + distrib[i+1];
      }
for (int i=0; i<=6000;i++)
      {
      vector_distribution.record(distrib[i]);
      }
#endif

#ifdef RESULTS
vector_results.record(stat_all_ptd.max());
vector_results.record(stat_all_ptd.mean());
vector_results.record(stat_all_ptd.stddev());
vector_results.record(all_pdv);
```

```
#endif

mean = stat_all_ptd.mean();
max = stat_all_ptd.max();
stddev = stat_all_ptd.stddev();
ev << "min: " << stat_all_ptd.min() << endl;
ev << "max: " << stat_all_ptd.max() << endl;
ev << "mean: " << stat_all_ptd.mean() << endl;
ev << "stddev: " << stat_all_ptd.stddev() << endl;
ev << "max packet-to-packet delay variation: " << all_pdv << endl;
ev << "--------------------------------------" << endl;
}
char *file_name = "/infres/ir60/rhd/makke/simulation_results.txt";
FILE *fichier;
fichier = fopen(file_name,"a");
fprintf(fichier,"%f\t%f\t%f\t%f\t%f\t",mean,max,stddev,d95,d999);
fclose(fichier);
}
```

Fichier amr.ned

```
simple AMRManagement
 parameters:
  number_of_users;
 gates:
  out: out;
endsimple

simple AMRGenerator
 parameters:
  number_of_users;

 gates:
  in: in;
  out: out;
endsimple

module AMR_Generator
 parameters:
  Number_Of_Users;
 gates:
  out: out;

 submodules:
  AMRManagement : AMRManagement
   parameters:
            number_of_users = Number_Of_Users;
  AMRGenerator : AMRGenerator
   parameters:
            number_of_users = Number_Of_Users;
 connections:
   AMRManagement.out --> AMRGenerator.in;
   AMRGenerator.out --> out;

endmodule

simple AMR_Sink
 parameters:
```

```
 gates:
       in: in;
endsimple
```

C.2 La couche CPS de l'AAL2 avec les ordonnanceurs

Fichier cps_atm.h

```
#ifndef __CPS_ATM_H
#define __CPS_ATM_H

#include <stdio.h>
#include <string.h>
#include <time.h>
#include "omnetpp.h"
#include "cminicell.h"
#include "ccell.h"
#include "csig.h"

class CPS_Insertion : public cSimpleModule
{
Module_Class_Members(CPS_Insertion,cSimpleModule,0)
virtual void initialize();
virtual void handleMessage(cMessage *msg);
virtual void finish();
void CPS_header(cMiniCell *msg);
void queueing(cMiniCell *msg);
void put_fifo(cMiniCell *msg);
void put_rr(cMiniCell *msg);
void put_wrr(cMiniCell *msg);
void put_edf(cMiniCell *msg);
void put_prior(cMiniCell *msg);
void put_dwrr(cMiniCell *msg);
void init_dwrr();
void update_dwrr();
void generate_new_cell();
void packaging();
void get_fifo();
void get_rr();
void get_wrr();
void get_prior();
void get_edf();
void get_dwrr();
void send_cell();
void Maal_send_request();
void obligation();
void timer_cu_expired();
void authorization();

private:
int id;
double Timer_CU;
double BW;
double SCR;
double cell_ia_time;
cSig *new_cell;
cSig *timercu;
cMiniCell *minicell_in_service;
```

```
cCell *cell_in_gare;
cCell *ready_cell;
int lock;
int rest;
int permit;
int timer_expired;
int newpacket;
int cell_in_gare_state;
int state;
int free_place;
cQueue FIFOqueue;
cQueue voice_queue;
cQueue web_queue;
cQueue ftp_queue;
cQueue email_queue;
cQueue RRqueue[248];
double FIFO_size;
cStdDev stat_FIFO_queue;
cStdDev stat_FIFO_size;
cOutVector vect_FIFO_queue;
cOutVector vect_FIFO_size;
cOutVector vector_results;
double voice_size;
double data_size;
cStdDev stat_voice_queue;
cStdDev stat_data_queue;
cMessage *update;
double C;
int nb_zeros;
double vbw[4];
double prob[4];
int dwrr_weight[4];
int dwrr_table[4];
double dwrr_queue_length[4];
double dwrr_queue_size[4];
int dwrr_round;
int dwrr_index;
int RRindex;
int wrr_table[wrr_cycle];
int wrr_index;
double edf_time[4];
};

class CPS_Extraction : public cSimpleModule
{
Module_Class_Members(CPS_Extraction,cSimpleModule,0)
virtual void initialize();
virtual void handleMessage(cMessage *msg);
void start_extraction(cMessage *msg);
void extract(cCell *msg);
void send_minicell();
void CPS_header(cMiniCell *msg);

private:
double service_time;
cCell *cell_in_service;
cSig *end_extraction;
cQueue queue;
cMiniCell *minicell_to_send;
```

```
};

class ATM_Emission : public cSimpleModule
{
Module_Class_Members(ATM_Emission,cSimpleModule,0)
virtual void initialize();
virtual void handleMessage(cMessage *msg);
void emission(cMessage *msg);
void ATM_header(cCell *msg);
void calculate_utilization(cCell *msg);
void finish();

private:
int id;
int sn;
double Speed;
double delta;
double BW;
double T_pcr;
double TAT_pcr;
double taw_pcr;
double SCR[25];
double T_scr[25];
double TAT_scr[25];
double taw_scr[25];
double nb_non_conforming_cells[25];
double cell_emission_time;
cQueue buffer;
cStdDev stat_buffer;
cOutVector vect_buffer;
cStdDev stat_filling;
cCell *under_emission_cell;
cMessage *end_emission;
long nb_of_cells;
double total_cells;
long sum;
long nb_of_full_cells;
cMessage *test_msg;
double cnb;
double last_check;
double max_rate;
double R;
cStdDev stat_rate;
cOutVector vect_rate;
cOutVector vector_results;
};

class ATM_Reception : public cSimpleModule
{
Module_Class_Members(ATM_Reception,cSimpleModule,0)
virtual void initialize();
virtual void handleMessage(cMessage *msg);
};

class SINKENTRY : public cSimpleModule
{
Module_Class_Members(SINKENTRY,cSimpleModule,0)
virtual void initialize();
virtual void handleMessage(cMessage *msg);
```

```
};

class RNCENTRY : public cSimpleModule
{
Module_Class_Members(RNCENTRY,cSimpleModule,0)
virtual void initialize();
virtual void handleMessage(cMessage *msg);
};

#endif
```

Fichier cps_atm.cc

```
#include "cps_atm.h"

Define_Module( CPS_Insertion )


void CPS_Insertion::initialize()
{
id = par("id");
Timer_CU = par("Timer_CU");
Timer_CU = Timer_CU/1000000.0;
BW = par("VC_PCR");
SCR = par("VC_SCR");
if ( (id == 0)||(id == monitored_nodeB) )
{
if (id == 0)
      {
      ev << "HUB --> RNC: Timer-CU = " << Timer_CU*BW/424 << "xT = " <<
1000*Timer_CU << " ms          T = " << 1000000*424/BW << " us" << endl;
      vector_results.setName("Hub buffer size");
      }
else
      {
      ev << "NodeB " << id << " --> HUB: Timer-CU = " << Timer_CU*BW/424 <<
"xT = " << 1000*Timer_CU << " ms          T = " << 1000000*424/BW << " us"
<< endl;
      vector_results.setName("NodeB buffer size");
      }
switch (scheduling)
      {
      case FIFO: ev << "scheduling policy: FIFO" << endl; break;
      case PRIOR: ev << "scheduling policy: PRIOR" << endl; break;
      case WRR:
            ev << "scheduling policy: WRR" << endl;
            ev << "voice weight: " << voice_weight << "/" << wrr_cycle <<
endl;
            ev << "web weight: " << web_weight << "/" << wrr_cycle << endl;
            ev << "cycle: " << wrr_cycle << endl;
             break;
      case EDF:
            ev << "scheduling policy: EDF" << endl;
            ev << "voice deadline: " << 1000*voice_deadline << " ms" <<
endl;
            ev << "web deadline: " << 1000*web_deadline << " ms" << endl;
            break;
      case DWRR:
            ev << "scheduling policy: DWRR" << endl;
```

```
            ev << "voice time constraint: " << 1000*voice_T << " ms" <<
endl;
            ev << "web time constraint: " << 1000*web_T << " ms" << endl;
            ev << "voice PLR: " << voice_PLR << endl;
            ev << "web PLR: " << web_PLR << endl;
            ev << "reference time scale: " << 1000*ref_T << " ms" << endl;
            ev << "cycle: " << dwrr_cycle << endl;
            ev << "update interval: " << 1000*update_interval << " ms" <<
endl;
            break;
      };
}
switch (ATC)
      {
      case DBR:
            cell_ia_time = (double) (8*cell_size/(double)BW);
            if ( (id == 0)||(id == monitored_nodeB) )
                  ev << "ATC: DBR" << endl;
            break;
      case UBR:
            if ( (id == 0)||(id == monitored_nodeB) )
                  ev << "ATC: UBR" << endl;
            break;
      case SBR:
            cell_ia_time = (double) (8*cell_size/(double)BW);
            if ( (id == 0)||(id == monitored_nodeB) )
                  ev << "ATC: SBR" << endl;
            break;
      };
if ( (id == 0)||(id == monitored_nodeB) )
      ev << "****************************************" << endl;
cell_in_gare =  NULL;
ready_cell = NULL;
lock = NO;
free_place = 0;
cell_in_gare_state = IDLE;
minicell_in_service = NULL;
rest = 0;
newpacket = NO;

switch (ATC)
      {
      case DBR:
            permit = NO;
            break;
      case UBR:
            permit = YES;
            break;
      case SBR:
            permit = NO;
            break;
      };

timer_expired = NO;
state = IDLE;
voice_size = 0;
stat_voice_queue.setName("voice queue size");
data_size = 0;
stat_data_queue.setName("data queue size");
```

```
FIFO_size = 0;
stat_FIFO_queue.setName("FIFO queue length");
stat_FIFO_size.setName("FIFO size");
vect_FIFO_queue.setName("FIFO queue length");
vect_FIFO_size.setName("FIFO size");

if (scheduling == WRR)
{
RRindex = 0;
wrr_index = 0;
if (voice_weight >0)
for (int i=wrr_index;i<wrr_index+voice_weight;i++)
      wrr_table[i] = VOICE;
wrr_index = wrr_index + voice_weight;
if (web_weight >0)
for (int i=wrr_index;i<wrr_index+web_weight;i++)
      wrr_table[i] = WEB;
wrr_index = wrr_index + web_weight;
if (ftp_weight>0)
for (int i=wrr_index;i<wrr_index+ftp_weight;i++)
      wrr_table[i] = FTP;
wrr_index = wrr_index + ftp_weight;
if (email_weight>0)
for (int i=wrr_index;i<wrr_index+email_weight;i++)
      wrr_table[i] = EMAIL;
wrr_index = 0;
}
if (scheduling == EDF)
{
for (int i=0; i<4;i++)
      edf_time[i] = 0;
}
if (scheduling == DWRR)
{
C = (double)((47*BW)/(48*53*8.0));
for (int i=0; i<4; i++)
      {
      dwrr_queue_length[i] = 0;
      dwrr_queue_size[i];
      }
dwrr_index = 0;
dwrr_round = 0;
nb_zeros = 0;
update_dwrr();
init_dwrr();
}
timercu = NULL;
switch (ATC)
      {
      case DBR:
            new_cell = new cSig("NewCell",NEWCELL);
            scheduleAt(simTime(),new_cell);
            break;
      case UBR:
            break;
      case SBR:
            new_cell = new cSig("NewCell",NEWCELL);
            scheduleAt(simTime(),new_cell);
            break;
```

```
      };
}

void CPS_Insertion::finish()
{
if ( (id == 0)||(id == monitored_nodeB) )
{
if (id == 0)
      ev << "HUB :" << endl;
else
      ev << "NodeB " << id << " :" << endl;
if (scheduling == FIFO)
      {
      #ifdef RESULTS
      vector_results.record(stat_FIFO_size.max());
      #endif
      }
else
      {
      #ifdef RESULTS
      vector_results.record(stat_voice_queue.max());
      vector_results.record(stat_data_queue.max());
      #endif
      }
ev << "----------------------------------------" << endl;
}
}

void CPS_Insertion::handleMessage(cMessage *msg)
{
#ifdef TRACE
#endif
switch ( msg->kind() )
      {
      case TIMER_CU: timer_cu_expired();break;
      case NEWCELL: Maal_send_request();break;
      case UPDATE: update_dwrr();break;
      default: CPS_header((cMiniCell *)msg);break;
      };
}

void CPS_Insertion::CPS_header(cMiniCell *msg)
{
msg->addLength(CPS_Header);
queueing(msg);
if (state != FULL)
      packaging();
}

void CPS_Insertion::generate_new_cell()
{
#ifdef TRACE
ev << "-----------------------------------" << endl;
ev << "Time: " << simTime() << endl;
ev << "generate new cell..." << endl;
ev << "scheduling Timer-CU at: " << simTime()+Timer_CU << endl;
ev << "-----------------------------------" << endl;
#endif
cCell *cell = new cCell("cell");
```

```
cell->setKind(CELL);
cell->setLength(cell_size);
cell->setType(CELL);
cell->setTimestamp();
CELLHEADER header;
header.initialize();
header.VPI = id;
header.VCI = id;
header.OSF = 0;
header.PAD = cell_load_size;
header.timestamp = simTime();
cell->setHeader(header);
cell_in_gare = cell;
free_place = cell_load_size;
lock = NO;
timer_expired = NO;
cell_in_gare_state = IDLE;
if ( timercu != NULL )
      {
      delete cancelEvent(timercu);
      timercu = NULL;
      }
timercu = new cSig("Timer_CU",TIMER_CU);
scheduleAt(simTime()+Timer_CU,timercu);
}

void CPS_Insertion::Maal_send_request()
{
switch (autho_or_oblig)
      {
      case AUTHORIZATION: authorization();break;
      case OBLIGATION: obligation();break;
      }
scheduleAt(simTime()+cell_ia_time,new_cell);
}


void CPS_Insertion::authorization()
{
CELLHEADER header;
header.initialize();
switch ( state )
      {
      case IDLE:
                  permit = YES;
                  break;
      case PART:
                  permit = YES;
                  break;
      case SEND:
                  ready_cell = cell_in_gare;
                  cell_in_gare = NULL;
                  cell_in_gare_state = IDLE;
                  header = ready_cell->header();
                  header.PAD = free_place;
                  ready_cell->setHeader(header);
                  send_cell();
                  ready_cell = NULL;
                  state = IDLE;
```

```
                  if ( timercu != NULL )
                        {
                        delete cancelEvent(timercu);
                        timercu = NULL;
                        }
                  break;
      case FULL:
                  header = ready_cell->header();
                  header.PAD = 0;
                  ready_cell->setHeader(header);
                  send_cell();
                  ready_cell = NULL;
                  switch (cell_in_gare_state)
                        {
                        case FULL:
                                    state = FULL;
                                    ready_cell = cell_in_gare;
                                    cell_in_gare = NULL;
                                    cell_in_gare_state = IDLE;
                                    break;
                        case IDLE:
                                    state = IDLE;
                                    packaging();
                                    break;
                        case PART:
                                    state = PART;
                                    packaging();
                                    break;
                        case SEND:
                                    state = SEND;
                                    if (locking_or_not == NO_LOCKING)
                                          packaging();
                                    else
                                          {
                                          if ( timer_expired == NO )
                                                packaging();
                                          }
                                    break;
                        }
                  break;
      }
}

void CPS_Insertion::obligation()
{
CELLHEADER header;
header.initialize();
switch (state)
      {
      case IDLE:
                  break;
      case PART:
                  ready_cell = cell_in_gare;
                  cell_in_gare = NULL;
                  cell_in_gare_state = IDLE;
                  header = ready_cell->header();
                  header.PAD = free_place;
                  ready_cell->setHeader(header);
                  send_cell();
```

```
                ready_cell = NULL;
                state = IDLE;
                if ( timercu != NULL )
                      {
                      delete cancelEvent(timercu);
                      timercu = NULL;
                      }
                break;
        case SEND:
                ready_cell = cell_in_gare;
                cell_in_gare = NULL;
                cell_in_gare_state = IDLE;
                header = ready_cell->header();
                header.PAD = free_place;
                ready_cell->setHeader(header);
                send_cell();
                ready_cell= NULL;
                state = IDLE;
                if ( timercu != NULL )
                      {
                      delete cancelEvent(timercu);
                      timercu = NULL;
                      }
                break;
        case FULL:
                header = ready_cell->header();
                header.PAD = 0;
                ready_cell->setHeader(header);
                send_cell();
                ready_cell = NULL;
                switch (cell_in_gare_state)
                      {
                      case FULL:
                                  ready_cell = cell_in_gare;
                                  cell_in_gare = NULL;
                                  cell_in_gare_state = IDLE;
                                  state = FULL;
                                  break;
                      case IDLE:
                                  state = IDLE;
                                  packaging();
                                  break;
                      case PART:
                                  if (timer_expired == YES)
                                        {
                                        timer_expired = NO;
                                        state = SEND;
                                        }
                                  else
                                        state = PART;
                                  packaging();
                                  break;
                      case SEND:
                                  state = SEND;
                                  if (locking_or_not == NO_LOCKING)
                                        packaging();
                                  else
                                        {
                                        if ( timercu->isScheduled() )
```

```
                                                packaging();
                                            }
                                        break;
                                }
                        break;
        }
}


void CPS_Insertion::put_fifo(cMiniCell *msg)
{
double size = (double)msg->length();
FIFO_size += size;
FIFOqueue.insertHead(msg);

#ifdef TRACE
      ev << "fifo length:  " << FIFOqueue.length() << endl;
#endif

if ( (simTime() >= collect_statistics_start_time) && (simTime() <=
collect_statistics_stop_time) )
      {
      stat_FIFO_size.collect(FIFO_size);
      //vect_FIFO_size.record(FIFO_size);
      double len = (double) FIFOqueue.length();
      stat_FIFO_queue.collect(len);
      //vect_FIFO_queue.record(len);
      }
}

void CPS_Insertion::put_rr(cMiniCell *msg)
{
int index = msg->header().link1.CID - 1;
if (index > 247)
      {
      ev << "Warning ! the number of connections is more than 248" << endl;
      ev << "the packets concerning this connection are rejected" << endl;
      delete msg;
      }
else
      RRqueue[index].insertHead(msg);
}

void CPS_Insertion::put_prior(cMiniCell *msg)
{
double size = (double) msg->length();
switch ( msg->kind() )
      {
      case VOICE :
            voice_size += size;
            if ( (simTime() >= collect_statistics_start_time) && (simTime()
<= collect_statistics_stop_time) )
                  stat_voice_queue.collect(voice_size);
            voice_queue.insertHead(msg);
            break;
      case WEB :
            data_size += size;
            if ( (simTime() >= collect_statistics_start_time) && (simTime()
<= collect_statistics_stop_time) )
```

```
                  stat_data_queue.collect(data_size);
            web_queue.insertHead(msg);
            break;
      case FTP : ftp_queue.insertHead(msg);break;
      case EMAIL : email_queue.insertHead(msg);break;
      }
}

void CPS_Insertion::put_wrr(cMiniCell *msg)
{
double size = (double)msg->length();
switch( msg->kind() )
      {
      case VOICE :
            voice_size += size;
            if ( (simTime() >= collect_statistics_start_time) && (simTime()
<= collect_statistics_stop_time) )
                  stat_voice_queue.collect(voice_size);
            voice_queue.insertHead(msg);
            break;
      case WEB :
            data_size += size;
            if ( (simTime() >= collect_statistics_start_time) && (simTime()
<= collect_statistics_stop_time) )
                  stat_data_queue.collect(data_size);
            web_queue.insertHead(msg);
            break;
      case FTP : ftp_queue.insertHead(msg);break;
      case EMAIL : email_queue.insertHead(msg);break;
      }
}

void CPS_Insertion::init_dwrr()
{
nb_zeros = 0;
dwrr_index = 0;
dwrr_round = 0;
for (int i=0; i<4; i++)
      {
      dwrr_table[i] = dwrr_weight[i];
      dwrr_round += dwrr_table[i];
      if (dwrr_table[i] == 0)
            {
            dwrr_round += 1;
            nb_zeros++;
            }
      }
}

void CPS_Insertion::update_dwrr()
{
switch (bit_or_packet)
      {
      case BIT:
            if (with_PLR == YES)
                  {
                  vbw[0] = ((1 -
voice_PLR)*ref_T*dwrr_queue_size[0])/voice_T;
                  vbw[1] = ((1 - web_PLR)*ref_T*dwrr_queue_size[1])/web_T;
```

```
                  vbw[2] = ((1 - ftp_PLR)*ref_T*dwrr_queue_size[2])/ftp_T;
                  vbw[3] = ((1 -
email_PLR)*ref_T*dwrr_queue_size[3])/email_T;
                  }
            else
                  {
                  vbw[0] = (ref_T*dwrr_queue_size[0])/voice_T;
                  vbw[1] = (ref_T*dwrr_queue_size[1])/web_T;
                  vbw[2] = (ref_T*dwrr_queue_size[2])/ftp_T;
                  vbw[3] = (ref_T*dwrr_queue_size[3])/email_T;
                  }
      break;
      case PACKET:
            if (with_PLR == YES)
                  {
                  vbw[0] = ((1 -
voice_PLR)*ref_T*dwrr_queue_length[0])/voice_T;
                  vbw[1] = ((1 -
web_PLR)*ref_T*dwrr_queue_length[1])/web_T;
                  vbw[2] = ((1 -
ftp_PLR)*ref_T*dwrr_queue_length[2])/ftp_T;
                  vbw[3] = ((1 -
email_PLR)*ref_T*dwrr_queue_length[3])/email_T;
                  }
            else
                  {
                  vbw[0] = (ref_T*dwrr_queue_length[0])/voice_T;
                  vbw[1] = (ref_T*dwrr_queue_length[1])/web_T;
                  vbw[2] = (ref_T*dwrr_queue_length[2])/ftp_T;
                  vbw[3] = (ref_T*dwrr_queue_length[3])/email_T;
                  }
      break;
      };


double total = 0;
for (int i=0; i<4; i++)
      total += vbw[i];
ev << "time : " << simTime() << endl;
ev << "total = " << total << endl;

for (int i=0; i<4; i++)
      {
      if (total == 0)
            prob[i] = 0.25;
      else
            prob[i] = (double)(vbw[i]/total);

      if (prob[i] == 0)
            dwrr_weight[i] = 0;
      else
            {
            dwrr_weight[i] = (int)around(prob[i]*dwrr_cycle);
            if (dwrr_weight[i] == 0)
                  dwrr_weight[i] = 1;
            }
      ev << "dwrr_queue[" << i << "] = " << dwrr_queue_length[i] << "
packets ... virtual bw[" << i << "] = " << vbw[i] << " ... prob[" << i <<
```

```
"] = " << prob[i] << " ... weight[" << i << "] = " << dwrr_weight[i] <<
endl;
      dwrr_queue_length[i] = 0;
      dwrr_queue_size[i] = 0;
      }
ev << "=========================================" << endl;
update = NULL;
update = new cMessage("update",UPDATE);
scheduleAt(simTime()+update_interval,update);
}

void CPS_Insertion::put_dwrr(cMiniCell *msg)
{
int index;
double size = (double)msg->length();
switch( msg->kind() )
      {
      case VOICE :
            voice_size += size;
            if ( (simTime() >= collect_statistics_start_time) && (simTime()
<= collect_statistics_stop_time) )
                  stat_voice_queue.collect(voice_size);
            voice_queue.insertHead(msg);
            ++dwrr_queue_length[0];
            dwrr_queue_size[0] += 8*size;
            break;
      case WEB :
            data_size += size;
            if ( (simTime() >= collect_statistics_start_time) && (simTime()
<= collect_statistics_stop_time) )
                  stat_data_queue.collect(data_size);
            web_queue.insertHead(msg);
            ++dwrr_queue_length[1];
            dwrr_queue_size[1] += 8*size;
            break;
      case FTP :
            ftp_queue.insertHead(msg);
            ++dwrr_queue_length[2];
            dwrr_queue_size[2] += 8*size;
            break;
      case EMAIL :
            email_queue.insertHead(msg);
            ++dwrr_queue_length[3];
            dwrr_queue_size[3] += 8*size;
            break;
      }
}

void CPS_Insertion::put_edf(cMiniCell *msg)
{
double size = (double)msg->length();
switch( msg->kind() )
      {
      case VOICE :
            if ( edf_time[0] == 0 )
                  edf_time[0] = msg->timestamp()+voice_deadline;
            voice_size += size;
            if ( (simTime() >= collect_statistics_start_time) && (simTime()
<= collect_statistics_stop_time) )
```

```
                  stat_voice_queue.collect(voice_size);
            voice_queue.insertHead(msg);
            #ifdef TRACE
            ev << "**************************************" << endl;
            ev << simTime() << " :put packet in voice queue" << endl;
            ev << "**************************************" << endl;
            #endif
            break;
      case WEB :
            #ifdef TRACE
            ev << "======================================" << endl;
            ev << simTime() << " :put packet in web queue" << endl;
            #endif

            if ( edf_time[1] == 0)
                  {
                  edf_time[1] = msg->timestamp()+web_deadline;
                  #ifdef TRACE
                  ev << "deadline for the new packet: " << edf_time[1] <<
endl;
                  #endif
                  }
            #ifdef TRACE
            ev << "======================================" << endl;
            #endif
            data_size += size;
            if ( (simTime() >= collect_statistics_start_time) && (simTime()
<= collect_statistics_stop_time) )
                  stat_data_queue.collect(data_size);
            web_queue.insertHead(msg);

            break;
      case FTP :
            if ( edf_time[2] == 0 )
                  edf_time[2] = msg->timestamp()+ftp_deadline;
            ftp_queue.insertHead(msg);
            #ifdef TRACE
            ev << simTime() << " :put packet in ftp queue" << endl;
            #endif
            break;
      case EMAIL :
            if ( edf_time[3] == 0 )
                  edf_time[3] = msg->timestamp()+email_deadline;
            email_queue.insertHead(msg);
            #ifdef TRACE
            ev << simTime() << " :put packet in email queue" << endl;
            #endif
            break;

      }
}

void CPS_Insertion::queueing(cMiniCell *msg)
{
switch (scheduling)
      {
      case FIFO : put_fifo(msg);break;
      case RR : put_rr(msg);break;//for one type of traffic
      case PRIOR : put_prior(msg);break;
```

```
      case WRR : put_wrr(msg);break;
      case EDF : put_edf(msg);break;
      case DWRR : put_dwrr(msg);break;
      }
}

void CPS_Insertion::timer_cu_expired()
{
#ifdef TRACE
ev << "Time: " << simTime() << " Timer-CU expired" << endl;
#endif
delete timercu;
timercu = NULL;
timer_expired = YES;
switch (state)
      {
      case IDLE:
                  break;
      case PART:
                  if (permit == YES)
                        {
                        switch (ATC)
                              {
                              case DBR:
                                    permit = NO;
                                    break;
                              case UBR:
                                    break;
                              case SBR:
                                    permit = NO;
                                    break;
                              };
                        ready_cell = cell_in_gare;
                        cell_in_gare = NULL;
                        cell_in_gare_state = IDLE;
                        CELLHEADER header;
                        header.initialize();
                        header = ready_cell->header();
                        header.PAD = free_place;
                        ready_cell->setHeader(header);
                        send_cell();
                        ready_cell = NULL;
                        state = IDLE;
                        }
                  else
                        state = SEND;
                  break;
      case SEND:
                  timer_expired = YES;
                  if (cell_in_gare_state != FULL)
                        cell_in_gare_state = SEND;
                  break;
      case FULL:
                  timer_expired = YES;
                  if (cell_in_gare_state != FULL)
                        cell_in_gare_state = SEND;
                  break;
      }
if (locking_or_not == LOCKING)
```

```
                              lock == YES;
}


void CPS_Insertion::get_fifo()
{
if (!FIFOqueue.empty())
      {
      minicell_in_service = (cMiniCell *)FIFOqueue.getTail();
      double size = minicell_in_service->length();
      FIFO_size = FIFO_size - size;
      #ifdef TRACE
            ev << "fifo length:  " << FIFOqueue.length() << endl;
            ev << "-----------------------------------" << endl;
            ev << "packaging new packet" << endl;
            ev << "-----------------------------------" << endl;
      #endif
      }
else
      {
      #ifdef TRACE
            ev << "FIFO queue empty!!!" << endl;
      #endif
      }
}

void CPS_Insertion::get_rr()
{
int all_empty = YES;
for (int i=0;i<248;i++)
      {
      if (RRqueue[RRindex].empty())
            {
            RRindex++;
            if (RRindex > 247)
                        RRindex = 0;
            }
      else
            {
            all_empty = NO;
            break;
            }
      }
if (all_empty == NO)
      {
                  #ifdef TRACE
                  ev << "serving user " << RRindex + 1 << endl;
                  #endif

      minicell_in_service = (cMiniCell *)RRqueue[RRindex].getTail();
      RRindex++;
      if (RRindex > 247)
            RRindex = 0;
      }
else
      {
      #ifdef TRACE
      ev << "all RR queues empty !" << endl;
```

```
      #endif
      }

#ifdef TRACE
ev << "index: " << RRindex + 1 << endl;
#endif
}

void CPS_Insertion::get_prior()
{
if ( !voice_queue.empty() )
      {
      minicell_in_service = (cMiniCell *)voice_queue.getTail();
      double size = (double) minicell_in_service->length();
      voice_size -= size;
      }
else
      if ( !web_queue.empty() )
            {
            minicell_in_service = (cMiniCell *)web_queue.getTail();
            double size = (double) minicell_in_service->length();
            data_size -= size;
            }
      else
            if ( !ftp_queue.empty() )
                  minicell_in_service = (cMiniCell *)ftp_queue.getTail();
            else
                  if ( !email_queue.empty() )
                        minicell_in_service = (cMiniCell
*)email_queue.getTail();
                  else
                        {
                        #ifdef TRACE
                        ev << "all queues empties...!!!" << endl;
                        #endif
                        }
}

void CPS_Insertion::get_wrr()
{
double size;
for (int i=0; i<wrr_cycle; i++)
      {
      switch ( wrr_table[wrr_index] )
            {
            case VOICE:
                  if ( !voice_queue.empty() )
                        {
                        minicell_in_service = (cMiniCell
*)voice_queue.getTail();
                        size = (double) minicell_in_service->length();
                        voice_size -= size;
                        }
                  break;
            case WEB:
                  if ( !web_queue.empty() )
                        {
                        minicell_in_service = (cMiniCell
*)web_queue.getTail();
```

```
                        size = (double) minicell_in_service->length();
                        data_size -= size;
                        }
                  break;
            case FTP:
                  if ( !ftp_queue.empty() )
                        minicell_in_service = (cMiniCell
*)ftp_queue.getTail();
                  break;
            case EMAIL:
                  if ( !email_queue.empty() )
                        minicell_in_service = (cMiniCell
*)email_queue.getTail();
                  break;
            }
      wrr_index++;
      if (wrr_index >= wrr_cycle)
            wrr_index =0;
      if (minicell_in_service != NULL)
            break;
      }
}

void CPS_Insertion::get_dwrr()
{
double size;
int index;
minicell_in_service = NULL;
for (int i=0; i<dwrr_round; i++)
{
if (nb_zeros == 4)
      {
      index = dwrr_index;
      init_dwrr();
      dwrr_index = index;
      }
switch (dwrr_index)
      {
      case 0:
            if (dwrr_table[dwrr_index] != 0)
                  {
                  if ( !voice_queue.empty() )
                        {
                        minicell_in_service = (cMiniCell
*)voice_queue.getTail();
                        size = (double)minicell_in_service->length();
                        voice_size -= size;
                        }
                  dwrr_table[dwrr_index] -= 1;
                  if (dwrr_table[dwrr_index] == 0)
                        {
                        nb_zeros++;
                        dwrr_index++;
                        }
                  }
            else
                  dwrr_index++;
            break;
      case 1:
```

```
            if (dwrr_table[dwrr_index] != 0)
                  {
                  if ( !web_queue.empty() )
                        {
                        minicell_in_service = (cMiniCell
*)web_queue.getTail();
                        size = (double) minicell_in_service->length();
                        data_size -= size;
                        }
                  dwrr_table[dwrr_index] -= 1;
                  if (dwrr_table[dwrr_index] == 0)
                        {
                        nb_zeros++;
                        dwrr_index++;
                        }

                  }
            else
                  dwrr_index++;
            break;
      case 2:
            if (dwrr_table[dwrr_index] != 0)
                  {
                  if ( !ftp_queue.empty() )
                        {
                        minicell_in_service = (cMiniCell
*)ftp_queue.getTail();
                        }
                  dwrr_table[dwrr_index] -= 1;
                  if (dwrr_table[dwrr_index] == 0)
                        {
                        nb_zeros++;
                        dwrr_index++;
                        }
                  }
            else
                  dwrr_index++;
            break;
      case 3:
            if (dwrr_table[dwrr_index] != 0)
                  {
                  if ( !email_queue.empty() )
                        {
                        minicell_in_service = (cMiniCell
*)email_queue.getTail();
                        }
                  dwrr_table[dwrr_index] -= 1;
                  if (dwrr_table[dwrr_index] == 0)
                        {
                        nb_zeros++;
                        dwrr_index = 0;
                        }
                  }
            else
                  dwrr_index = 0;
            break;
      };
if (minicell_in_service != NULL)
                  break;
```

```
}
}

void CPS_Insertion::get_edf()
{
double min;
double size;
int index_min;
int start_index = 5;
for (int i=0;i<4;i++)
      {
      #ifdef TRACE
      ev << "edf_time[" << i << "] = " << edf_time[i] << endl;
      #endif
      if (edf_time[i] != 0)
            {
            min = edf_time[i];
            index_min = i;
            start_index = i+1;
            #ifdef TRACE
            ev << "start_index = " << start_index << endl;
            #endif
            break;
            }
      }
if (start_index != 5)
      {
      #ifdef TRACE
      ev << "start_index != 5" << endl;
      #endif
        for (int i=start_index;i<4;i++)
            {
            #ifdef TRACE
            ev << "edf_time[" << i << "] = " << edf_time[i] << endl;
            #endif
            if ( (edf_time[i] != 0)&&( edf_time[i] < min) )
                  {
                  min = edf_time[i];
                  index_min = i;
                  }
            }
      #ifdef TRACE
      ev << simTime() << " : get...index_min = " << index_min << endl;
      #endif
      switch (index_min)
            {
            case 0:
                  minicell_in_service = (cMiniCell *)voice_queue.getTail();
                  size = (double) minicell_in_service->length();
                  voice_size -= size;
                        if ( voice_queue.empty() )
                              {
                              edf_time[0] = 0;
                              }
                        else
                              {
                              cMiniCell *mini = (cMiniCell
*)voice_queue.tail();
```

```
                              edf_time[0] = mini->timestamp() +
voice_deadline;
                              #ifdef TRACE
                              ev << "new deadline for voice: " <<
edf_time[0] << endl;
                              #endif
                              //mini = NULL;
                              }
                        //ev << "==============================" << endl;
                        break;
            case 1:
                  minicell_in_service = (cMiniCell *)web_queue.getTail();
                  size = (double) minicell_in_service->length();
                  data_size -= size;
                        #ifdef TRACE
                        ev << simTime() << " :get packet from web queue" <<
endl;
                        #endif
                        if ( web_queue.empty() )
                              {
                              edf_time[1] = 0;
                              }
                        else
                              {
                              cMiniCell *mini = (cMiniCell
*)web_queue.tail();
                              edf_time[1] = mini->timestamp() +
web_deadline;
                              #ifdef TRACE
                              ev << "new deadline for web: " << edf_time[1]
<< endl;
                              #endif
                              //mini = NULL;
                              }
                        break;
            case 2: minicell_in_service = (cMiniCell *)ftp_queue.getTail();
                        if ( ftp_queue.empty() )
                              edf_time[2] = 0;
                        else
                              {
                              cMiniCell *mini = (cMiniCell
*)ftp_queue.tail();
                              edf_time[2] = mini->timestamp() +
ftp_deadline;
                              //mini = NULL;
                              }
                        break;
            case 3: minicell_in_service = (cMiniCell
*)email_queue.getTail();
                        if ( email_queue.empty() )
                              edf_time[3] = 0;
                        else
                              {
                              cMiniCell *mini = (cMiniCell
*)email_queue.tail();
                              edf_time[3] = mini->timestamp() +
email_deadline;
                              //mini = NULL;
                              }
```

```
                        break;
            }

      }
else
      ev << "all EDF queues empties !!!" << endl;
}



void CPS_Insertion::packaging()
{
#ifdef TRACE
ev << "packaging..." << endl;
#endif
if (lock == NO)
{
if (minicell_in_service == NULL)
      {
      switch (scheduling)
            {
            case FIFO : get_fifo();break;
            case RR : get_rr();break;
            case WRR : get_wrr();break;
            case PRIOR : get_prior();break;
            case EDF : get_edf();break;
            case DWRR : get_dwrr();break;
            }
      if ( minicell_in_service != NULL )
            {
            rest = minicell_in_service->length();
            newpacket = YES;
            }
      }
else
       {
       #ifdef TRACE
       ev << "there is a remained packet under packaging" << endl;
       #endif
       }
if (minicell_in_service != NULL)
      {
      if (cell_in_gare == NULL)
            generate_new_cell();
      if (cell_in_gare->header().empty_cell() == YES)
            {
            CELLHEADER header;
            header.initialize();
            header = cell_in_gare->header();
            if (newpacket == YES)
                  header.OSF = 1;
            else

                  header.OSF = rest + 1;
            cell_in_gare->setHeader(header);
            }
      if (newpacket == YES)
                  for (int i=0; i<7;i++)
                        {
```

```
                        if ( cell_in_gare-
>header().headers[i].empty_minicell() == YES )
                                {
                                MINICELLHEADER miniheader;
                                miniheader.initialize();
                                miniheader = minicell_in_service->header();

                                CELLHEADER header;
                                header.initialize();
                                header = cell_in_gare->header();
                                header.headers[i] = miniheader;
                                cell_in_gare->setHeader(header);
                                newpacket = NO;
                                break;
                                }
                        }
      int len = rest;
      if (len <= free_place)
            {
            free_place = free_place - len;
            rest = 0;
            delete minicell_in_service;
            minicell_in_service = NULL;
            if (free_place == 0)
                  {
                  ready_cell = cell_in_gare;
                  cell_in_gare = NULL;
                  cell_in_gare_state = IDLE;
                  state = FULL;
                  if ( timercu != NULL )
                              {
                              delete cancelEvent(timercu);
                              timercu = NULL;
                              timer_expired = YES;
                              }
                  if (permit == YES)
                        {
                        CELLHEADER header;
                        header.initialize();
                        header = ready_cell->header();
                        header.PAD = free_place;
                        ready_cell->setHeader(header);
                        send_cell();
                        ready_cell = NULL;
                        state = IDLE;
                        packaging();
                        }
                  }
            else
                  {
                  if (state == SEND)
                        state == SEND;
                  else
                        state = PART;
                  cell_in_gare_state = PART;

                  if ( free_place <= (47-enable_size) )
                        {
                        state = SEND;
```

```
                        cell_in_gare_state = SEND;
                        if (permit == YES)
                              {
                              if ( timercu != NULL )
                                    {
                                    delete cancelEvent(timercu);
                                    timercu = NULL;
                                    timer_expired = YES;
                                    }
                              switch (ATC)
                                    {
                                    case DBR:
                                          permit = NO;
                                          break;
                                    case UBR:
                                          break;
                                    case SBR:
                                          permit = NO;
                                          break;
                                    };
                              ready_cell = cell_in_gare;
                              cell_in_gare = NULL;
                              cell_in_gare_state = IDLE;
                              CELLHEADER header;
                              header.initialize();
                              header = ready_cell->header();
                              header.PAD = free_place;
                              ready_cell->setHeader(header);
                              send_cell();
                              ready_cell = NULL;
                              state = IDLE;
                              packaging();
                              }
                        else
                              {
                              if (locking_or_not == NO_LOCKING)
                                    packaging();
                              else
                                    {
                                    if ( timercu->isScheduled() )
                                    packaging();
                                    }
                              }
                        }
                  else
                        {
                        if (locking_or_not == NO_LOCKING)
                              packaging();
                        else
                              {
                              if ( timercu->isScheduled() )
                                    packaging();
                              }
                        }
                  }
            }
      else
            {
            rest = len - free_place;
```

```
#ifdef TRACE
ev << "the rest before packaging it: " << rest << endl;
#endif
free_place = 0;
state = FULL;
newpacket = NO;
ready_cell = cell_in_gare;
cell_in_gare = NULL;
cell_in_gare_state = IDLE;
generate_new_cell();
CELLHEADER header;
header.initialize();
header = cell_in_gare->header();
header.OSF = rest + 1;
cell_in_gare->setHeader(header);
free_place = free_place - rest;
rest = 0;
delete minicell_in_service;
minicell_in_service = NULL;
if (free_place == 0)
      {
      cell_in_gare_state = FULL;
      if ( timercu != NULL )
            {
            delete cancelEvent(timercu);
            timercu = NULL;
            timer_expired = YES;
            }
      }
else
      {
      if ( free_place <= (47-enable_size) )
            cell_in_gare_state = SEND;
      else
            cell_in_gare_state = PART;
      }
if (permit == YES)
      {
      CELLHEADER header;
      header.initialize();
      header = ready_cell->header();
      header.PAD = 0;
      ready_cell->setHeader(header);
      send_cell();
      ready_cell = NULL;
      switch (cell_in_gare_state)
            {
            case PART:
                  state = PART;
                  packaging();
                  break;
            case FULL:
                  state = FULL;
                  ready_cell = cell_in_gare;
                  cell_in_gare = NULL;
                  cell_in_gare_state = IDLE;
                  break;
            case SEND:
                  state = SEND;
```

```
                              packaging();
                              break;
                        }
                  }
            }
      }
else
      {
      #ifdef TRACE
      ev << "no packet to packaging!!!" << endl;
      #endif
      }
}
}

void CPS_Insertion::send_cell()
{
lock = YES;
switch (ATC)
      {
      case DBR:
            permit = NO;
            break;
      case UBR:
            break;
      case SBR:
            permit = NO;
            break;
      };
CELLHEADER header;
header.initialize();
header = ready_cell->header();
header.timestamp = simTime();
ready_cell->setTimestamp();
ready_cell->setHeader(header);
sendDelayed(ready_cell,insertion_processing_time,"out");
#ifdef TRACE
ev << "*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=" << endl;
ev << "Time: " << simTime() << endl;
ev << "Ev: send cell" << endl;
ev << "Source: " << id << endl;
ev << "OSF: " << header.OSF << endl;
ev << "PAD: " << header.PAD << endl;
ev << "Rest = " << rest << endl;
#endif
if ( header.empty_cell() )
      {
      #ifdef TRACE
      ev << "-------------------------------------" << endl;
      ev << "empty CPS-PDU !" << endl;
      ev << "-------------------------------------" << endl;
      #endif
      }
else
      for (int i=0;i<7;i++)
       {
       if ( !header.headers[i].empty_minicell() )
            {
            #ifdef TRACE
```

```
            ev << "------------------------------------" << endl;
            ev << "minicell " << i << " :" << endl;
            ev << "source: " << header.headers[i].link1.VPI << "." <<
header.headers[i].link1.VCI << "." << header.headers[i].link1.CID << endl;
            ev << "LI: " << header.headers[i].LI << endl;
            ev << "FP-PDU length: " << header.headers[i].FP_PDU_length <<
endl;
            ev << "MORE: " << header.headers[i].MORE << endl;
            ev << "------------------------------------" << endl;
            #endif
            }
        }
#ifdef TRACE
ev << "*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=" << endl;
#endif
lock = NO;// don't delete this line !!!
}

Define_Module( CPS_Extraction )

void CPS_Extraction::initialize()
{
service_time = extraction_processing_time;
cell_in_service = NULL;
end_extraction = new cSig("EndExtraction",ENDEXTRACTION);
minicell_to_send = NULL;
}

void CPS_Extraction::handleMessage(cMessage *msg)
{
start_extraction(msg);
}

void CPS_Extraction::start_extraction(cMessage *msg)
{
if (msg == end_extraction)
      {
      #ifdef TRACE
      ev << "Time: " << simTime() << ": end extraction" << endl;
      #endif
      extract((cCell *)cell_in_service);
      if (queue.empty())
            cell_in_service = NULL;
      else
            {
            cell_in_service = (cCell *)queue.getTail();
            #ifdef TRACE
            ev << "Time: " << simTime() << ": start extraction" << endl;
            #endif
            scheduleAt(simTime()+service_time,end_extraction);
            }
      }
else
      {
      cCell *cell = (cCell *)msg;
      if (cell_in_service == NULL)
            {
            cell_in_service = cell;
            #ifdef TRACE
```

```
            ev << "Time: " << simTime() << ": start extraction" << endl;
            #endif
            scheduleAt(simTime()+service_time,end_extraction);
            }
      else
            {
            queue.insertHead(cell);
            }
      }
}

void CPS_Extraction::extract(cCell *cell)
{

CELLHEADER header = cell->header();

if (header.empty_cell() == NO)
{
simtime_t time = header.timestamp;
if (minicell_to_send != NULL)
      send_minicell();

int length_left = cell_size - ATM_Header - 1;
length_left = length_left - header.OSF + 1;

for (int i=0;i<7;i++)
      {
      if ( header.headers[i].empty_minicell() == YES )
            break;
      else
            {
            MINICELLHEADER miniheader;
            miniheader = header.headers[i];
            int len = miniheader.LI + CPS_Header;
            cMiniCell *minicell = new cMiniCell;
            switch (miniheader.TYPE)
                  {
                  case VOICE: minicell->setName("Voice");minicell-
>setType(VOICE);break;
                  case WEB: minicell->setName("Web");minicell-
>setType(WEB);break;
                  case FTP: minicell->setName("Ftp");minicell-
>setType(FTP);break;
                  case EMAIL: minicell->setName("Email");minicell-
>setType(EMAIL);break;
                  };
            minicell->setTimestamp(miniheader.timestamp);
            minicell->setKind(miniheader.TYPE);
            minicell->setLength(len);
            minicell->setHeader(miniheader);
            minicell_to_send = minicell;
            if (len <= length_left)
                  {
                  length_left = length_left - len;
                  send_minicell();
                  }
            else
                  {
                  length_left = 0;
```

```
                    break;
                    }
                }
        }
}
else
        {
        }
delete cell;
}

void CPS_Extraction::send_minicell()
{
CPS_header(minicell_to_send);
minicell_to_send = NULL;
}

void CPS_Extraction::CPS_header(cMiniCell *msg)
{
int len = minicell_to_send->length();
minicell_to_send->setLength(len - CPS_Header);
send(minicell_to_send,"out");
}

Define_Module( ATM_Emission )

void ATM_Emission::initialize()
{
id = par("id");
double a,b,c;

Speed = par("Link_Speed");
delta = (double)(8*cell_size/(double)Speed);

BW = par("VC_PCR");
T_pcr = (double)(8*cell_size/(double)BW);
TAT_pcr = 0;
a = (double)T_pcr/(double)delta;
b = (double)80*(1- (delta/T_pcr));
c = (double)max(a,b);
taw_pcr = (double)c*delta;

SCR[0] = par("VC_SCR");
double rapport = BW/SCR[0];
for (int i=1;i<25;i++)
        {
        rapport += 0.05;
        SCR[i] = (double)BW/(double)rapport;
        }

for (int i=0;i<25;i++)
        {
        T_scr[i] = (double)(8*cell_size/(double)SCR[i]);
        double IBT = (double)((MBS - 1)*(T_scr[i] - T_pcr));
        a = (double)T_scr[i]/(double)delta;
        b = (double)80*(1- (delta/T_scr[i]));
        c = (double)max(a,b);
        double tolerance_scr = (double)c*delta;
        taw_scr[i] = (double)(tolerance_scr + IBT);
```

```
      TAT_scr[i] = 0;
      nb_non_conforming_cells[i] = 0;
      }

cell_emission_time = (double) 8*cell_size/(double)Speed;
end_emission = new cMessage("EndEmission",ENDEMISSION);
stat_buffer.setName("buffer length");
vect_buffer.setName("buffer length");
stat_filling.setName("filling");
under_emission_cell = NULL;
sn = 0;
sum = 0;
nb_of_cells = 0;
total_cells = 0;
nb_of_full_cells = 0;
R = 0;

if ( (id == 0)||(id == monitored_nodeB) )
{
if (id == 0)
      {
      ev << "HUB --> RNC Iub PCR = " << BW/1000.0 << " Kbps" <<endl;
      if (ATC == SBR)
            {
            ev << "HUB --> RNC Iub: " << endl;
            }
      stat_rate.setName("Hub-RNC BitRate");
      vect_rate.setName("Hub-RNC BitRate");
      vector_results.setName("Hub-RNC VC performance");
      }
else
      {
      ev << "NodeB --> HUB Iub PCR = " << BW/1000.0 << " Kbps" <<endl;
      if (ATC == SBR)
            {
            ev << "NodeB --> HUB Iub: " << endl;
            }
      stat_rate.setName("NodeB-Hub BitRate");
      vect_rate.setName("NodeB-Hub BitRate");
      vector_results.setName("NodeB-Hub VC performance");
      }
}
#ifdef BITRATE
if ( (id == 0)||(id == monitored_nodeB) )
{
test_msg = new cMessage("Test",TEST_MSG);
scheduleAt(collect_statistics_start_time + T,test_msg);
}
#endif
cnb = 0;
last_check = 0;
max_rate = 0;
}

void ATM_Emission::handleMessage(cMessage *msg)
{
if (msg == test_msg)
      {
```

```
      if ( (simTime() >= collect_statistics_start_time) && (simTime() <=
collect_statistics_stop_time) )
            {
            double t = 1000*T;
            double bitrate = (double) 424*R/(double)t;
            stat_rate.collect(bitrate);
            }
      R = 0;
      scheduleAt(simTime()+T,test_msg);
      }
else
      emission(msg);
}

void ATM_Emission::finish()
{
#ifdef STATISTICS

double utile_bandwidth = (double) 47*nb_of_cells;
double filling_ratio= (double) sum/(double)utile_bandwidth;

double time_interval = collect_statistics_stop_time -
collect_statistics_start_time;
double total_rate = (double) 8*cell_size*nb_of_cells/(double)time_interval;
double utilization_ratio = (double) total_rate/(double)BW;
total_rate = (double)total_rate/1000.0;

double ratio_full_cells = (double)100*nb_of_full_cells/(double)nb_of_cells;
double ratio_timeout_cells = (double)(100 - ratio_full_cells);

if ( (id == 0)||(id == monitored_nodeB) )
{
if (id == 0)
      ev << "Hub --> RNC link: " << endl;
else
      ev << "NodeB --> Hub link: " << endl;

#ifdef RESULTS
vector_results.record(stat_rate.max());
vector_results.record(stat_rate.mean());
vector_results.record(100*filling_ratio);
vector_results.record(100*utilization_ratio);
vector_results.record(ratio_full_cells);
vector_results.record(ratio_timeout_cells);
vector_results.record(53*stat_buffer.max());
#endif
ev << "Average ATM BitRate: " << stat_rate.mean() << " Kbps " << endl;
ev << "Peak ATM BitRate: " << stat_rate.max() << " Kbps" << endl;
ev << "filling ratio: " << 100*filling_ratio << " %" << endl;
ev << "utilization of the VC bandwidth: " << 100*utilization_ratio << " %"
<< endl;
ev << "----------------------------------------" << endl;
endl;
if (ATC == SBR)
      for (int i=0;i<25;i++)
      {
      double non_conforming_cells_ratio =
(double)100*nb_non_conforming_cells[i]/(double)total_cells;
      }
```

```
#endif
}
}

void ATM_Emission::ATM_header(cCell *msg)
{
CELLHEADER header = msg->header();
header.SN = sn;
sn = 1 - sn;
msg->setHeader(header);
}

void ATM_Emission::calculate_utilization(cCell *cell)
{
if ( (simTime() >= collect_statistics_start_time) && (simTime() <=
collect_statistics_stop_time) )
      {
      R++;
      nb_of_cells++;
      CELLHEADER header = cell->header();
      int pad = header.PAD;
      sum += 47 - pad;
      double fill = (double)100*(47 - pad)/47.0;
      stat_filling.collect(fill);
      if (pad == 0)
            nb_of_full_cells++;

      if (last_check == 0)
            {
            last_check = simTime();
            cnb = 1;
            }
      else
            {
            if (cnb == TBS)
                  {
                  double t = 1000*(simTime() - last_check);
                  double rate = (double)8*cell_size*cnb/(double)t;
                  if (rate > max_rate )
                        max_rate = rate;
                  last_check = simTime();
                  cnb = 1;
                  }
            else
                  cnb++;
            }
      }
}

void ATM_Emission::emission(cMessage *msg)
{
long num;
double new_time;
double new_value;
if (msg == end_emission)
{
      calculate_utilization(under_emission_cell);
      #ifdef TRACE
```

```
      ev << "Time: " << simTime() << "end emission...length: " <<
under_emission_cell->length() << endl;
      #endif
      send(under_emission_cell,"out");
      under_emission_cell = NULL;
      if ( buffer.empty() == NO )
      {
      under_emission_cell = (cCell *)buffer.getTail();
      ATM_header(under_emission_cell);
      if ( (simTime() >= collect_statistics_start_time) && (simTime() <=
collect_statistics_stop_time) )
            total_cells++;
      num = (long)ceil(simTime()/(double)delta);
      new_time = (double)num*delta;
      #ifdef TRACE
      ev << "Time: " << new_time << "start emission" << endl;
      #endif
            double nt;
      double tol;
      double t = (double)new_time;
      switch (ATC)
            {
            case DBR:
                  if (TAT_pcr == 0)
                        {
                        TAT_pcr = (double)t;
                        };
                  tol = (double)(TAT_pcr-taw_pcr);
                  if (t < tol)
                        {
                        nt = (long)ceil(tol/(double)delta);
                        nt = (double)nt*delta;
                        new_time = nt;
                        new_value = (double)((double)max(nt,TAT_pcr) +
T_pcr);
                        TAT_pcr = (double)new_value;
                        }
                  else
                        {
                        new_value = (double)((double)max(t,TAT_pcr) +
T_pcr);
                        TAT_pcr = (double)new_value;
                        }
            break;
            case SBR:
                  if (TAT_pcr == 0)
                        TAT_pcr = (double)t;
                  for (int i=0;i<25;i++)
                        if (TAT_scr[i] == 0)
                              TAT_scr[i] = (double)t;

                  tol = (double)(TAT_pcr-taw_pcr);
                  if (t < tol)
                        {
                        nt = (long)ceil(tol/(double)delta);
                        nt = (double)nt*delta;
                        new_time = nt;
                        new_value = (double)((double)max(nt,TAT_pcr) +
T_pcr);
```

```
                        TAT_pcr = (double)new_value;
                        }
                  else
                        {
                        new_value = (double)((double)max(t,TAT_pcr) +
T_pcr);
                        TAT_pcr = (double)new_value;
                        }
                  t = new_time;

                  for (int i=0;i<25;i++)
                  {
                  tol = (double)(TAT_scr[i]-taw_scr[i]);
                  if (t < tol)
                        {
                        //non conforming cell
                        if ( (simTime() >= collect_statistics_start_time)
&& (simTime() <= collect_statistics_stop_time) )
                              {
                              nb_non_conforming_cells[i]++;
                              }
                        }
                  else
                        {
                        new_value = (double)((double)max(t,TAT_scr[i]) +
T_scr[i]);
                        TAT_scr[i] = (double)new_value;
                        }
                  }
            break;
            };
      scheduleAt(new_time+delta,end_emission);
      }
}
else
{
cCell *cell = (cCell *)msg;

if ( (buffer.empty() == YES)&&(under_emission_cell == NULL) )
      {
      under_emission_cell = cell;
      ATM_header(under_emission_cell);
      if ( (simTime() >= collect_statistics_start_time) && (simTime() <=
collect_statistics_stop_time) )
            total_cells++;
      num = (long)ceil(simTime()/(double)delta);
      new_time = (double)num*delta;
      #ifdef TRACE
      ev << "Time: " << new_time << "start emission" << endl;
      #endif
      double tol;
      double nt;
      double t = (double)new_time;
      switch (ATC)
            {
            case DBR:
                  if (TAT_pcr == 0)
                        {
                        TAT_pcr = (double)t;
```

```
                        };
                  tol = (double)(TAT_pcr-taw_pcr);
                  if (t < tol)
                        {
                        nt = (long)ceil(tol/(double)delta);
                        nt = (double)nt*delta;
                        new_time = nt;
                        new_value = (double)((double)max(nt,TAT_pcr) +
T_pcr);
                        TAT_pcr = (double)new_value;
                        }
                  else
                        {
                        new_value = (double)((double)max(t,TAT_pcr) +
T_pcr);
                        TAT_pcr = (double)new_value;
                        }
            break;
            case SBR:
                  if (TAT_pcr == 0)
                        TAT_pcr = (double)t;
                  for (int i=0;i<25;i++)
                        if (TAT_scr[i] == 0)
                              TAT_scr[i] = (double)t;

                  tol = (double)(TAT_pcr-taw_pcr);
                  if (t < tol)
                        {
                        nt = (long)ceil(tol/(double)delta);
                        nt = (double)nt*delta;
                        new_time = nt;
                        new_value = (double)((double)max(nt,TAT_pcr) +
T_pcr);
                        TAT_pcr = (double)new_value;
                        }
                  else
                        {
                        new_value = (double)((double)max(t,TAT_pcr) +
T_pcr);
                        TAT_pcr = (double)new_value;
                        }
                        t = new_time;

                        for (int i=0;i<25;i++)
                        {
                        tol = (double)(TAT_scr[i]-taw_scr[i]);
                        if (t < tol)
                              {
                              if ( (simTime() >=
collect_statistics_start_time) && (simTime() <=
collect_statistics_stop_time) )
                                    {
                                    nb_non_conforming_cells[i]++;
                                    }
                              }
                        else
                              {
                              new_value = (double)((double)max(t,TAT_scr[i])
+ T_scr[i]);
```

```
                              TAT_scr[i] = (double)new_value;
                              }
                        }
            break;
            };
            scheduleAt(new_time+delta,end_emission);
      }
else
      {
      buffer.insertHead(cell);
      if ( (simTime() >= collect_statistics_start_time) && (simTime() <=
collect_statistics_stop_time) )
            {
            double len = (double) buffer.length();
            stat_buffer.collect( len );
            }
      }
}
}

Define_Module( ATM_Reception )

void ATM_Reception::handleMessage(cMessage *msg)
{
send(msg,"out");
}

Define_Module( RNCENTRY )


void RNCENTRY::initialize()
{
}

void RNCENTRY::handleMessage(cMessage *msg)
{
cCell *cell = (cCell *)msg;
if ( (cell->header().VPI == 0)||(cell->header().VPI == monitored_nodeB) )
      send(cell,"out");
else
      {
      ev << "=======================================" << endl;
      ev << simTime() << ": delete cell from RNCEntry" << endl;
      ev << "VPI = " << cell->header().VPI << endl;
      ev << "=======================================" << endl;
      delete cell;
      }
}

Define_Module( SINKENTRY )

void SINKENTRY::initialize()
{
}

void SINKENTRY::handleMessage(cMessage *msg)
{
cMiniCell *minicell = (cMiniCell *)msg;
if (minicell->header().link1.VPI == monitored_nodeB)
```

```
      send(minicell,"out");
else
      {
      ev << "=======================================" << endl;
      ev << simTime() << ": delete minicell from SinkEntry" << endl;
      ev << "VPI = " << minicell->header().link1.VPI << endl;
      ev << "=======================================" << endl;
      delete minicell;
      }
}
```

Fichier cps_atm.ned

```
simple CPS_Insertion
      parameters:
            id,
            Timer_CU,
            VC_PCR,
            VC_SCR;
      gates:
            in: in;
            out: out;
endsimple

simple CPS_Extraction
      gates:
            in: in;
            out: out;
endsimple

simple ATM_Emission
      parameters:
            id,
            VC_PCR,
            VC_SCR,
            Link_Speed;
      gates:
            in: in;
            out: out;
endsimple

simple ATM_Reception
      gates:
            in: in;
            out: out;
endsimple

simple SINKENTRY
      gates:
            in: in;
            out: out;
endsimple

simple RNCENTRY
      gates:
            in: in;
            out: out;
endsimple
```

Liste des acronymes

3G	Third generation
3GPP	Third Generation Partnership Project
AAL	ATM Adaptation Layer
AAL2	ATM Adaptation Layer – type 2
AAL5	ATM Adaptation Layer – type
ABR	Available Bit Rate
ABT	ATM Block Transfer
ABT-DT	ABT – Delayed Transmission
ABT-IT	ABT - Immediate Transmission
AF	Assured Forwarding
ALCAP	Access Linl Control Application Protocol
AM	Acknowledged Mode
AMR	Adaptive Multi-Rate codec
ATC	ATM Transfer Capability
ATM	Asynchronous Transfer Mode
BS	Bearer Service
CAC	Connection Admission Control
CBR	Constant Bit Rate
CDMA	Code Division Multiple Access
CID	Channel Identifier
CN	Core Network
CPS	Common Part Sublayer
CRC	Cyclic Redundancy Check
CS	Circuit Switched
DBR	Deterministic Bit Rate
DCH	Dedicated Channel
DiffServ	Differentiated Services
DL	Down Link
D-RNC	Drift – RNC
D-RNS	Drift – RNS
EDF	Earliest Deadline First
EF	Expedited Forwarding
ETSI	European Telecommunication Standards Institute
FDD	Frequency Division Duplex
FDMA	Frequency Division Multiple Access
FIFO	First In First Out
FP	Frame Protocol
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HDLC	High level Data Link Control
IETF	Internet Engineering Task Force
IMT-2000	International Mobile Telecommunications for the year 2000
IntServ	Integrated Services
IP	Internet Protocol
ITU-T	International Telecommunication Union – Telecommunication standardization sector
L2TP	Layer 2 Transport Protocol
MAC	Medium Access Control
MBS	Maximum Burst Size
MPLS	Multi-Protocol Label Switching
MRU	Maximum Receive Unit
MT	Mobile Termination
MTU	Maximum Transmission Unit

PCR	Peak Cell Rat
PDU	Protocol Data Unit
PPP	Point-to-Point Protocol
PPP-ML-MC	PPP with Multi-Link Multi-Class extensions
PPP-MP	PPP Multi-link Protocol
PPP-mux	PPP multiplexing
PQ	Priority Queueing
PS	Packet Switched
PSTN	Public Switched Telephone Network
PVC	Permanent Virtual Circuit
QoS	Quality of Service
RAB	Radio Access Bearer
RB	Radio Bearer
RFC	Request For Comments
RLC	Radio Link Control
RNC	Radio Network Controller
RNL	Radio Network Layer
RNS	Radio Network Subsystem
RR	Round Robin
RRC	Radio Resource Control
RTP	Real Time Transport Protocol
SAP	Service Access Point
SAR	Segmentation And Reassembly
SBR	Statistical Bit Rate
SCR	Sustainable Cell Rate
SDU	Service Data Unit
SID	Silence Descriptor
S-RNC	Serving – RNC
S-RNS	Serving – RNS
SSCS	Service Specific Convergence Sublayer
SSCS-ADT	SSCS – Assured Data Transfer
SSCS-SAR	SSCS – Segmentation And Reassembly
SSCS-TED	SSCS – Transmission Error Detection
StdDev	Standard Deviation of delay
SVC	Switched Virtual Circuit
TB	Transport Block
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
TE	Terminal Equipment
TM	Transparent Mode
TNL	Transport Network Layer
TTI	Transmission Time Interval
UDD	Unconstrained Delay Data
UDP	User Datagram Protocol
UE	User Equipment
UL	Up Link
UM	Unacknowledged Mode
UMTS	Universal Mobile Telecommunication System
UTRAN	UMTS Terrestrial Radio Access Network
UUI	User-to-User Information
VBR	Variable Bit Rate
VC	Virtual Channel
VCC	Virtual Channel Connection
VCI	Virtual Channel Identifier
VoIP	Voice over IP
VP	Virtual Path
VPC	Virtual Path Connection

VPI	Virtual Path Identifier
WRR	Weighted Round Robin