ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE ÉLECTRIQUE
M.Ing.

PAR
MAMMOUH, Bouchaib

CONCEPTION D'ALGORITHMES DE MISE EN TRAME SELON LA NORME IEEE802.16E

MONTRÉAL, LE 30 AVRIL 2007

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. François Gagnon, directeur de mémoire
Département de génie électrique à l'École de technologie supérieure

Michel Kadoch, président du jury
Département de génie électrique à l'École de technologie supérieure

Mme Christine Tremblay, membre du jury
Département de génie électrique à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY

LE 18 AVRIL 2007

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je souhaite remercier *M. François Gagnon, Ph.D*, pour m'avoir dirigé tout au long de la maîtrise ainsi que pour ses judicieux conseils qu'il m'a fournis pour la réalisation de cette mémoire.

Je souhaite aussi remercier les compagnies *Ultra Electronics* et *SR Telecom* pour leurs implication dans ce projet et aussi leurs support financiers.

Je souhaite aussi remercier les membres du LACIME pour leur soutien et aide et surtout *Mlle. Marie-Ève Grandmaison* pour son support et sa gestion de ce projet.

Je vaudrais aussi remercier les membres de ma famille, ma femme et mon père pour m'avoir soutenu tout au long des mes études.

CONCEPTION D'ALGORITHMES DE MISE EN TRAME SELON LA NORME IEEE 802.16E

MAMMOUH, Bouchaib

RÉSUMÉ

Les systèmes OFDMA ou WiMAX sont une alternative intéressante aux systèmes d'accès large bande filaire que soit le câble ou l'xDSL. En effet, la capacité des réseaux de transport est devenue très importante et la demande en services utilisant une large bande ne cesse de croître. En revanche, l'accès au réseau à un coût raisonnable n'a pu suivre cette évolution. C'est pourquoi les systèmes d'accès multiple OFDMA suscitent beaucoup d'intérêt pour l'industrie et aussi pour la recherche scientifique.

L'objectif principal de cette mémoire est d'augmenter les performances d'un système OFDMA au niveau du débit tout en assurant une meilleure qualité de service. Afin d'atteindre ce but, différentes méthodes peuvent s'avérer très utiles. Ces méthodes sont :

- Élaborer un ordonnancement efficace pour servir tous les usagers tout en respectant la qualité de service exigée et en augmentent le débit du réseau.
- Gérer efficacement les files d'attente au niveau de la station de base, BS, et la station usager, SS.
- Construire et remplir d'une manière optimale la trame à transmettre de telle sorte à respecter pleinement l'ordre d'arrivée des paquets et à réduire au maximum les espaces «*slots*» qui resteront vides dans la trame.
- Augmenter la performance du système en utilisant la technologie des antennes adaptatives.

Le mémoire présenté dans ce rapport décrit une seule méthode parmi les méthodes citées ci-haut à savoir le remplissage optimal de la trame tout en respectant l'ordre des arrivées des paquets fourni par l'ordonnateur «*Scheduler*».

La simulation a été faite sur une base de génération des tailles des paquets selon la loi de Poisson.

Deux algorithmes seront présentés dans ce mémoire afin de choisir la meilleure option. Le premier algorithme est basé sur la combinaison des différents Bursts pour déterminer de quelle manière chaque Burst sera inséré dans la trame. Le deuxième algorithme est basé sur le principe du plus grand en premier, c'est-à-dire que la trame sera remplie en premier lieu par le Burst qui occupera le plus grand nombre de slots puis le suivant jusqu'au remplissage final de la trame.

La performance de ces algorithmes sera évaluée sur la base du taux de remplissage de la trame ainsi que le temps d'exécution de chaque algorithme.

DESIGN OF IEEE 802.16E FRAMING ALGORITHMS

MAMMOUH, Bouchaib

ABSTRACT

The OFDMA or WiMAX systems are an interesting alternative to the broadband access systems which are the cables or the xDSL. Indeed, the transport networks capacity becomes very important and the demand for services using broadband grows rapidly. On the other hand, the access to the network at a reasonable cost could not follow this evolution. This is why the OFDMA systems cause much interest for industry and also for scientific research.

The principal objective of this thesis is to increase the performances of the OFDMA system on the level of the throughput while ensuring a better quality of service. In order to achieve this goal, various methods can prove very useful. These methods are:

- Design an effective scheduling algorithm to serve all the users while respecting the quality of service required and increase the throughput of the network by it.
- Manage effectively the queues on the level of the base station, BS, and the user station, SS.
- Build and fill with an optimal manner the frame to be transmitted by fully respecting the order of arrival of the packets and reducing the spaces, slots, which will remain empty in the frame.
- Increase the performance of the system by using the adaptive antennas technology.

The thesis presented here describes only one method among the mentioned methods above. This method is the optimal filling of the frame while respecting the order of the arrivals of the packets provided by the scheduler.

The simulation was made on a basis of generation of the sizes of the packets according to a Poisson law.

Two algorithms will be presented in this thesis in order to choose the best option. The first algorithm is based on the combination of different bursts to determine which combinations

will ensure the optimal filling of the frame. The second algorithm is based on the principle of largest first, i.e. the frame will be filled initially by burst which will occupy the greatest number of slots then the following until the final filling of the frame.

The performance of these algorithms will be evaluated on the basis of rate of filling of the frame as well as the execution time of each algorithm.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AAS	Adaptive Antenna System also Advanced Antenna System
ACK	Acknowledge
AES	Advanced Encryption Standard
AMC	Adaptive Modulation and Coding
ARQ	Automatic Repeat reQuest
BE	Best Effort
CC	Convolutional Code
CCM	Counter with Cipher-block chaining Message authentication code
CID	Connection Identification
CINR	Carrier to Interference + Noise Ratio
CMAC	block Cipher-based Message Authentication Code
CP	Cyclic Prefix
CPS	Common Part Sublayer
CQI	Channel Quality Indicator
CRC	Cyclic Redundancy Check
CS	Convergence Sublayer
CTC	Convolutional Turbo Code
DHCP	Dynamic Host Configuration Protocol
DL	Downlink
DOCSIS	Data Over Cable Service Interface Specification
DSL	Digital Subscriber Line
EAP	Extensible Authentication Protocol

ErtPS	Extended Real-Time Packet Service
FCH	Frame Control Header
FDD	Frequency Division Duplex
FFT	Fast Fourier Transform
FTP	File Transfer Protocol
FUSC	Fully Used Sub-Channel
GHz	GégaHertz
HARQ	Hybrid Automatic Repeat reQuest
HMAC	keyed Hash Message Authentication Code
HTTP	Hyper Text Transfer Protocol
Hz	Hertz
IE	Information Element
IEEE	Institute of Electrical and Electronics Engineers
IEFT	Internet Engineering Task Force
IFFT	Inverse Fast Fourier Transform
IP	Internet Protocol
ISI	Inter-Symbol Interference
LDPC	Low-Density-Parity-Check
LOS	Line of Sight
MAC	Media Access Control
MAN	Metropolitan Area Network
MAP	Media Access Protocol
Mbps	Mégabits par seconde

MD	Message Digest
MIMO	Multiple Input Multiple Output
ms	milliseconde
MS	Mobile Station
MHz	Mégahertz
NLOS	Non Line-of-Sight
nrtPS	Non-Real-Time Packet Service
OFDM	Orthogonal Frequency Division Multiplex
OFDMA	Orthogonal Frequency Division Multiple Access
PDU	Protocol Data Unit
PKM	Public Key Management
PS	Privacy Sublayer
PUSC	Partially Used Sub-Channel
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
QoS	Quality of service
RTG	Receive/transmit Transition Gap
rtPS	Real-Time Packet Service
SIM	Subscriber Identify Module
SINR	Signal to Interference + Noise Ratio
SMS	Short Message Service
SNIR	Signal to Noise + Interference Ratio

SNR	Signal to Noise Ratio
S-OFDMA	Scalable Orthogonal Frequency Division Multiple Access
SS	Subscriber Station
STC	Space Time Coding
TDD	Time Division Duplex
TEK	Traffic Encryption Key
TFTP	Trivial File Transfer Protocol
TTG	Transmit/receive Transition Gap
UGS	Unsolicited Grant Service
UL	Uplink
USIM	Universal Subscriber Identify Module
VoIP	Voice over Internet Protocol
WiMAX	Worldwide Interoperability for Microwave Access

INTRODUCTION

Situation du problème

De nos jours, l'utilisation des réseaux de télécommunications augmente constamment et la demande accrue de performance nous incite à développer de nouvelles solutions et architectures de réseaux. L'apparition de la norme IEEE 802.16e est le résultat de nombreuses années de recherche afin de combler la demande d'utilisation des multimédias, de la mobilité ainsi que les communications sans fil.

La norme IEEE 802.16, et spécifiquement l'IEEE 802.16e, définit la couche physique et la couche MAC [1, 2, 3]. Au niveau de la couche MAC, la norme présente la sous couche de convergence qui est responsable d'intégrer d'autres protocoles tel qu'IP, ATM et Ethernet. Aussi, ce standard présente la sous couche de la sécurité en définissant les protocoles qui devront être employés pour la transmission et l'échange des messages entre la station de base et les stations abonnées. Enfin, la définition de la sous couche commune est présentée par le standard IEEE802.16e. Cette sous couche est le cœur de la couche MAC, elle effectue les opérations de gestion de la puissance des stations abonnées, le service d'ordonnancement, la gestion des connections et le contrôle de la liaison sans fil. Au niveau de la couche physique, la norme présente la structure de la trame, les modes de la FFT, la structure du symbole OFDMA ainsi que la sous-canalisation. Aussi, la définition des techniques MIMO et AAS sont présentées par la norme. Cette norme a été élaborée pour des fréquences comprises dans les bandes 10-66 GHz et 2-11 GHz.

La couche MAC est composée de trois sous couches. La sous couche MAC CS «*Service Specific Convergence Sublayer*» qui joue le rôle d'interface avec les couches supérieures ou bien les systèmes externes. Elle a entre autre le rôle de classer les paquets selon leur provenance et leur destination afin de les répartir sur la bonne connexion MAC. La sous couche MAC CPS «*Common Part Sublayer*» est le coeur de la couche MAC, à savoir qu'elle s'occupe de l'allocation de bande, l'établissement de la maintenance des connexions. Elle reçoit des paquets provenant de la sous couche MAC CS que cette dernière aura classée en

connexions de différentes qualités de service. La sous couche MAC PS «*Privacy Sublayer*» est la couche de sécurité qui s'occupe du cryptage des données, de l'échange des clefs etc.

La couche physique est basée principalement sur la modulation OFDM. Cette technique de modulation divise la largeur de bande disponible en plusieurs sous bandes. Le flux de données est divisé alors en plusieurs sous flux parallèles et chaque sous flux de données est modulé et transmis sur une sous porteuse orthogonale séparée. La durée de chaque symbole est alors plus grande, ce qui augmente la robustesse de l'OFDM. Aussi, l'introduction du préfixe cyclique «Cyclic Prefix», CP, peut complètement éliminer les interférences inter-symboles, ISI, tant que le préfixe cyclique est plus grand que le délai de propagation du canal.

L'interconnexion ou la communication entre la couche physique et la couche MAC nécessite des mécanismes et des algorithmes performants qui permettent à la couche MAC de bien gérer la couche physique ainsi que de recueillir l'information disponible à la couche physique. Parmi ces algorithmes, le constructeur de la trame, indique à la couche physique comment la trame doit être envoyée et son contenu. Cet algorithme doit être logé dans la couche MAC et doit permettre un traitement rapide et efficace des paquets reçus tout en respectant l'ordre des arrivés des paquets tout en minimisant l'espace vide dans une trame.

Objectifs du mémoire

Cette recherche vise essentiellement à développer des algorithmes capables de construire efficacement la trame selon la norme IEEE 802.16e ou WiMAX. Les algorithmes doivent prendre en considération le fait que la trame en question doit être construite sur un plan bidimensionnel, selon l'axe des fréquences et l'axe du temps. De plus, certaines régions de la trame doivent être rectangulaires comme les rafales de données de la voie descendante. Aussi, ces algorithmes seront évalués selon leurs performances en temps d'exécution ou de remplissage de la trame, le taux de remplissage de la trame et la taille maximale du paquet à insérer dans la trame.

Mise en contexte

Tel que mentionné précédemment, cette recherche a pour objectif de développer des algorithmes de mise en trame selon la norme IEEE 802.16e ou WiMAX mobile. Le développement des ces algorithmes prend en considération l'existence préalable de l'ordonnanceur. Cet ordonnanceur fournit les paquets classés à ces algorithmes selon les demandes en QoS des différentes stations abonnées.

Il est très important de mentionner que la norme IEEE 802.16e a été ratifiée récemment, ce qui rend les résultats de recherche reliées à cette norme peu accessibles. En effet, les recherches sur les algorithmes de mise en trame selon la norme IEEE 802.16e sont inexistantes, à notre connaissance, ce travail est le premier en son genre et il peut être utilisé comme une base pour les recherches futures concernant la mise en trame des paquets selon la norme IEEE 802.16e.

Par contre, certains algorithmes comme celui du sac à dos «*knapsack algorithm*» peuvent être d'une grande utilité pour développer un algorithme de mise en trame selon la norme IEEE 802.16e. Ceci est dû au fait que la trame possède une forme rectangulaire et que cette trame doit être remplie par des petites formes rectangulaires selon le nombre de rafales de données disponibles.

Avec ces contraintes, l'évaluation comparative des algorithmes présentés dans ce mémoire n'est pas évidente à faire. Mais la performance de chacun des algorithmes développés est mesurée et quantifiée selon les critères de taux de remplissage de la trame ainsi que le temps de traitement pour le remplissage de cette trame.

Organisation des chapitres

Ce mémoire est réparti en 4 chapitres. Le premier chapitre fait état des normes qui régissent les systèmes de communications sans fil WiMAX. Il précise les revues de la norme IEEE 806.16e et l'OFDMA. Le deuxième chapitre traite en détail les normes et les conventions entourant la structure de la trame selon la norme IEEE 802.16e et le WiMAX mobile. Au chapitre 3, l'objectif principal est de trouver et d'évaluer des algorithmes afin de remplir une

trame de forme rectangulaire avec des données qui, en général, n'occupent pas toujours une région rectangulaire. Selon l'arrivée des rafales, la trame doit être construite en temps réel tout en minimisant les tranches qui seront laissées vides.

CHAPITRE 1

Revue de la norme IEEE 806.16e et l'OFDMA

Introduction

La technologie de WiMAX, basée sur la norme IEEE 802.16-2004, a rapidement prouvé qu'elle jouera un rôle principal dans les réseaux fixes sans fil à large bande. Incontestablement, le WiMAX fixe, basé sur la norme IEEE 802.16-2004 [1], s'est avéré être une alternative rentable par rapport aux services de câble et de DSL. En décembre 2005, l'IEEE a ratifié l'amendement 802.16e à la norme 802.16 [2]. Cet amendement ajoute les dispositifs et les attributs à la norme qui sont nécessaires pour soutenir la mobilité. Le forum WiMAX définit maintenant des profils d'exécution et de certification de système basés sur l'amendement mobile d'IEEE 802.16e et il définit l'architecture du réseau nécessaire pour mettre en application un réseau bout à bout du WiMAX mobile.

Le WiMAX mobile est une solution large bande qui permet la convergence des réseaux fixe et mobile large bande à travers une technologie commune d'accès par radio large bande commun et une architecture de réseau flexible. L'interface sans-fil du WiMAX mobile adopte la technique de l'accès multiple par division orthogonale de la fréquence, OFDMA, pour une performance des multi trajets améliorée dans des environnements obstrués, NLOS. L'OFDMA extensible a été introduite dans l'amendement IEEE 802.16e afin de supporter les canaux de 1.25 à 20 MHz. Le système des profils du WiMAX mobile permet aux systèmes mobiles d'être configurés en se basant sur les options communes. Ceci assure une fonctionnalité de base pour les terminaux et les stations de base qui sont interopérables.

Quelques éléments du profil de la station de base sont spécifiés comme étant optionnels afin de fournir une flexibilité additionnelle pour un déploiement basé sur des scénarios de déploiement spécifiques qui requièrent différentes configurations telles qu'une capacité optimisée ou une convergence optimisée.

Les systèmes WiMAX offrent l'extensibilité dans la technologie de l'accès radio et l'architecture du réseau ce qui peut donner une grande flexibilité dans les options de déploiement des réseaux et le service offert. Les quelques options supportées par le WiMAX mobile sont :

- ***Taux de transfert de données élevé :*** Avec l'intégration de la technologie entrée multiple sortie multiple, MIMO, une modulation et un codage adaptatif avancé permettront à la technologie WiMAX mobile de supporter un taux de transfert de données jusqu'à 63 Mbps en voie montante et jusqu'à 28 Mbps en voie descendante par secteur avec un canal de 10 MHz.

- ***La qualité de service, QoS :*** La composante principale et fondamentale dans l'architecture de la couche MAC de la norme IEEE 802.16e est la qualité de service, QoS. Cette fonction est importante pour supporter les services comme la voie sur IP, les applications multimédia etc.

- ***L'extensibilité :*** Les contraintes reliées à l'obtention des licences pour des fréquences dans différentes régions du monde incitent à mettre en œuvre un système flexible qui peut être fonctionnel avec plusieurs types de largeur de bande de 1.25 MHz à 20 MHz. Cela permet aussi d'offrir un accès Internet à prix abordable dans certaines régions rurales en plus d'augmenter la capacité d'accès dans les régions métropolitaine.

- ***La sécurité :*** Le WiMAX mobile fournit les options de sécurité les plus avancées avec notamment le protocole d'authentification extensible, EAP, le chiffrement authentifié à l'aide du standard de chiffrement avancé, AES, et les mécanismes de protection des messages de contrôle utilisant la technique du code d'authentification de message à base de fonction de chiffrement, CMAC, et à base de fonction de hachage, HMAC.

- ***La mobilité :*** La technologie mobile WiMAX supporte un mécanisme optimisé de transfert intercellulaire avec une latence inférieure à 50 ms afin d'assurer l'opération des applications en temps réel comme la voix sur IP, VoIP, sans dégradation du

service. Le mécanisme de gestion des clés flexibles permet d'assurer le maintien de la sécurité durant le transfert intercellulaire.

1.1 Description de la couche physique

1.1.1 Principe de l'OFDMA

La modulation OFDM «Orthogonal Frequency Division Multiplexing» est une technique de multiplexage qui divise la largeur de bande disponible en plusieurs sous canaux comme l'indique la figure 1.1 [6,7].

Dans un système OFDM, le flux des données de l'entrée est divisé en plusieurs sous blocs de taux de transmission réduit, ce qui augmente la durée du symbole et chaque sous bloc de données est modulé et transmis sur une sous-porteuse orthogonale distincte. L'augmentation de la durée du symbole améliore la robustesse de l'OFDM en ce qui concerne le délai de propagation.

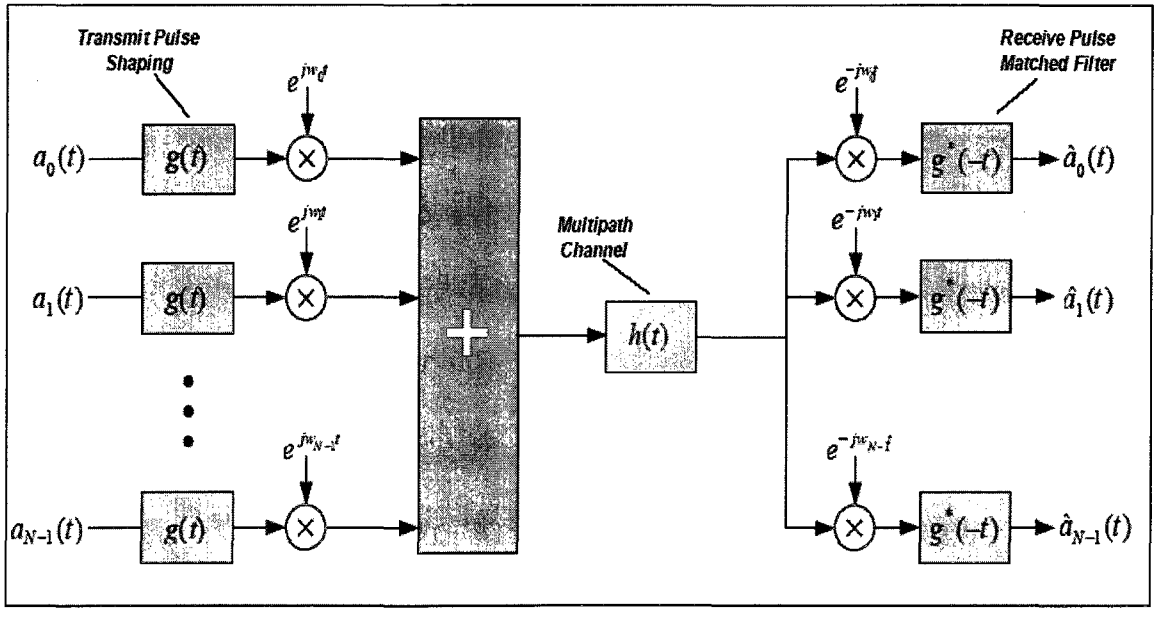

Figure 1.1 ***L'architecture de base d'un système OFDMA.***[1]

[1] Tirée de l'article «Mobile WiMAX – Part I : A Technical Overview and Performance Evaluation»

De plus, l'introduction du préfixe cyclique «Cyclic Prefix», CP, peut complètement éliminer le phénomène d'interférence inter-symbole, ISI, tant que la durée du CP est plus longue que le délai de propagation du canal. Le préfixe cyclique est une répétition des derniers échantillons de la portion du bloc des données et il est inséré au début des données utiles comme illustré par la figure 1.2. Le CP prévient les interférences inter-blocs et permet une égalisation fréquentielle moins complexe. L'ajout du préfixe entraîne un ajout de l'en-tête ce qui réduit l'efficacité spectrale. Tandis que le CP réduit l'efficacité spectrale, son impact est similaire au facteur de décroissance «roll-off factor» des filtres en cosinus surélevé. Une grande fraction de la largeur de bande assignée au canal peut être utilisée pour la transmission de données, ce qui aide à diminuer la perte de l'efficacité spectrale due au préfixe cyclique CP.

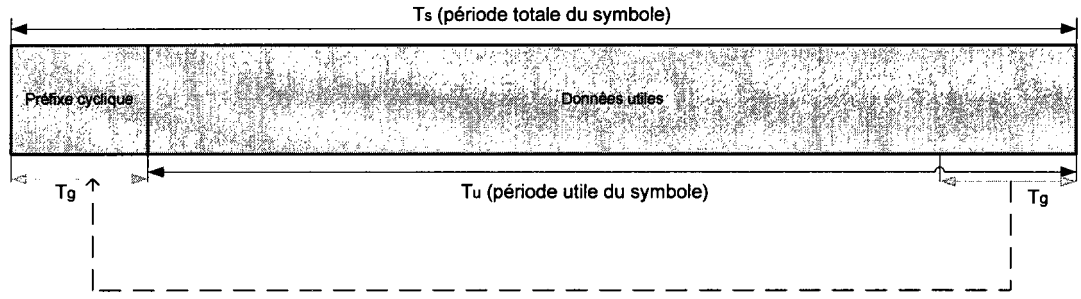

Figure 1.2 ***L'insertion du préfixe cyclique, CP.***

La technique de modulation OFDM exploite la diversité fréquentielle des canaux multi trajets en codant et en entrelaçant l'information à travers de multiples bandes de fréquences. La modulation OFDM peut être réalisée à l'aide de la transformée de Fourrier inverse, IFFT, qui permet d'avoir un grand nombre d'ondes sous-porteuses, jusqu'à 2048, avec une complexité faible. Dans les systèmes OFDM, les ressources sont disponibles dans les domaines temporels, symboles OFDM, et dans les domaines fréquentiels, sous-porteuses. Les ressources temporelles et fréquentielles peuvent être organisées en sous-canaux pour une allocation aux usagers individuels. L'OFDMA, «Orthogonal Frequency Division Multiple Access», est une technique de multiplexage à accès multiple qui fournit une opération de multiplexage des flux de données qui proviennent de multiples usagers. L'OFDMA est une technique d'accès multiple pour les systèmes OFDM.

1.1.2 Structure du symbole OFDMA et de la sous canalisation

La structure du symbole OFDMA consiste en trois types de sous-porteuses telle qu'illustré à la figure 1.3. Les trois sous-porteuses en question sont :

- Celles pour la transmission des données.
- Celles des pilotes pour l'estimation et la synchronisation.
- Les sous-porteuses nulles qui ne servent pas à la transmission mais sont plutôt utilisées comme bandes de garde.

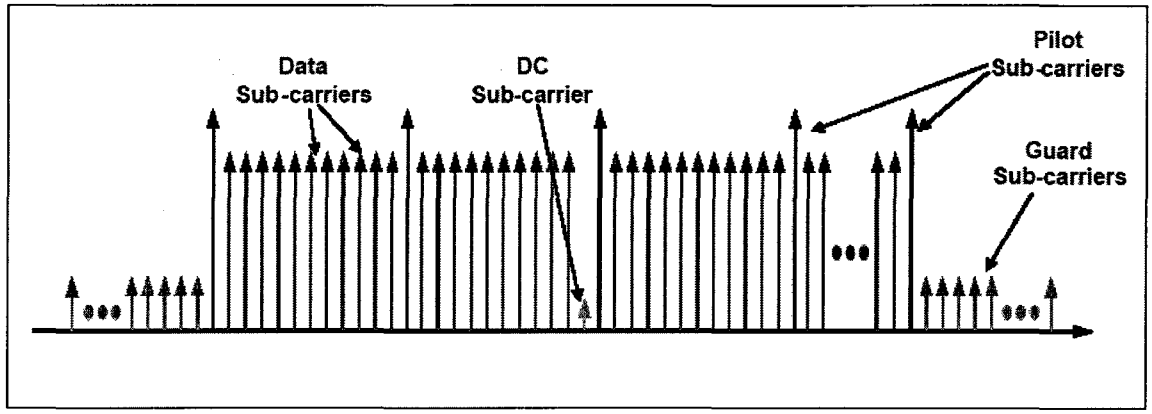

Figure 1.3 ***La structure d'une sous porteuse OFDMA.***[2]

Les sous-porteuses actives, données et pilotes, sont groupées en sous ensembles appelées sous-canaux. La couche physique d'un système OFDMA supporte la sous canalisation dans les deux voies, descendante, DL, et montante, UL. L'unité minimum fréquence-temps de la sous canalisation est une tranche, qui est égale à 48 sous-porteuses de données [8].

Il y a deux types de permutation de la sous-porteuse pour la sous canalisation : La permutation distribuée et la permutation adjacente. La permutation par diversité distribue les sous-porteuses pseudo aléatoirement pour former un sous-canal. La permutation par diversité inclut le mode des sous-porteuses entièrement utilisées de la voie descendante, DL-FUSC «Fully Used Sub-Carrier», le mode des sous-porteuses partiellement utilisées de la

[2] Tirée de l'article «Mobile WiMAX – Part I : A Technical Overview and Performance Evaluation»

voie descendante, DL-PUSC «Partially Used Sub-Carrier» et le mode des sous-porteuses partiellement utilisées de la voie montante, UL-PUSC, ainsi que d'autres options de permutations additionnelles. Avec le mode DL-PUSC, pour chaque paire de symboles OFDMA, les sous-porteuses disponibles sont groupées en blocs contenant 14 sous-porteuses adjacentes par symbole, avec des allocations des données et des pilotes dans chaque bloc des symboles de parité paire et impaire comme indiqué par la figure 1.4.

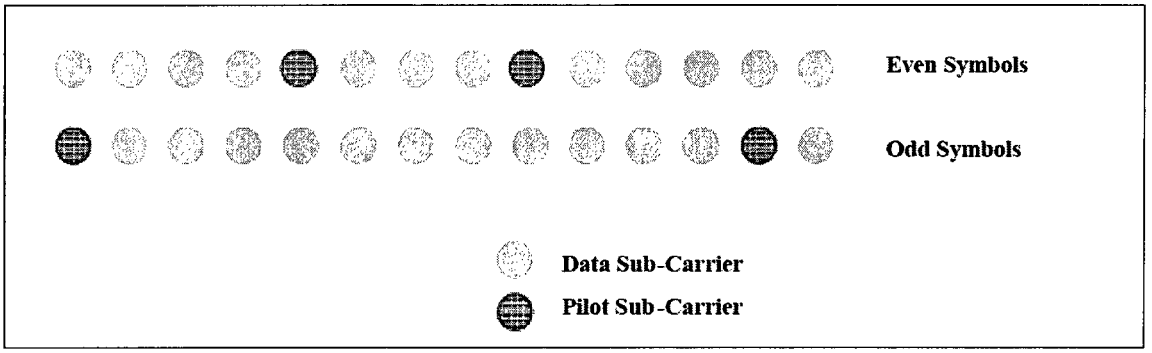

Figure 1.4 ***Le sous-canal DL en diversité fréquentielle.***[3]

Un mécanisme de réarrangement est utilisé pour former des groupes de blocs comme celui où chaque groupe est fait de blocs qui sont distribués à travers l'espace des sous-porteuses. Le sous-canal dans un groupe contient deux blocs et constitué de 48 sous-porteuses de données et 8 sous-porteuses de pilotes.

Par analogie à la structure des blocs pour la voie descendante, la structure de la tuile, «*Tile*», est définie pour le mode des sous-porteuses partiellement utilisées de la voie montante «*Uplink-Partially Used Sub-channel*», UL-PUSC, telle qu'illustrée à la figure 1.5.

[3] Tirée du standard «Part 16 : Air Interface for Fixed Broadband and Wireless Access Systems»

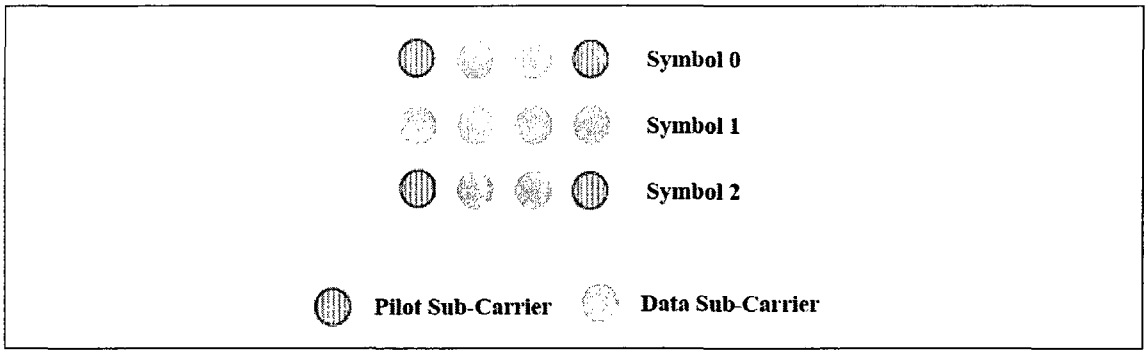

Figure 1.5 ***La structure de la tuile pour l'UL-PUSC.***[4]

L'espace des sous-porteuses disponibles est divisé en plusieurs tuiles et chaque six tuiles forment une tranche. Une tranche est composée de 48 sous-porteuses de données et de 24 sous-porteuses de pilotes dans 3 symboles OFDM.

La permutation adjacente regroupe un bloc de sous-porteuses contiguës pour former un sous-canal contrairement à la permutation distribuée qui regroupe les sous-porteuses pseudo aléatoirement pour former un sous-canal.

1.1.3 OFDMA extensible

La norme IEEE 802.16e Wireless MAN OFDMA est basée sur le concept de l'OFDMA extensible, S-OFDMA. Le S-OFDMA supporte plusieurs largeurs de bande pour adresser d'une manière flexible les besoins variables du spectre et de mode d'utilisation. L'extensibilité est supportée en ajustant la taille de la transformée de Fourier rapide, FFT, tout en fixant la distance entre les sous-porteuses à 10.94 kHz. L'unité ressource, qui est la largeur de bande de la sous-porteuse et la durée du symbole, est fixe. Cela rend l'impact des couches supérieures, la couche MAC par exemple, minimal lorsque la largeur de bande ou le nombre de point FFT varie. Les paramètres du S-OFDMA sont listés dans le tableau 1.1.

---

[4] Tirée du standard «Part 16 : Air Interface for Fixed Broadband and Wireless Access Systems»

Tableau 1.1

Les paramètres de l'extensibilité de l'OFDMA

Paramètres	Valeurs			
Largeur de bande (MHz)	1.25	5	10	20
Fréquence d'échantillonnage (MHz)	1.4	5.6	11.2	22.4
Taille FFT (*NFFT*)	128	512	1024	2048
Nombre de sous-canaux	2	8	16	32
Espacement entre les fréquences des sous-porteuses	10.94 kHz			
Temps utile du symbole (µs)	91.4			
Temps de garde (µs)	11.4			
Durée du symbole OFDMA (µs)	102.9			
Nombre de symboles OFDMA (Trame de 5 ms)	48			

1.1.4 Structure de la trame OFDMA

La couche physique de la norme IEEE 802.16e supporte les modes de duplexage temporel, TDD, et fréquentiel, FDD. Cependant, la première version des profils, modulation et codage, de la certification WiMAX mobile inclut seulement le mode TDD. Le mode FDD sera inclus dans les prochaines versions. Le mode FDD sera considéré par le forum WiMAX pour satisfaire des marchés spécifiques où les réglementations locales interdisent l'utilisation du mode TDD. Pour remédier aux problèmes d'interférence, le mode TDD nécessite un système de synchronisation. Néanmoins, le mode TDD est préféré comme technique de duplexage pour les raisons suivantes :

- Le mode TDD permet l'ajustement du rapport de la liaison descendante sur la liaison montante, DL/UL, pour supporter efficacement le trafic DL/UL asymétrique. Tandis qu'en mode FDD, les liaisons descendantes et montantes ont toujours des largeurs de bande fixes et généralement égales.

- Le mode TDD assure une réciprocité de canal pour bien supporter l'adaptation du lien, le MIMO et autres technologies avancées de l'antenne.

- Contrairement au mode FDD, qui requiert une paire de canaux, le mode TDD requiert seulement un canal simple pour la liaison montante et la liaison descendante fournissant une grande flexibilité d'adaptation aux diverses attributions globales de spectre. De plus, les conceptions des émetteurs et des récepteurs pour l'implémentation du mode TDD sont moins complexes et donc moins coûteuses.

La figure 1.6 illustre la structure de la trame OFDMA pour l'implémentation en mode TDD. Chaque trame est divisée en sous-trames UL et DL séparées par des espaces de garde TTG et RTG pour prévenir les collisions de transmissions des sous-trames UL et DL. Dans une trame, les informations de contrôle sont utilisées pour assurer une opération optimale du système :

- ***Préambule*** : Ce message, utilisé pour la synchronisation, est le premier symbole OFDMA de la trame.

- ***FCH*** «Frame Control Header» : Le message FCH suit le préambule. Il fournit l'information de la configuration de la trame comme la longueur du message MAP, le mécanisme de codage et les sous canaux utilisable.

- ***DL-MAP et UL-MAP*** : Ils fournissent l'allocation des sous canaux et autres informations de contrôle pour les sous-trames DL et UL respectivement.

- ***UL Ranging*** : Le sous-canal de signalisation UL est alloué aux stations mobiles pour faire les ajustements de la puissance, de la fréquence et du temps ainsi que les demandes de la largeur de bande.

- ***UL CQICH*** : Le canal UL CQICH est alloué à la station mobile pour le retour de l'information sur l'état du canal.

- ***UL ACK*** : L'UL ACK est alloué à la station mobile pour le retour des acquittements des messages de la demande de répétition automatique hybride, HARQ, de la voie descendante.

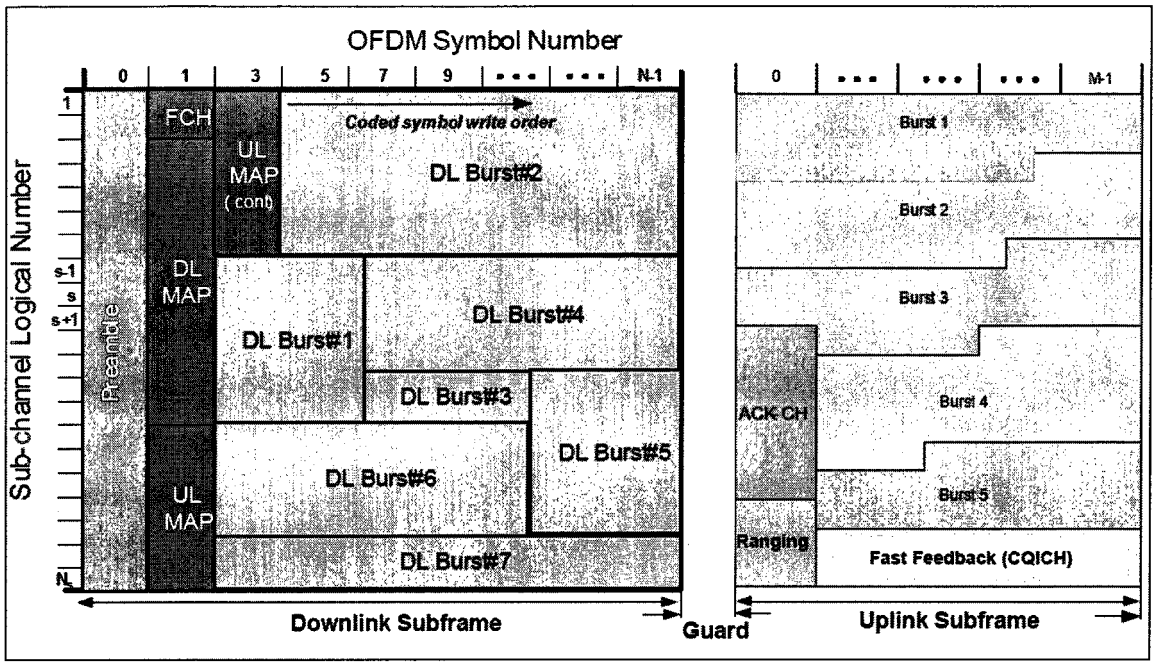

Figure 1.6 ***La structure de la trame WiMAX OFDMA.***[5]

1.1.5 Autres options avancées de la couche physique

La modulation et le codage adaptatifs, AMC, la demande de répétition automatique hybride, HARQ, et la rétroaction rapide du canal, CQICH, ont été introduits dans les systèmes WiMAX mobile pour améliorer la couverture et la capacité dans les applications mobiles.

Le support des modulations QPSK, 16QAM et 64QAM est obligatoire dans la voie descendante avec le WiMAX mobile. Dans la voie montante, la modulation 64QAM est optionnelle. Le codage conventionnel, CC, et le codage conventionnel Turbo, CTC, avec un taux de codage variable et un codage répétitif sont supportés. Le codage en bloc Turbo et le code de contrôle de parité de faible densité, LDPC, sont supportés comme des options. Le tableau 1.2 résume la modulation et le codage supportés dans le WiMAX mobile.

[5] Tirée de l'article «Mobile WiMAX – Part I : A Technical Overview and Performance Evaluation»

Tableau 1.2

Les modulations et les codages supportés

		Liaison descendante DL	Liaison montante UL
Modulation		QPSK, 16QAM, 64QAM	QPSK, 16QAM, *64QAM*
Taux de codage	CC	1/2, 2/3, 3/4, 5/6	1/2, 2/3, *5/6*
	CTC	1/2, 2/3, 3/4, 5/6	1/2, 2/3, *5/6*
	Répétition	x2, x4, x6	x2, x4, x6

La combinaison de différentes modulations et différents taux de codage fournit une résolution fine des débits telle qu'elle est illustrée au tableau 1.3 qui montre les débits pour les canaux de 5 MHz et 10 MHz avec des sous-canaux PUSC. La durée de la trame est de 5 ms. Chaque trame a 48 symboles OFDM, avec 44 symboles OFDM disponibles pour la transmission des données.

Tableau 1.3

Les débits de la couche physique avec le sous-canal PUSC

Paramètres		DL	UL	DL	UL
Largeur de bande (MHz)		5		10	
Taille FFT		512		1024	
Sous-porteuses nulles		92	104	184	184
Sous-porteuses de pilotes		60	136	120	280
Sous-porteuses de données		360	272	720	560
Sous-canaux		15	17	30	35
Période du symbole (µs)		102.9			
Durée de la trame (ms)		5			
Symboles OFDMA/Trame		48			
Symboles OFDM de données		44			
Modulation	Taux de codage	Canal de 5 MHz		Canal de 10 MHz	
		Débit en DL (Mbps)	Débit en UL (Mbps)	Débit en DL (Mbps)	Débit en UL (Mbps)
QPSK	1/2 CTC, 6x	0.53	0.38	1.06	0.78
	1/2 CTC, 4x	0.79	0.57	1.58	1.18
	1/2 CTC, 2x	1.58	1.14	3.17	2.35
	1/2 CTC, 1x	3.17	2.28	6.34	4.70
	3/4 CTC	4.75	3.43	9.50	7.06
16QAM	1/2 CTC	6.34	4.57	12.07	9.41
	3/4 CTC	9.50	6.85	19.01	14.11
64QAM	1/2 CTC	9.50	6.85	19.01	14.11
	2/3 CTC	12.67	9.14	26.34	18.82
	3/4 CTC	14.26	10.28	28.51	21.17
	5/6 CTC	15.84	11.42	31.68	23.52

Le planificateur «Scheduler» de la station de base détermine le débit approprié, profil de la rafale, pour chaque allocation de la rafale en se basant sur la taille du tampon, les conditions

de propagation du canal, etc. L'indicateur de la qualité du canal, CQI, est utilisé pour fournir les informations sur l'état du canal à partir de la station abonnée au planificateur de la station de base. Ces informations peuvent être convoyées par le message CQICH incluant le rapport de la porteuse sur l'interférence plus le bruit, CINR, le mode de sélection de la technique entrée multiple sortie multiple, MIMO, et le choix sélectif de la fréquence du sous canal. Avec le mode TDD, l'adaptation du lien peut prendre aussi de l'avantage de la réciprocité du canal pour fournir une mesure plus précise de la condition du canal.

Le message HARQ est supporté par le WiMAX mobile. Le HARQ est activé en utilisant le mécanisme du protocole "N-channel Stop and Wait" qui fournit une réponse rapide aux erreurs des paquets et améliore la couverture de bord des cellules. Le HARQ combiné avec le CQICH et l'AMC fournit une adaptation robuste du lien dans un environnement mobile pour des vitesses aux alentours de 120 km/h.

1.2 Description de la couche MAC

Introduction

La norme IEEE 802.16 a été développée dès le début pour la livraison des services large bande incluant la voix, les données et le vidéo. La couche MAC est essentiellement basée sur celle de la norme DOCSIS et elle peut supporter un très grand trafic de données avec une demande crête de débits très élevée [9]. Tandis que le même canal peut supporter simultanément le flux vidéo et le trafic de la voix, les ressources allouées à un terminal par le planificateur de la couche MAC peut varier d'un seul intervalle de temps à une trame entière. Ceci donne une très grande marge dynamique de débit à un terminal spécifique et ce, à n'importe quel moment. En outre, l'information de l'allocation des ressources est convoyée par les massages MAP au début de chaque trame, le planificateur peut effectivement changer l'allocation des ressources de trame en trame pour s'adapter à la nature en rafale du trafic.

1.2.1 Adressage et connections

Chaque station abonnée doit avoir une adresse MAC de 48 bits, tel que défini dans la norme IEEE 802.16e. Cette adresse définit d'une façon unique la station abonnée parmi un

ensemble de possibilités de types de vendeurs et d'équipements [1]. Elle est utilisée durant la première tentative de connexion et aussi comme une partie du processus d'authentification par lequel la station de base et la station abonnée vérifient leurs identités respectives.

Les identificateurs des connexions, CID, sont identifiés par 16 bits. À l'initialisation de la station abonnée, deux paires de connexions de gestion dans les deux sens, une connexion de base et une connexion primaire, doivent être établies entre la station de base et la station abonnée. Une troisième connexion, de gestion, peut être optionnellement générée. Les trois paires de connexions reflètent le fait qu'il y a trois différents niveaux de la qualité de service, QoS, pour la gestion du trafic entre la station abonnée et la station de base. La connexion de base est utilisée par la couche MAC de la station de base et par la couche MAC de la station abonnée pour échanger des messages de gestion courts et urgents du MAC. La connexion primaire de gestion est utilisée par la couche MAC de la station de base et par la couche MAC de la station abonnée pour échanger des messages de gestion longs et tolérants au délai du MAC. La connexion secondaire de gestion est utilisée par la station de base et la station abonnée afin de transférer les messages tolérants au délai, DHCP, TFTP, etc.

1.2.2 Formats des paquets

Les unités de données de protocole de la couche MAC, PDU, doivent être sous la forme illustrée à la figure 1.7 [1]. Chaque PDU doit commencer avec un en-tête générique du MAC de longueur fixe. L'en-tête peut être suivi par les données utiles du PDU MAC. Les informations des données utiles peuvent varier en longueur, permettant au PDU d'avoir une longueur variable. Ceci permet à la couche MAC d'accommoder différents types de trafic des couches supérieures sans connaître les formats ou les patrons de bit de ces messages.

Le PDU MAC peut contenir un système de contrôle de redondance cyclique, CRC.L'implémentation du CRC est optionnelle.

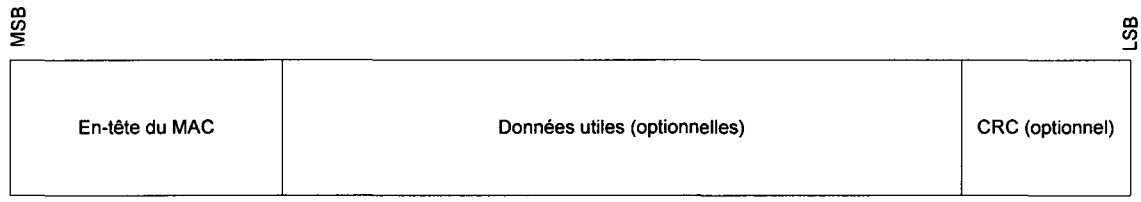

Figure 1.7 ***Le format du paquet de données de la couche MAC.***

1.2.3 Qualité de service

Avec un lien sans-fil rapide, une capacité symétrique DL/UL, une granularité fine des ressources et un mécanisme flexible d'allocation des ressources, le WiMAX mobile peut obtenir une qualité de service, QoS, acceptable pour un grand nombre d'applications et de services de données. À la couche MAC, la qualité de service est fournie à travers des chemins de service telle qu'illustré à la figure 1.8 [3]. C'est un chemin unidirectionnel de paquets qui est fourni avec un ensemble particulier des paramètres QoS. Avant de fournir un certain type de service, la station de base et le terminal usager établissent, en premier lieu, un lien logique unidirectionnel entre leurs couches MAC.

La couche MAC associe les paquets qui traversent son interface à un service de flux qui sera délivré à travers la connexion. Les paramètres de la QoS associés au service de flux définissant l'ordre de transmission dans l'interface sans-fil. La QoS orientée connexion peut fournir un contrôle accru à travers cette interface. Les paramètres du service de flux peuvent être gérés dynamiquement à travers les messages MAC pour accommoder la demande dynamique des services. Le WiMAX mobile supporte une grande variété de services de données et d'applications avec différentes conditions de la qualité de service. Le tableau 1.5 décrit l'ensemble des combinaisons de ces services et applications.

Tableau 1.4

Les applications et la qualité de service du WiMAX Mobile

Catégorie de la QoS	Applications	Spécifications de la QoS
rtPS Service en temps réel de paquet	contenu audio et vidéo	• Débit minimum réservé • Débit maximum soutenu • Tolérance maximum du temps de latence • Priorité du trafic
ErtPS Service en temps réel étendu de paquet	La voix avec la détection de l'activité (VoIP)	• Débit minimum réservé • Débit maximum soutenu • Tolérance maximum du temps de latence • Tolérance à la gigue • Priorité du trafic
nrtPS Service non en temps réel de paquet	Protocole de transfert de fichiers (FTP)	• Débit minimum réservé • Débit maximum soutenu • Priorité du trafic
BE Service au mieux	Transfert de données, navigation sur le Web, etc.	• Débit maximum soutenu • Priorité du trafic

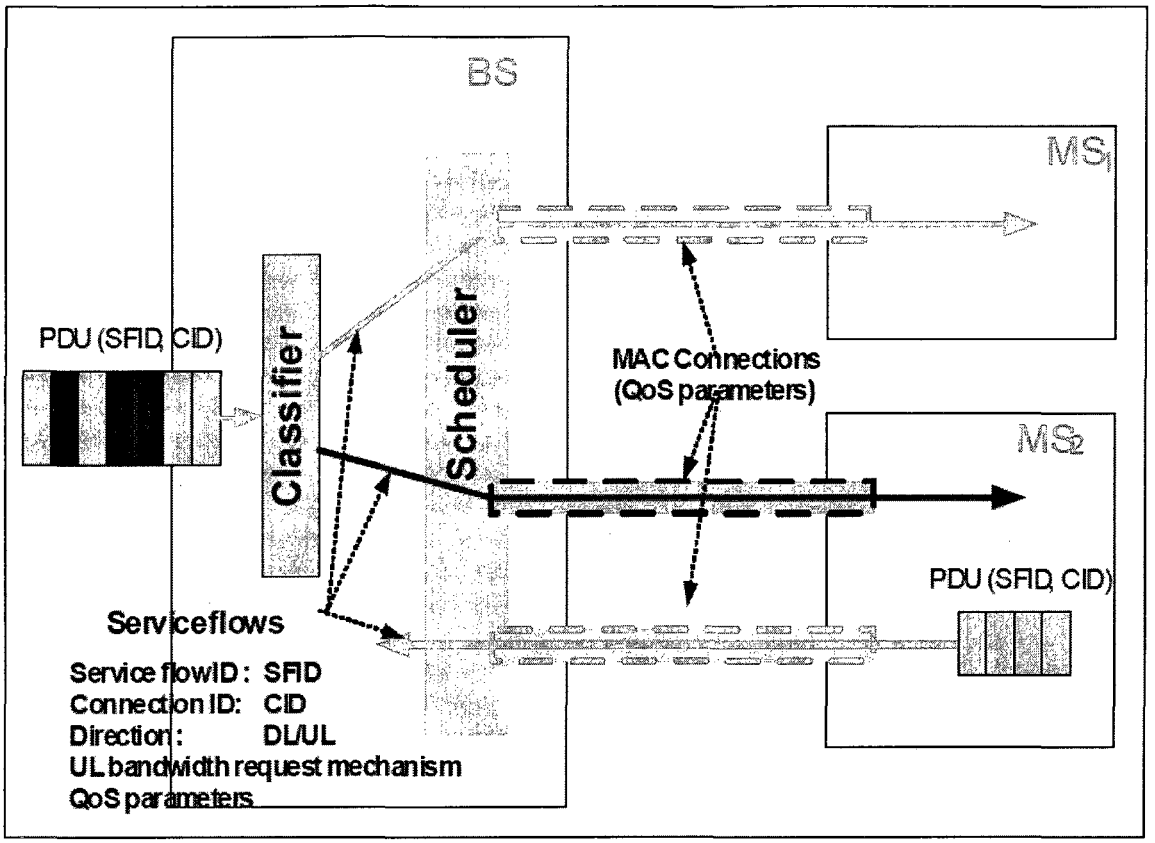

Figure 1.8 ***Le support QoS du WiMAX mobile.***[6]

1.2.4 Service d'ordonnancement

Le service de planification de la couche MAC est conçu pour livrer efficacement les services de données large bande incluant la voix, données et le vidéo à travers un canal large bande sans fil variant dans le temps.

Le service de planification de la couche MAC a les propriétés suivantes pour accommoder les services de données large bande :

- ***Planification de données rapide*** : Le planificateur doit efficacement allouer les ressources disponibles en réponse à une rafale du trafic de données et des conditions variantes dans le temps du canal. Le planificateur est situé dans chaque station de base pour permettre une réponse rapide aux besoins du trafic et aux conditions du

[6] Tirée de l'article «Mobile WiMAX – Part I : A Technical Overview and Performance Evaluation»

canal. Les paquets sont associés à un type de service qui définit les paramètres de la qualité de service dans la couche MAC, ce qui permet au planificateur de déterminer l'ordre de transmission de chaque paquet à travers l'interface sans fil. Le message CQICH fournit l'information rapide sur l'état du canal pour permettre au planificateur de bien choisir le codage et la modulation appropriés pour chaque allocation. La modulation/codage adaptative combinée avec le HARQ fournit une transmission robuste à travers un canal variant dans le temps.

- ***Planification pour les liaisons descendantes et montantes*** : Le service de planification est fourni pour les trafics des liaisons montantes, UL, et descendantes, DL. Pour que le planificateur fasse une allocation efficace des ressources et fournisse le niveau de la qualité de service désiré en DL et en UL, les stations abonnées doivent fournir les informations pertinentes et instantanées comme les conditions du trafic et les besoins en QoS.

- ***Allocation dynamique des ressources*** : La couche MAC supporte l'allocation des ressources dans le domaine temps-fréquence en DL et en UL pour chaque trame. L'allocation des ressources est énoncée dans les messages MAP à chaque début de trame. Donc, l'allocation des ressources peut changer à chaque trame et ce, en réponse à un changement de trafic et des conditions du canal. En plus, la quantité des ressources dans chaque allocation peut varier d'une tranche à une trame entière. L'allocation des ressources fine et granulaire permet une qualité de service supérieure pour les trafics de données.

- ***Orienté QoS*** : Le planificateur de la couche MAC manipule le transport des données sur une base de connexion par connexion. Chaque connexion est associée à un service simple de données avec un ensemble de paramètres de QoS qui mesurent les aspects de son comportement. Avec la capacité d'allouer dynamiquement les ressources en DL et en UL, le planificateur peut fournir un niveau de la qualité de service supérieur pour les trafics en DL et en UL.

- ***Planificateur sélectif de fréquence*** : Le planificateur peut opérer sur différents types de sous-canaux. Pour la diversité en fréquence comme pour la permutation PUSC, où les sous-porteuses dans les sous-canaux sont distribuées pseudo aléatoirement sur la largeur de bande, les sous-canaux sont de qualité similaire. La planification en diversité de fréquence peut supporter la qualité de service avec une granularité fine et une planification flexible de ressource en temps-fréquence. Avec la permutation adjacente comme la permutation AMC, les sous-canaux peuvent subir différentes atténuations. La planification par sélection de fréquence peut allouer les usagers mobiles à leurs sous-canaux les plus forts. Ce type de planification peut améliorer la capacité du système avec un ajout modéré d'en-tête CQI en voie montante, UL [10].

1.2.5 La gestion de la mobilité

La durée de vie de la batterie et le transfert intercellulaire sont deux questions critiques pour les applications mobiles. Le WiMAX mobile supporte le mode en veille «sleeping mode» et le mode au repos «Idle» pour permettre une opération de la puissance efficace de la station mobile. Le WiMAX mobile supporte aussi un transfert intercellulaire progressif pour permettre à une station mobile de passer d'une station de base à une autre à une vitesse véhiculaire sans interruption de la connexion.

1.2.6 La sécurité

Le WiMAX mobile supporte les meilleures options de sécurité en adoptant la meilleure technologie disponible aujourd'hui. Le support existe pour une authentification mutuelle usager/composant, un protocole flexible de gestion de clé, un chiffrement robuste du trafic, une protection de base du message de commande et de gestion ainsi que des optimisations de protocole de sécurité pour des transferts intercellulaires rapides.

Les aspects d'utilisation des dispositifs de sécurité sont :

- ***Protocole de gestion de clé*** : Le protocole de gestion de clé Version 2, PKMv2, est la base de la sécurité du WiMAX mobile comme défini dans la norme 802.16e.

- ***Authentification usager/terminal*** : Le WiMAX mobile supporte l'authentification de l'usager et du terminal en utilisant le protocole EAP en fournissant un support pour

les qualifications qui sont basées sur le SIM «Subscriber Identify Module», l'USIM «Universal Subscriber Identify Module», le certificat numérique ou le nom d'usager/mot de passe.

- ***Chiffrement du trafic*** : L'AES-CCM est le chiffre utilisé pour protéger toutes les données d'utilisateur au-dessus de l'interface MAC du WiMAX mobile. CCM est un générique authentifiant et chiffrant le mode de chiffre de bloc. Dans ces spécifications, CCM est employé avec le chiffre de bloc d'AES. Les clés utilisées pour acheminer le chiffre sont générées à partir de l'authentification EAP. Une machine d'état de chiffrement du trafic qui a un mécanisme de rafraîchissement périodique de la clé de chiffrement du trafic, TEK, permet d'améliorer la transition soutenue des clés pour augmenter la protection.

- ***Protection du message de contrôle*** : Les paramètres sont protégés en utilisant des mécanismes de l'AES basé sur le CMAC ou la fonction de hachage cryptographique MD5 basé sur le HMAC.

- ***Support rapide de transfert intercellulaire*** : Une procédure d'établissement du lien de 3 manières est adoptée par la norme WiMAX mobile afin d'optimiser les mécanismes de réauthentification pour offrir des transferts intercellulaires rapides. Ce mécanisme est également utile pour empêcher toute attaque lors de la procédure de transfert intercellulaire.

1.3 Conclusion

La norme IEEE 802.16e et particulièrement le WiMAX mobile est une solution large bande qui permet la convergence des réseaux fixe et mobile large bande à travers une technologie commun à accès par radio large bande commun et une architecture de réseau flexible. L'interface sans-fil du WiMAX mobile adopte la technique de l'accès multiple par division orthogonale de la fréquence, OFDMA, pour une performance des multi trajets améliorée dans des environnements ne possédant pas la caractéristique d'une vue direct, NLOS.

Les systèmes WiMAX offrent une grande flexibilité dans les options de déploiement des réseaux et le service offert. Le WiMAX mobile supporte un taux de transfert de données élevé, une qualité de service acceptable, une extensibilité de la largeur de bande, une sécurité à base de la meilleure technologie disponible aujourd'hui et une mobilité avancée.

La couche physique de la norme IEEE 802.16e est basée sur la technique de modulation et le mode d'accès OFDMA. La couche MAC est définie pour supporter un très grand trafic de données avec une demande crête de débits très élevée.

La trame de la norme IEEE 802.16e peut être en mode TDD ou FDD. Cependant, Le mode TDD est plus répandu grâce aux nombreux avantages qu'il représente. Dans le chapitre 2, la structure de la trame IEEE 802.16e sera présentée.

CHAPITRE 2

La structure de la trame IEEE 802.16e

Introduction

La trame de la norme IEEE 802.16e ou WiMAX mobile est construite selon un repère bidimensionnel. Ce repère est caractérisé par deux axes : axe fréquentiel et axe temporel. L'axe fréquentiel est représenté par les sous-canaux et l'axe temporel est représenté par les symboles OFDMA tel qu'illustré à la figure 2.1.

La trame de l'OFDMA dans la norme IEEE 802.16e est divisée en deux sous trames : une sous-trame en voie descendante, DL, et une sous-trame en voie montante, UL. Il y a deux types de trames selon la norme, la trame en mode temporel, TDD, et la trame en mode fréquentiel, FDD. Dépendamment des contraintes rencontrées, l'un ou l'autre mode peut être utilisé pour construire la trame.

Ce chapitre traite en détails des normes et des conventions entourant la structure de la trame selon la norme IEEE 802.16e et le WiMAX mobile.

2.1 Mode de duplexage

Les systèmes WiMAX mobile ou IEEE 802.16e peuvent supporter deux modes de duplexage à savoir le mode temporel, TDD, et le mode fréquentiel, FDD.

En général, pour des raisons économiques et d'interopérabilité, le choix du mode TDD est plus approprié. Le mode TDD permet une allocation dynamique de la largeur de bande entre les trafics de la liaison descendante, DL, et ceux de la liaison montante, UL. Cette allocation dynamique est plus intéressante puisque les applications large bande ont besoin d'un débit en voie descendante plus élevé que le débit en voie montante. En plus, le mode TDD supporte des techniques plus efficaces de l'entrée multiple sortie multiple, MIMO, puisque les canaux en mode TDD sont réciproques et par conséquent, ils permettent un ensemble de techniques

de MIMO plus grand et plus efficace. La conception d'un émetteur/récepteur en mode TDD est moins complexe et par la suite moins coûteuse.

Ces avantages du mode TDD ne rendent pas l'utilisation du mode FDD inexistante. En effet, le mode FDD sera considéré par le forum WiMAX pour satisfaire des marchés spécifiques où les réglementations locales interdisent l'utilisation du mode TDD.

2.2 Structure de la trame point multipoint

Cette section décrit la structure de la trame OFDMA selon la norme IEEE 802.16e et les différents types d'allocations qui peuvent être créés par la couche MAC. La trame est composée de deux sous trames, la sous-trame de la voie descendante DL et la sous-trame de la voie montante UL. La figure 2.1 montre une trame OFDMA en mode TDD [3].

La bande disponible pour la transmission de la trame est divisée en plusieurs sous bandes. À chaque période de transmission, définit par le numéro de symbole OFDMA, la station de base transmis les données sur les différentes sous bandes disponibles en utilisant la technique de modulation OFDM. Pour la transmission de la sous trame de la voie montante, les stations abonnées peuvent transmettre sur leurs sous bandes allouées par la station de base en employant la technique de modulation OFDM.

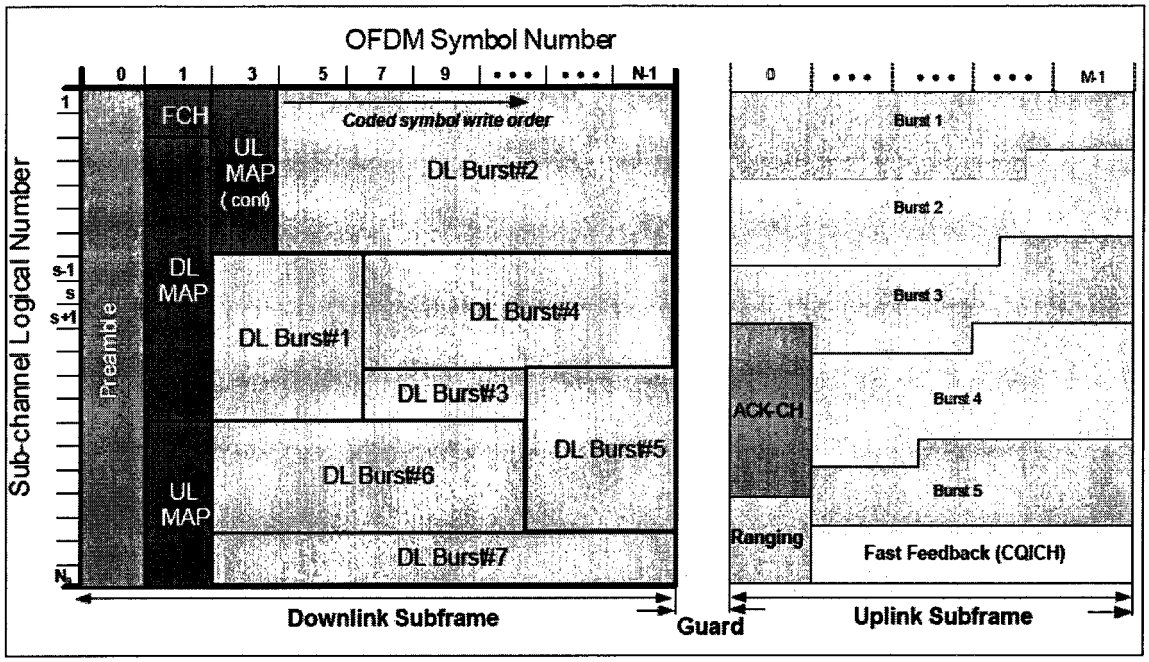

Figure 2.1 ***La structure la de la trame OFDMA en mode TDD.***[7]

2.2.1 Structure de la sous-trame de la voie descendante

Le premier symbole transmis est le préambule. La structure et le traitement du préambule par la station de base sont indiqués dans la norme. Il ne change pas pour chaque trame, ainsi le préambule est présent dans le premier symbole de la trame et n'est pas explicitement décrit dans le descripteur de la trame DL.

2.2.1.1 Les zones de la sous-trame DL

La sous-trame de la voie descendante est divisée en une ou plusieurs zones. Une zone est une région de la sous trame commençant à un symbole et finissant à un autre. Il y a trois types de zones DL. Les zones où les données sont transmises à l'aide d'une antenne simple sont définies comme des zones normales. Dans les zones de codage temps-espace «*space time coding*», STC, les données sont transmises en utilisant le codage STC d'Alamouti, la formation générale de faisceau ou le codage de MIMO «*Multiple Input Multiple Output*»à l'aide de deux antennes ou plus. Dans les zones du système d'antenne adaptatif «*Adaptive*

[7] Tirée de l'article «Mobile WiMAX – Part I : A Technical Overview and Performance Evaluation»

Antenna System», AAS, les données sont transmises en utilisant des techniques de formation de faisceaux spécifiques. Il convient de noter que la norme emploie le même élément d'information «*Information Element*», IE, pour décrire les zones normales et STC dans le message DL-MAP, mais dans le descripteur de la trame, elles sont différenciées [11].

La norme définit également un symbole facultatif de synchronisation commun, qui peut être transmis comme dernier symbole de la voie descendante, dans chaque quatrième trame. Bien que le symbole commun de synchronisation ne soit pas techniquement une zone, la structure en tranche de la trame peut exiger que des symboles additionnels qui précèdent le symbole commun de synchronisation soient laissés inutilisés. Il est commode de décrire le symbole commun de synchronisation comme type de zone dans les descripteurs de la trame.

Une zone spéciale de calibration d'AAS peut également être nécessaire pour être assignée sur demande. La couche physique peut signaler le besoin d'une action appropriée de calibration d'AAS et la couche MAC devrait assigner un nombre approprié de symboles de la fin dans la sous trame de la voie descendante, à un certain temps future.

2.2.1.2 Les rafales de la voie descendante

Dans chaque zone, les ressources de transmission sont allouées sous forme de rafales «Bursts» [11]. Les sections suivantes décrivent les différents types de rafales qui existent dans la sous trame de la voie descendante, DL.

2.2.1.2.1 Préambule

C'est un message utilisé pour la synchronisation de la trame ainsi que sa détection par les stations abonnées. Il occupe le premier symbole OFDM de la trame.

2.2.1.2.2 En-tête de contrôle de la trame FCH

Le message «*Frame control Header*», FCH, occupe toujours une région rectangulaire et il est aussi le premier message transmis de la trame après le préambule. Le message FCH contient les informations concernant le niveau de modulation et le taux de codage du message DL-MAP ainsi que sa longueur. Le message FCH est toujours encodé utilisant un niveau spécifique du taux de codage et de modulation. Il y a deux différents formats du message

FCH, un pour la FFT de 128 points et un autre pour le reste des tailles FFT. Pour les tailles de la FFT autre que 128, les quatre premières tranches contiennent 48 bits modulés par le QPSK avec un taux de codage de ½ et une répétition de codage de 4. Pour la FFT de 128 points, La première tranche est dédiée au message FCH et la répétition n'est pas appliquée [11].

La figure 2.2 montre la structure du message FCH [12].

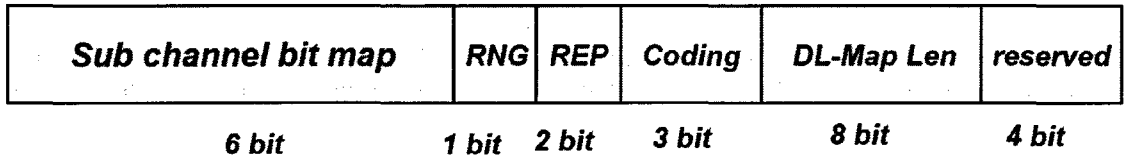

Figure 2.2 ***La structure du message FCH.***[8]

Le contenu du message FCH est le suivant [12] :

- Topogramme binaire des sous canaux utilisés «*Subchannel bitmap*».
  - Fait correspondre les sous canaux utilisés dans chaque segment.
  - Numérote logiquement les sous canaux alloués.
- Indication de changement de signalisation, *RNG*.
- Paramètres de codage du DL_MAP «*Coding*».
  - Indication de répétition de codage (1, 2, 4 ou 6).
  - Indication de codage CC (obligatoire), BTC, CTC ou ZT CC.
- Longueur DL-MAP «*DL-MAP Len*».

2.2.1.2.3 Message DL-MAP

Le message DL-MAP suit immédiatement le message FCH dans le même symbole OFDMA et le sous canal logique suivant. Ce message contient toutes les informations concernant les rafales de données utiles à transmettre par la station de base comme le profil et les tranches occupées. Aussi, le message DL-MAP indique la longueur du message UL-MAP. La norme définit plusieurs types de DL-MAP et UL-MAP. Le message DL-MAP de base commence

---

[8] Tirée de l'article «IEEE 802.16e- 2005 Air Interface Overview»

toujours après le message FCH, mais d'autres types de DL-MAP, comme le Sub-DL-MAP, peuvent commencer à partir d'autres endroits dans la sous-trame de la voie descendante. Il peut y avoir jusqu'à trois messages Sub-DL-MAP par trame. Le message Sub-DL-MAP est utilisé seulement avec le DL compressé et la structure apposée de l'UL-MAP.

L'allocation des tranches pour tous les différents types des messages MAP est faite de la même manière. La figure 2.3 illustre l'allocation des messages MAP dans la première zone de la sous-trame de la voie descendante. Les différentes allocations sont numérotées de 1 à 4. Chaque allocation d'un message débute avec une tranche pour un sous canal et un décalage du symbole OFDMA donnés. La taille de l'allocation est définie par le nombre de tranches occupées par chaque message. Les tranches sont toujours allouées selon l'axe des fréquences en premier lieu. Lorsque le dernier sous canal est atteint, l'allocation saute vers le haut au plus bas nombre du sous canal de la tranche suivante.

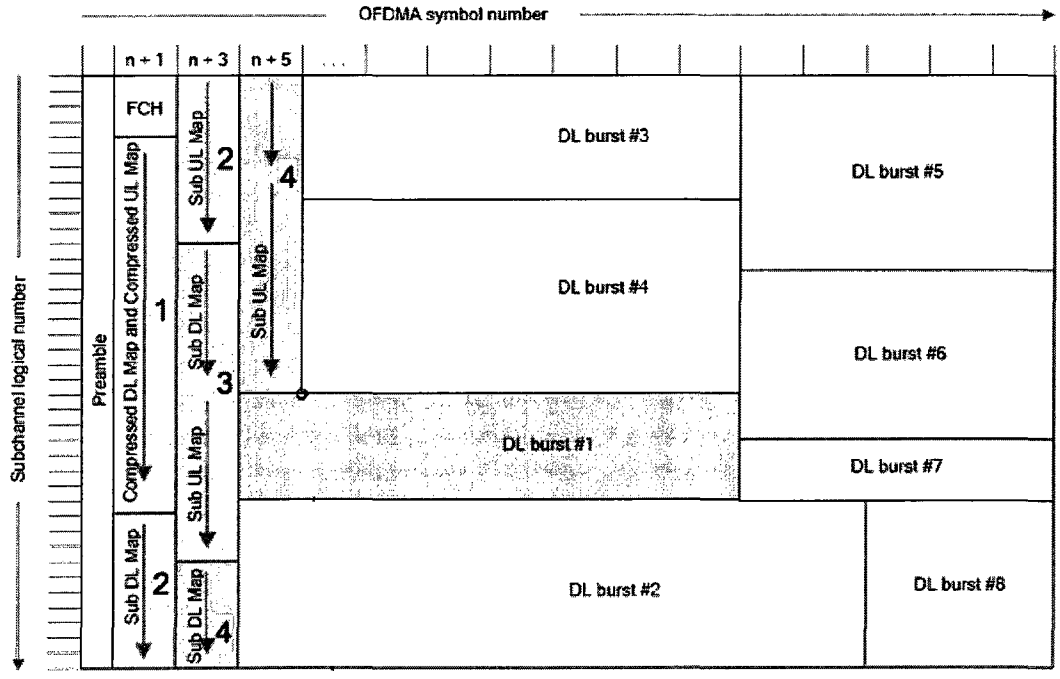

Figure 2.3 ***L'allocation des messages MAP de la sous-trame DL.***[9]

[9] Tirée de l'article «OFDMA PHY SAP Interface Specification for 802.16 Broadband Wireless Access Base Stations»

La figure 2.4 et le tableau 2.1 [1] [12] illustrent le format et le contenu du message DL-MAP :

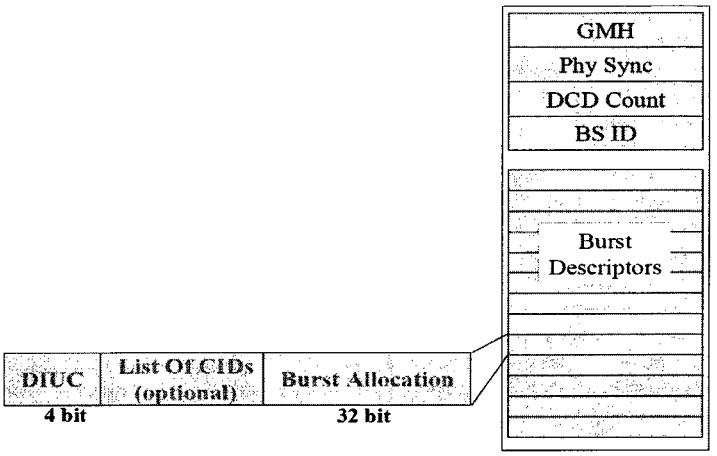

Figure 2.4 ***Le format du message DL-MAP.***[10]

[10] Tirée de l'article « IEEE 802.16e- 2005 Air Interface Overview »

Tableau 2.1

Le format du message DL-MAP[11]

Syntax	Size
DL-MAP_Message_Format() {	
Management Message Type = 2	8 bits
PHY Synchronization Field	*variable*
DCD Count	8 bits
Base Station ID	48 bits
Begin PHY Specific Section {	
for (i = 1; i <= n; i++) {	
DL-MAP_IE()	*variable*
}	
}	
if !(byte boundary) {	
Padding Nibble	4 bits
}	
}	

2.2.1.2.4 Les rafales de données normales

Les rafales de données normales ont été allouées dans des régions rectangulaires de la sous-trame DL [11]. Ces régions sont décrites par un sous canal de départ ainsi qu'un symbole OFDMA de départ et le nombre de sous canaux et symboles OFDMA requis. Ces rafales contiennent les paquets des données utiles destinées aux stations abonnées. La figue 2.3 montre un exemple de 8 rafales de données normales, elles sont étiquetées de DL burst #1 à DL burst #8.

[11] Tiré du standard « Part 16 : Air Interface for Fixed Broadband and Wireless Access Systems»

2.2.1.2.5 Message UL-MAP

La norme définit le message UL-MAP comme une rafale de données normales (lorsque la structure du DL compressé et UL apposé n'est pas utilisée) [11]. Le message UL- MAP est toujours placé dans la première rafale DL. Il contient toutes les informations concernant les rafales de données utiles à transmettre par les stations abonnées comme le profil et les tranches occupées. Le tableau 2.2 illustre le format du message UL-MAP.

Tableau 2.2

Le format du message UL-MAP[12]

Syntax	Size
UL-MAP_Message_Format() {	
Management Message Type = 3	8 bits
Uplink Channel ID	8 bits
UCD Count	8 bits
Allocation Start Time	32 bits
Begin PHY Specific Section {	
for (i = 1; i <= n; i++) {	
UL-MAP_IE()	*variable*
}	
}	
if !(byte boundary) {	
Padding Nibble	4 bits
}	
}	

2.2.1.2.6 L'allocation du PARP

L'allocation du PARP peut être créée n'importe où dans la sous trame pour permettre à la couche physique de minimiser le rapport de la puissance crête sur la puissance moyenne dans

[12] Tiré du standard « Part 16 : Air Interface for Fixed Broadband and Wireless Access Systems»

chaque symbole de la trame, en transmettant des données choisies spécialement pour minimiser ce rapport [11]. La couche MAC de la station de base doit informer la couche physique de l'endroit d'une telle allocation même si l'allocation n'est pas décrite dans le message DL-MAP puisque aucune des stations abonnées n'a besoin de recevoir et de traiter les données dans cette allocation. Pour des fins du descripteur de la trame, les allocations du PARP sont définies pour une allocation rectangulaire.

2.2.2 Structure de la sous trame de la voie montante

2.2.2.1 Les zones de la voie montante

La sous-trame de la voie montante, UL, est divisée en une ou plusieurs zones. Une zone est une région de la sous trame débutant à un symbole et finissant à un autre. Associé à une autre zone, c'est le mécanisme de la sous canalisation qui est utilisé dans cette zone [11].

La norme définit deux types de zones dans la sous trame de la voie montante. Les rafales transmises dans des zones non-AAS peuvent être reçues en utilisant une antenne simple ou en utilisant des antennes multiples. Dans les zones AAS, les rafales sont reçues à la station de base en utilisant des techniques spécifiques de construction de faisceaux [11].

2.2.2.2 Les rafales de la voie montante

Dans chaque zone, la transmission des ressources est allouée en rafales. Les rafales ne traversent pas les zones. Les sections suivantes décrivent les différents types de rafales qui existent dans la liaison montante [11].

2.2.2.2.1 Canal HARQ ACK

Les canaux HARQ ACK sont définis comme des régions rectangulaires dans la sous-trame. Ce message est utilisé pour transmettre les acquittements des paquets reçus par les stations abonnées. Ils ont une structure spécifique où une tranche transporte les acquittements ACK pour deux différents canaux HARQ à partir de différentes stations abonnées. Les acquittements ACK individuels sont définis comme des sous canaux HARQ ACK [11].

2.2.2.2.2 Circuit de retour rapide

Les circuits de retour rapide «Fast Feedback channel» sont définis comme des régions rectangulaires dans la sous-trame. Ces messages sont utilisés pour retourner rapidement l'information sur la qualité du canal. Ils ont aussi une structure spécifique dans laquelle les tranches sont partagées par plusieurs canaux. Les tranches dans une allocation du circuit de retour rapide peuvent transporter différents types de données. Les circuits de retour rapide normaux transportent 4 bits d'information, les circuits de retour rapide améliorés transportent 6 bits d'information ou 3 bits dans une demi-tranche. Les circuits primaires et secondaires transportent 6 et 4 bits respectivement. Les tranches sont allouées dynamiquement aux abonnés par la station de base périodiquement pour un temps limité ou illimité.

2.2.2.2.3 Régions de signalisation d'accès

Les messages de signalisation d'accès sont utilisés par les stations abonnées soit pour signaler l'accès au réseau soit pour faire une requête de la largeur de bande. Il y a plusieurs types de contention utilisés pour la signalisation d'accès et la requête de la largeur de bande. Ce sont : la signalisation d'accès initiale, la signalisation d'accès périodique et la requête de la largeur de bande [11].

Il y a deux types d'allocations de région basées sur la contention. Le premier type est employé pour la signalisation d'accès initiale et la signalisation d'accès du transfert intercellulaire. Le deuxième type d'allocation est la requête périodique de la largeur de bande. Il convient de noter que selon la norme, les deux types de signalisation sont alloués en utilisant le même élément de l'information, IE, mais il est commode pour les descripteurs de la trame de faire une distinction entre ces deux types d'allocations [11,12].

Les deux types d'allocations sont définis en tant qu'allocations rectangulaires. Chaque allocation se compose d'une ou de plusieurs tranches de signalisation d'accès. La figure 2.5 illustre un exemple de signalisation d'accès qui contient plusieurs tranches de signalisation d'accès [11].

Les dimensions d'une tranche dépendent du type de l'allocation et du nombre spécifique sur lequel la signalisation d'accès est accomplie. Pour la signalisation d'accès initiale et la signalisation d'accès du transfert intercellulaire, les choix sont 2 ou 4 symboles, alors que

pour la signalisation d'accès périodique et la requête de la largeur de bande, les choix sont 1 ou 3 symboles. Les secteurs inutilisés à la fin de l'allocation des régions peuvent exister si le nombre de symboles dans l'allocation n'est pas un multiple du nombre de symboles dans la région [11].

Dans la figure 2.5, le nombre de symboles requis pour transmettre le code approprié de signalisation d'accès et la requête de la largeur de bande (1, 2, 3 ou 4 symboles) est dénoté comme N1. N2 dénote le nombre de sous canaux requis pour transmettre un code de signalisation d'accès, 6 dans le cas du PUSC et TUSC1 et 8 dans le cas du PUSC optionnel, TUSC2 et AMC [11].

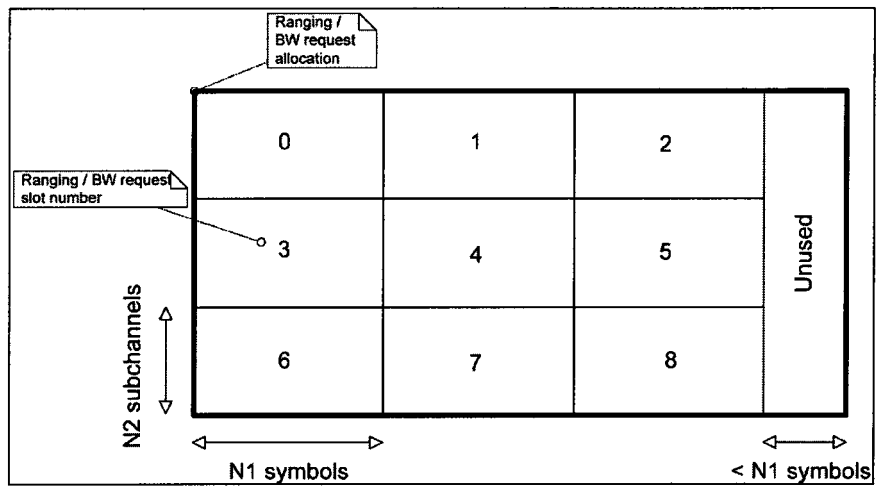

Figure 2.5 ***L'allocation de signalisation d'accès.***[13]

2.2.2.2.4 Les rafales de données normales

Les rafales de données normales de la voie montante, UL, sont définies par un indice de début de sous canal, un indice de début de symbole OFDMA et une durée exprimée en nombre de tranches [11]. Ces rafales contiennent les paquets des stations abonnées destinés à la station de base.

---

[13] Tirée de l'article «OFDMA PHY SAP Interface Specification for 802.16 Broadband Wireless Access Base Stations»

Les allocations sont créées en assignant des tranches le long de l'axe du temps jusqu'à la fin de la zone de l'allocation et ensuite il faut sauter au premier symbole/tranche dans le prochain sous canal et le processus continu le long de l'axe de temps, l'axe des symboles OFDMA.

La figure 2.6 illustre plusieurs allocation de rafales de données normales dans une zone, dénoté UL burst #1 à UL burst #4. L'ordre dans lequel les tranches sont allouées est présenté par les flèches qui recouvrent le message UL burst #2.

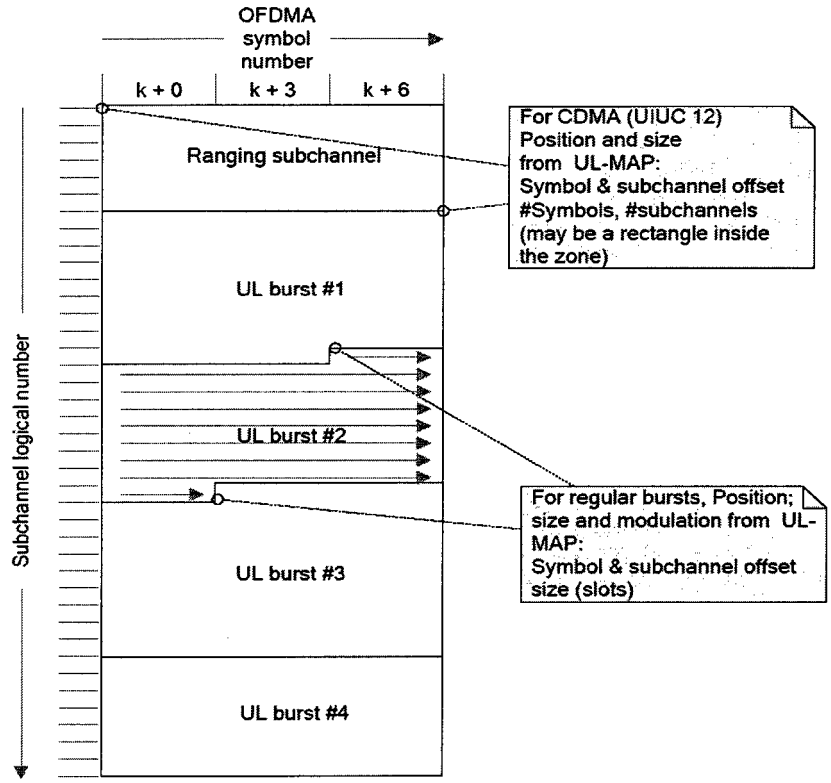

Figure 2.6 ***L'allocation des rafales de données normales dans la voie montante.***[14]

[14] Tirée de l'article «OFDMA PHY SAP Interface Specification for 802.16 Broadband Wireless Access Base Stations»

2.3 Préfixe de la sous trame de la voie descendante

Le message DL_Frame_Prefix est une structure de données transmise au début de chaque trame et contient l'information concernant la trame courante et il est contenu dans le message FCH. Les tableaux 2.3 et 2.4 définissent la structure du message DL_Frame_Prefix.

Tableau 2.3

Le format du préfixe de la trame DL pour toutes les tailles de FFT excepté 128

Syntaxe	Taille (Bits)	Notes
DL_Frame_Prefix_format() {		
Bitmap des sous-canaux utilisés	6	Bit #0 : Groupe 0 des sous-canaux
		Bit #1 : Groupe 1 des sous-canaux
		Bit #2 : Groupe 2 des sous-canaux
		Bit #3 : Groupe 3 des sous-canaux
		Bit #4 : Groupe 4 des sous-canaux
		Bit #5 : Groupe 5 des sous-canaux
réservé	1	Doit être mis à zéro
Repetition_Coding_Indication	2	00 – Aucun codage de répétition en DL-MAP
		01 – Codage de répétition de 2 utilisé en DL-MAP
		10 – Codage de répétition de 4 utilisé en DL-MAP
		11 – Codage de répétition de 6 utilisé en DL-MAP
Coding_Indication	3	0b000 – Codage CC utilisé en DL-MAP
		0b001 – Codage BTC utilisé en DL-MAP
		0b010 – Codage CTC utilisé en DL-MAP
		0b011 – Codage ZT CC utilisé en DL-MAP
		0b100 – Codage LDPC utilisé en DL-MAP
		0b101 to 0b111 -Réservé
DL-Map_Length	8	—
réservé	4	Doit être mis à zéro
}	—	—

Tableau 2.4

Le format du préfixe de la trame DL pour la taille de FFT de 128

Syntaxe	Taille (Bits)	Notes
DL_Frame_Prefix_format() {		
Bitmap des sous-canaux utilisés	1	0 : Sous-canal 0 est utilisé pour le segment 0
		Sous-canal 1 est utilisé pour le segment 1
		Sous-canal 2 est utilisé pour le segment 2
		1 : Utilise tous les sous-canaux
réservé	1	Doit être mis à zéro
Repetition_Coding_Indication	2	00 – Aucun codage de répétition en DL-MAP
		01 – Codage de répétition de 2 utilisé en DL-MAP
		10 – Codage de répétition de 4 utilisé en DL-MAP
		11 – Codage de répétition de 6 utilisé en DL-MAP
Coding_Indication	3	0b000 – Codage CC utilisé en DL-MAP
		0b001 – Codage BTC utilisé en DL-MAP
		0b010 – Codage CTC utilisé en DL-MAP
		0b011 – Codage ZT CC utilisé en DL-MAP
		0b100 – Codage LDPC utilisé en DL-MAP
		0b101 à 0b111 -Réservé
DL-Map_Length	5	—
}	—	—

Le tableau 2.5 illustre les sous-canaux de chaque groupe pour les différentes tailles de la FFT.

Tableau 2.5

Indexe des sous-canaux des groupes de 6 sous-canaux

Taille FFT	Groupe de sous-canaux	Sous-canaux	Taille FFT	Groupe de sous-canaux	Sous-canaux
2048	0	0-11	512	0	0-3
	1	12-19		1	4
	2	20-31		2	5-9
	3	32-39		3	-
	4	40-51		4	10-14
	5	52-59		5	-
1024	0	0-5	128	0	0
	1	6-9		1	-
	2	10-15		2	1
	3	16-19		3	-
	4	20-25		4	2
	5	26-29		5	-

Avant d'être inséré dans le message FCH, le message DL_Frame_Prefix de 12 bits doit être répété 4 fois pour former un bloc de 48 bits, qui est la taille minimale du bloc FEC.

2.4 Tranche et régions de données

Une tranche dans la couche physique de l'OFDMA est une unité bidimensionnelle qui requiert une dimension temporelle, nombre de symboles OFDMA, et une dimension fréquentielle, nombre de sous-canaux [1].

La définition d'une tranche OFDMA dépend de la structure du symbole OFDMA. Qui varie pour la sous-trame de la voie descendante et pour la sous-trame de la voie montante. Aussi, la définition de cette tranche dépend du mode des canaux partiellement utilisés, du mode des canaux entièrement utilisés, pour les permutations distribuées de la sous-porteuse ainsi que la permutation adjacente de la sous-porteuse.

Pour le mode des canaux entièrement utilisés de la liaison descendante et le mode optionnel des canaux entièrement utilisés de la liaison descendante utilisant la permutation distribuée de la sous-porteuse, une tranche est un sous-canal par un symbole OFDMA, 1x1.

Pour le mode des canaux partiellement utilisés de la liaison descendante utilisant la permutation distribuée de la sous-porteuse, une tranche est un sous canal par deux symboles OFDMA, 2x1.

Pour la permutation adjacente de la sous-porteuse, une tranche est un sous-canal par deux, trois ou six symboles OFDMA, 2x1, 3x1 ou 6x1.

Dans un système OFDMA, une région de données est une allocation bidimensionnelle d'un groupe de sous canaux contigus dans un groupe de symboles OFDMA contigus. Cette allocation peut être vue comme un rectangle, comme le rectangle 4x3 illustré à la figure 2.7.

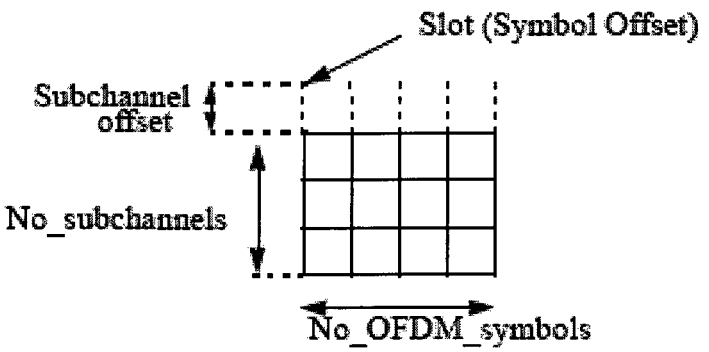

Figure 2.7 ***Exemple de la région de données qui définie l'allocation OFDMA.***[15]

[15] Tirée du standard « Part 16 : Air Interface for Fixed Broadband and Wireless Access Systems»

2.5 Segment

Un segment est une subdivision des ensembles des sous-canaux OFDMA disponibles. Un segment est utilisé pour déployer une instance unique du MAC [1].

2.6 Insertion des données OFDMA

Les données MAC doivent être insérées dans une région de données OFDMA de la voie montante UL et de la voie descendante DL en utilisant les algorithmes I et II définis ci-dessous [1,2].

I- La voie descendante DL [1] :

- Segmenter les données en blocs taillés pour s'insérer dans une tranche OFDMA. Chaque tranche doit s'étendre sur un ou plusieurs sous canaux sur l'axe des sous-canaux et deux symboles OFDMA sur l'axe de temps tel qu'illustré à la figure 2.8.
- Faire correspondre les tranches tel que la plus petite tranche occupe le plus petit sous canal dans le plus petit symbole OFDMA.
- Continuer l'insertion en augmentant l'indice des symboles OFDMA. Lorsque la limite de la région des données est atteinte, continue l'insertion à partir du plus petit indice du symbole OFDMA dans le prochain sous canal.

II- La voie montante UL [1]:

- Segmenter les données en blocs taillés pour s'insérer dans une tranche OFDMA. Chaque tranche doit s'étendre sur un ou plusieurs sous canaux sur l'axe des sous canaux et trois symboles OFDMA sur l'axe de temps tel qu'illustré à la figure 2.9.
- Faire correspondre les tranches tel que la plus petite tranche occupe le plus petit sous canal dans le plus petit symbole OFDMA.
- Continuer l'insertion en augmentant l'indice des symboles OFDMA. Lorsque la limite de la région des données est atteinte, continue l'insertion à partir du plus petit indice du symbole OFDMA dans le prochain sous canal disponible.

Les figures 2.8 et 2.9 illustrent l'ordre dans lequel les tranches OFDMA sont insérées aux sous-canaux et symboles OFDMA [1].

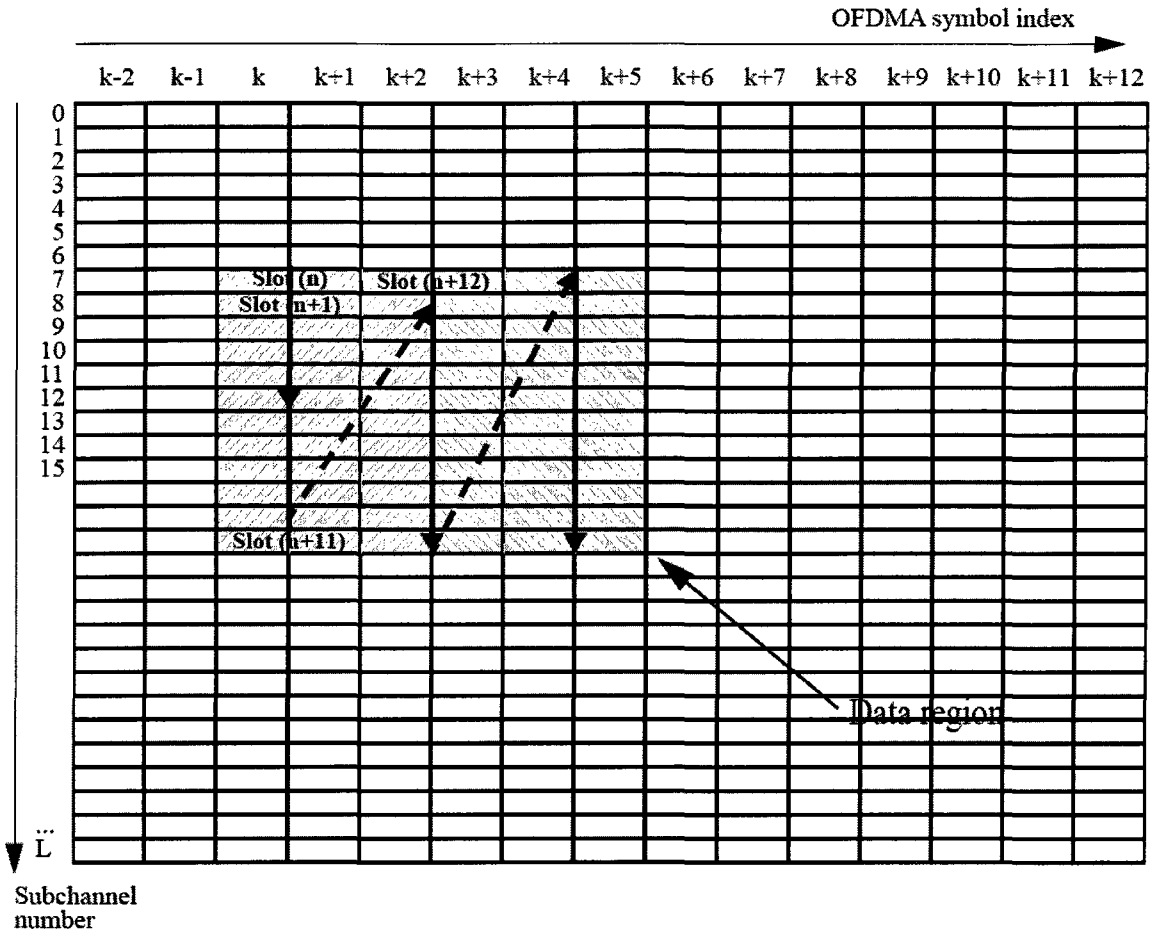

Figure 2.8 ***Exemple d'insertion des tranches OFDMA aux sous-canaux et symboles en voie descendante en mode PUSC.***[16]

[16] Tirée du standard « Part 16 : Air Interface for Fixed Broadband and Wireless Access Systems, Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1»

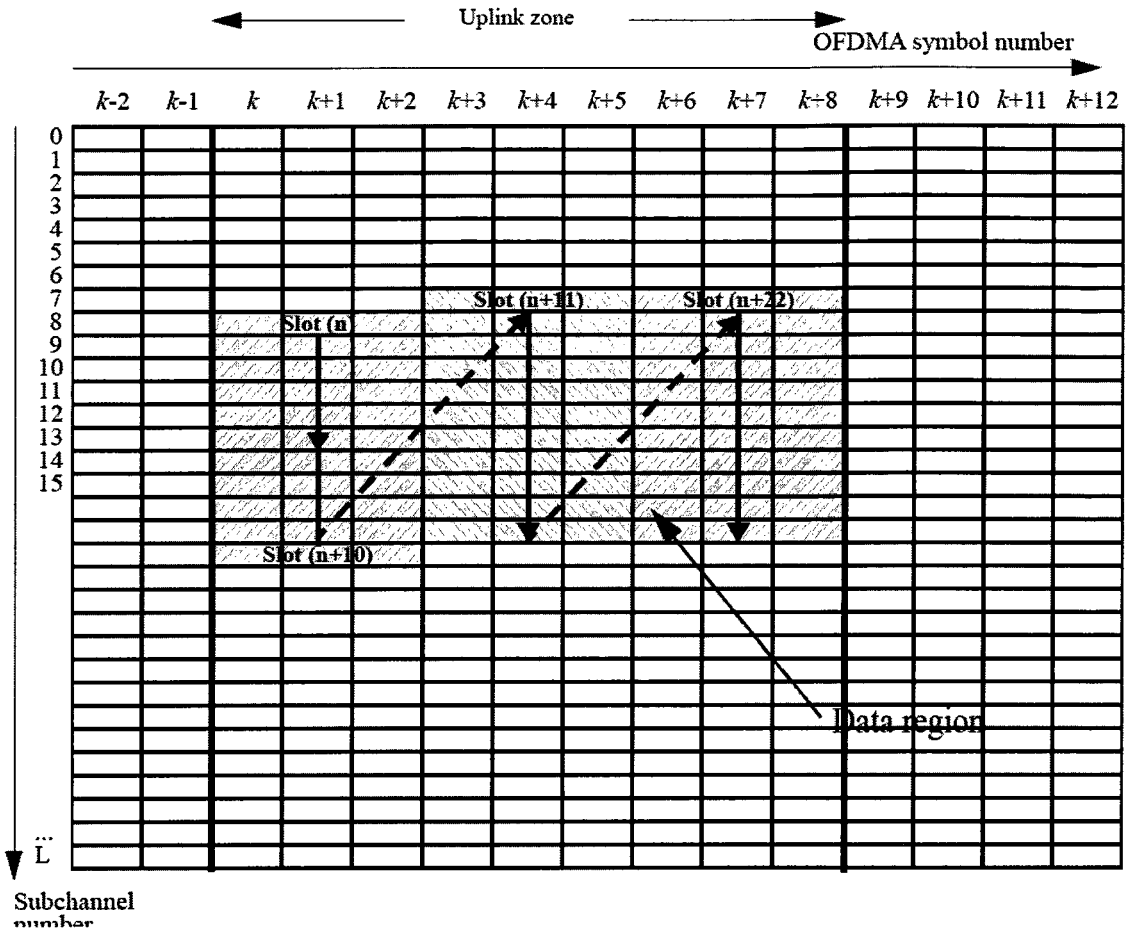

Figure 2.9 ***Exemple d'insertion des tranches OFDMA aux sous canaux et symboles en voie montante.***[17]

2.7 Conclusion

Dans ce chapitre, une présentation de la structure de la trame de la norme IEEE 802.16e ou WiMAX mobile a été introduite. Cette trame, divisée en deux sous-trames, est construite selon un repère bidimensionnel. Ce repère est caractérisé par deux axes : axe fréquentiel et axe temporel. L'axe fréquentiel est représenté par les sous-canaux et l'axe temporel est représenté par les symboles OFDMA.

---

[17] Tirée du standard « Part 16 : Air Interface for Fixed Broadband and Wireless Access Systems, Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1»

La sous-trame de la voie descendante contient des messages de contrôle ainsi que les données utiles de la voie descendante. Ces données sont regroupées en rafales et chaque rafale doit occuper une région rectangulaire de la sous-trame de la voie descendante.

La sous-trame de la voie montante contient des messages d'information ainsi que les données utiles de la voie montante.

Dans le chapitre 3, deux algorithmes seront proposés pour la mise en trame des paquets de données utiles ainsi que les données de contrôle.

CHAPITRE 3

Construction de la trame WiMAX mobile

3.1 Définition du problème

L'objectif principal est de remplir une trame de forme rectangulaire avec des données qui, en général, n'occupent pas toujours une région rectangulaire. Selon l'arrivée des paquets, la trame doit être construite en temps réel tout en minimisant les tranches qui seront laissées vides.

En résumé, le problème à résoudre est énoncé comme suit :

- Remplir la trame par des paquets qui arrivent selon un ordre déterminé par l'ordonnanceur (Scheduler).
- Respecter l'ordre des arrivés des paquets à 100%.
- Optimiser la construction de la trame pour réduire le nombre de tranches qui resteront vides.

La forme de la trame à remplir est illustrée à la figure 3.1 :

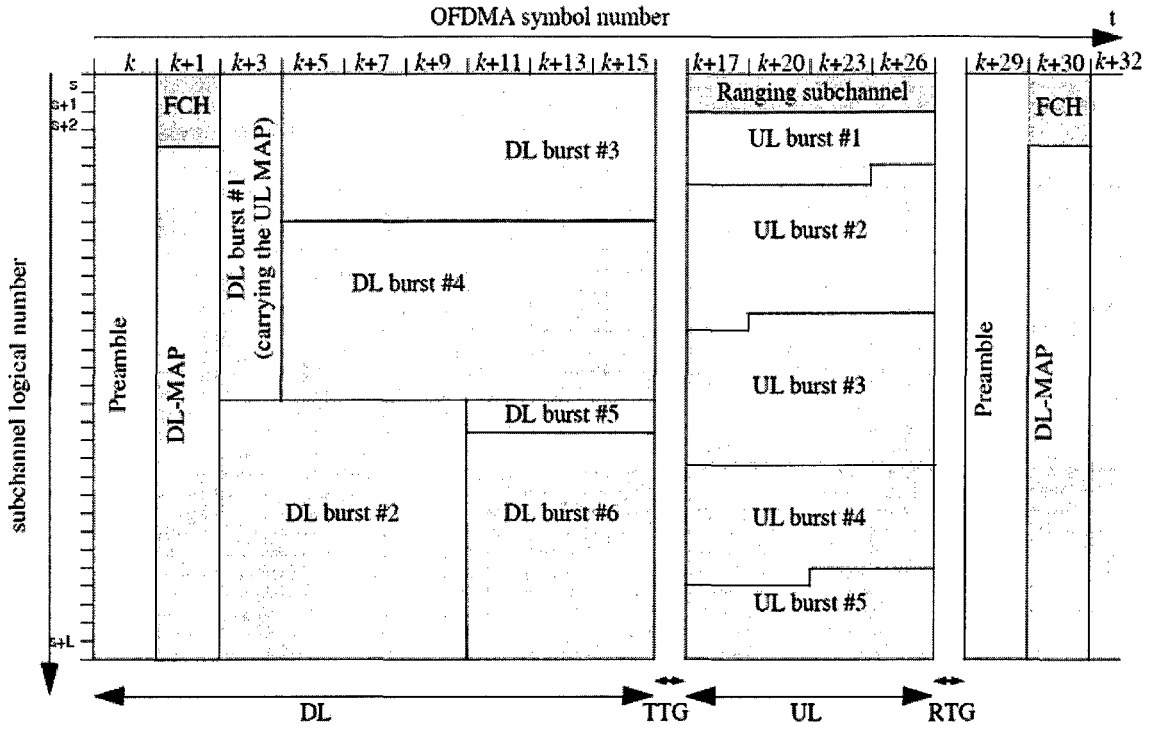

Figure 3.1 ***Exemple d'une trame en mode TDD.***[18]

La difficulté majeure pour remplir cette trame réside dans le fait que les rafales de données de la voie descendante, DL burst #1 à DL burst #N_p, devront être rectangulaires, où N_p est le nombre de rafales de la voie descendante, DL.

Pour les autres éléments de la trame, les messages de contrôle et les rafales de données de la voie montante, leurs formes ne causent pas de problème d'optimisation de remplissage.

3.2 Étapes à suivre pour construire la trame

L'étape de la construction de la trame survient après l'opération de l'ordonnancement «Scheduling». L'ordonnanceur fournit au constructeur de la trame les paquets selon un ordre

[18] Tirée du standard « Part 16 : Air Interface for Fixed Broadband and Wireless Access Systems, Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1»

respectant la qualité de service, QoS. Au fur et à mesure que les paquets arrivent au constructeur qui exécute son algorithme pour créer la trame en regroupant les paquets selon leurs profils. La figure 3.2 montre d'une manière globale le processus de construction de la trame.

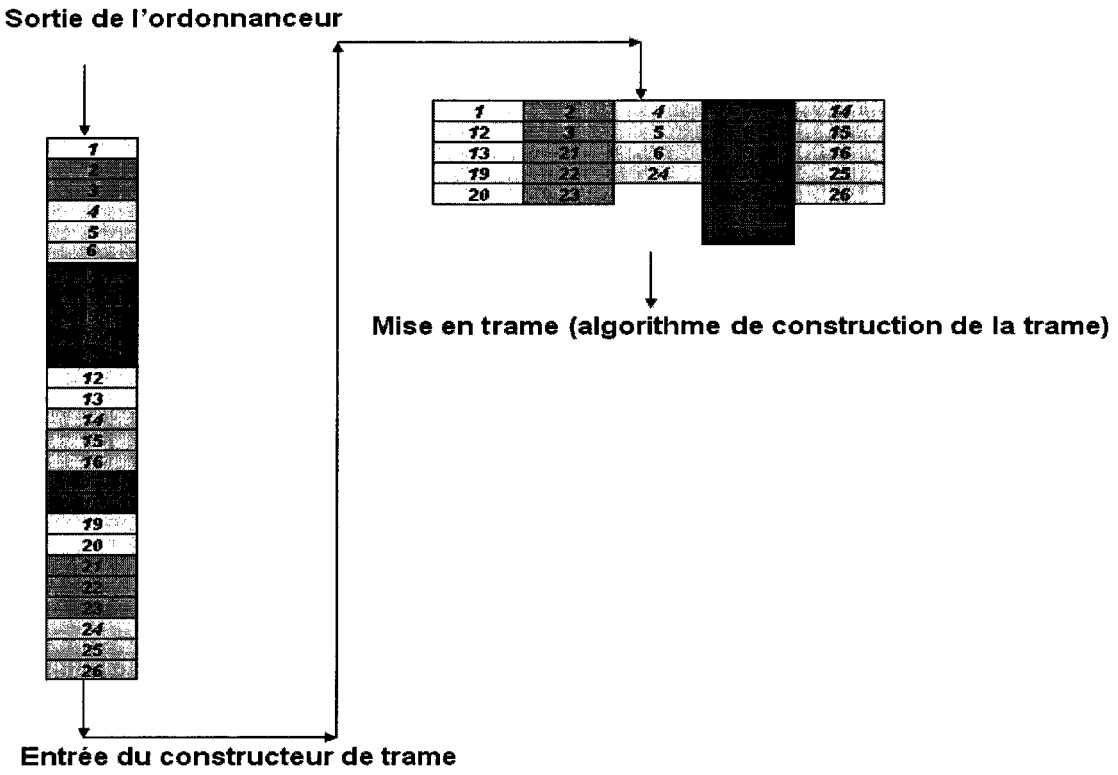

Figure 3.2 ***Vue globale du processus de construction de la trame.***

Les étapes à suivre pour construire la trame sont présentées ci-dessous selon l'ordre de construction de chaque message :

- **Préambule :** Ce message occupe toute la bande de fréquence, tous les sous-canaux, et le premier symbole OFDMA de la trame. Son profil, modulation et taux de codage, est : BPSK-1/2.
- **FCH :** Ce message occupe deux symboles OFDMA, le deuxième ainsi que le troisième, et les deux premiers sous-canaux (2x2). Son profil est QPSK-1/2.

- **DL-MAP et les rafales de données de la voie descendante, DL burst #1 à DL burst #Np :**
    - DL-MAP : Ce message occupe deux symboles OFDMA mais le nombre de sous-canaux occupés par ce message est variable. En fait, sa longueur est dépendante du nombre de connexions, CID, de chaque profil. Sa longueur est alors déterminée dynamiquement conjointement avec le contenu des rafales de données, DL burst #1 à DL burst #N_p. Son profil est QPSK-1/2.
    - DL burst #1 à DL burst #N_p : Ces messages doivent occuper une région rectangulaire. Ils occupent la région restante de la sous-trame de la voie descendante, DL, après l'allocation des messages Préambule, FCH, DL-MAP et UL-MAP. Il y a une relation entre le nombre de paquets qui seront contenus dans ces rafales et la longueur du message DL-MAP. Ceci impose de trouver une combinaison qui permet de maximiser le nombre de paquets qui seront transmis, tout en respectant de l'aspect rectangulaire de l'occupation de ces rafales et le vide qui sera laissé.
- **Sous canal de signalisation d'accès :** Ce message occupe les deux premiers sous-canaux et tous les symboles OFDMA de la sous-trame de la voie montante, UL.
- **Rafales de données de la voie montante, UL burst #1 à UL burst #Np :** Ces rafales sont assignées à des tranches l'une après l'autre (#1 à #N_p). La contrainte de la forme rectangulaire n'est pas présente dans ce cas.

3.3 Algorithmes de construction de la trame

Vu que la région à remplir n'est pas rectangulaire et qu'elle doit être remplie par des blocs rectangulaires, il peut y avoir des espaces qui restent vides. L'objectif est, alors, de trouver une méthode ou un algorithme, qui minimisera les espaces vides.

Le premier algorithme est basé sur la combinaison des différentes rafales de données de la voie descendante, *DL Burst*, possibles qui seront présentes dans la trame pour déterminer de quelle manière chaque rafale sera insérée dans la trame.

Le deuxième algorithme est basé sur le principe du plus grand en premier, c'est-à-dire que la trame sera remplie en premier lieu par la rafale de données de la voie descendante qui occupera le plus grand nombre de tranches puis la suivante jusqu'au remplissage final de la trame.

Les sections suivantes présenteront en détail les algorithmes proposés.

3.3.1 Paramètres de conception des algorithmes

Les paramètres de conception qu'il faut prendre en considération pour la conception d'un algorithme de mise en trame des paquets WiMAX sont : le nombre de profil, le nombre de paquets de chaque profil, le nombre des symboles OFDMA, le nombre de sous-canaux et le nombre de sous-porteuses de données par sous-canal.

La condition requise pour remplir la trame est :

$$\text{N\_subch} * \text{N\_datasub} * \text{N\_OFDMAsymb} \succ \sum_{i=1}^{Np} (8 * CR_i * NbitperSym_i * N_i)$$

Un remplissage optimal de la trame implique la minimisation de l'expression suivante :

$$\text{N\_subch} * \text{N\_datasub} * \text{N\_OFDMAsym b} - \sum_{i=1}^{Np} (8 * CR_i * NbitperSym_i * N_i)$$

Les paramètres constituant la formule ci-dessus sont définis comme suit :

- N_i : La taille totale des paquets présents dans chaque rafale de données en octets.
- N_subch : le nombre de sous canaux d'une trame.
- N_datasub : le nombre de sous porteuses de données par sous canal.
- N_OFDMAsymb : le nombre des symboles OFDMA d'une trame.
- N_p : Le nombre de profil présent.
- CR_i : Le taux de codage pour un profil i.

- NbitperSym i : Le nombre de bits par symbole pour un profil Ni.

3.3.2 Conception de l'algorithme 1

Vu que la région à remplir n'est pas rectangulaire et qu'elle doit être remplie par des blocs rectangulaires, il peut y avoir des espaces qui resteront vides. Alors, l'objectif est de trouver une méthode qui minimisera ces espaces vides.

Cet algorithme est basé sur la combinaison des Np *DL Bursts* pour déterminer de quelle manière chaque *DL Burst* sera inséré dans la trame. Les combinaisons possibles sont représentées par le tableau 3.1 (où 5 *DL Bursts* sont présents) :

Tableau 3.1

Les combinaisons choisies pour 5 rafales de données

Numéro de la combinaison	***DL Burst composant la combinaison***
Combinaison #1	*DL Burst #1*
Combinaison #2	*DL Burst #2*
Combinaison #3	*DL Burst #3*
Combinaison #4	*DL Burst #4*
Combinaison #5	*DL Burst #5*
Combinaison #6	*DL Burst #1 - DL Burst #2*
Combinaison #7	*DL Burst #1 - DL Burst #3*
Combinaison #8	*DL Burst #1 - DL Burst #4*
Combinaison #9	*DL Burst #1 - DL Burst #5*
Combinaison #10	*DL Burst #2 - DL Burst #3*
Combinaison #11	*DL Burst #2 - DL Burst #4*
Combinaison #12	*DL Burst #2 - DL Burst #5*
Combinaison #13	*DL Burst #3 - DL Burst #4*
Combinaison #14	*DL Burst #3 - DL Burst #5*
Combinaison #15	*DL Burst #4 - DL Burst #5*

L'algorithme suivant montre la procédure à suivre pour remplir la sous trame de la voie descendante. Cet algorithme consiste à essayer trois solutions, la solution A, B et C. La solution A sera essayée en premier lieu. Si cette dernière est adéquate, les deux autres solutions ne seront pas essayées. Or, si la solution A n'est pas adéquate, l'algorithme essaie la solution B. Si cette dernière est adéquate, la solution C ne sera pas essayée. La solution C sera essayée si et seulement si les deux solutions A et B ne sont pas adéquates.

Algorithme 1 :

Tant que ***(la somme des tailles des paquets reçus est inférieure à la taille de la région disponible pour les DL-Burst)***

Faire :

- *Recevoir les paquets classés par l'ordonnanceur.*
- *Classer ces paquets par ensemble de Burst #i (i = 1 à Nbre_DL-Bursts).*

Fin Tant que.

- *Mémoriser l'**indice** du dernier paquet reçu.*
- *Faire la combinaison des Bursts disponibles en 1 et 2 Bursts pour chaque combinaison.*
- *Si **la solution A** est adéquate (l'une des combinaisons possibles peut s'insérer dans région 1 avec le minimum d'espace vide), adopter la solution A,*
- *Sinon :*
    - *Si **la solution B** est adéquate (l'une des combinaisons possibles peut s'insérer dans région 2 avec le minimum d'espace vide), adopter la solution B,*
    - *Sinon : adopter **la solution C**.*
- *Fin Si*

Faire :

- *Recevoir le paquet suivant (indice +1),*
- *Déterminer son profil **i**,*

- *Insérer le paquet dans le Burst #i correspondant,*
- *Si le paquet est plus grand que l'espace libre, fragmenter le paquet.*
- *Fin Si.*
- *Si le dernier paquet ne s'insère pas dans son Burst #i,*
  Fin Faire. (Sortir de la boucle)
- *Sinon : indice ++*

Les sections suivantes présentent en détail les trois solutions que l'algorithme doit utiliser pour construire la trame d'une manière optimale.

Solution A :

La solution A consiste à remplir la région 1 par la meilleure combinaison des DL Bursts de telle sorte à réduire l'espace qui restera vide. Le reste de DL Bursts sera inséré dans la région 2.

La solution A peut être représentée par l'une des 2 figures 3.3 ou 3.4 :

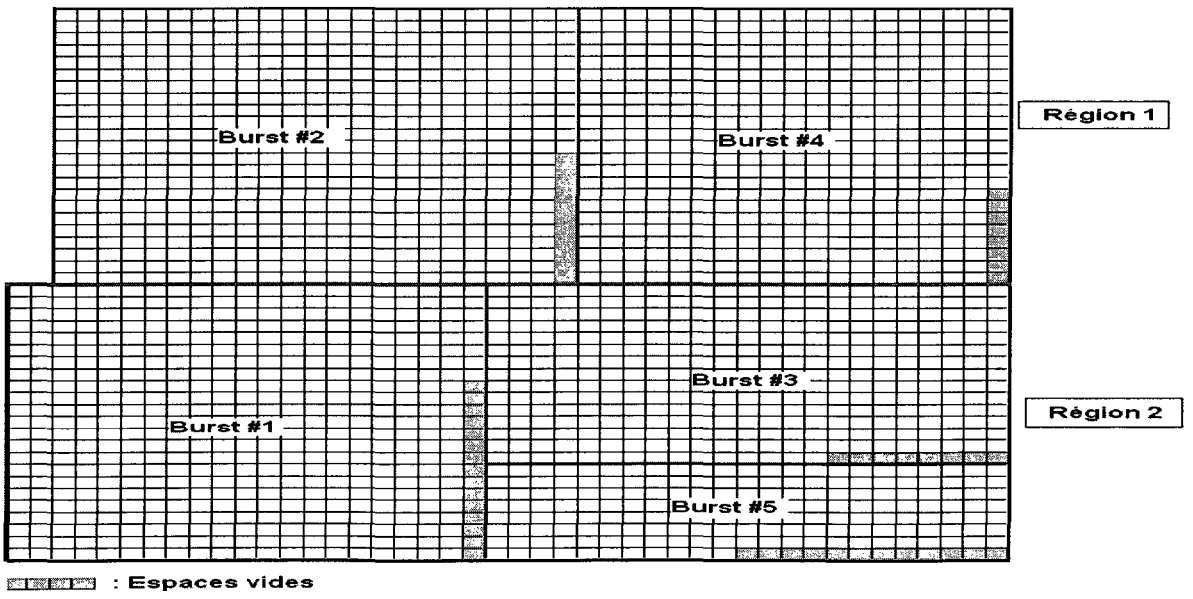

Figure 3.3 ***L'illustration de la sous trame de la voie descendante où la région 1 est occupée par 2 rafales de données.***

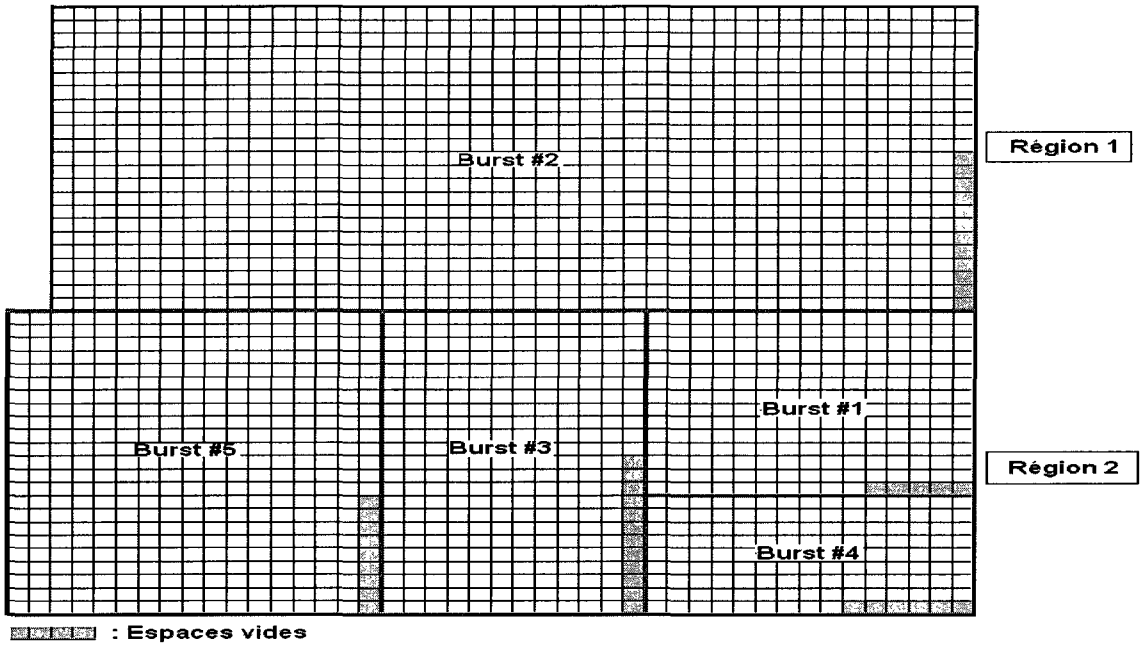

Figure 3.4 ***L'illustration de la sous trame de la voie descendante où la région 1 est occupée par une seule rafale de données.***

L'algorithme de la solution A est décrit ci-dessous :

- *Déterminer quelle combinaison peut être insérée dans la région 1,*
- *Si cette combinaison se compose d'une seule rafale,*
    - *Insérer cette rafale dans la région 1,*
    - *Choisir, parmi les 4 rafales restantes, la plus grande en l'insérant dans la région 2 tel quelle occupe tous les sous-canaux de cette région,*
    - *Choisir, parmi les 3 rafales restantes, la plus grande en l'insérant dans la région 2 tel qu'elle occupe tous les sous-canaux de cette région,*
    - *Insérer les 2 rafales qui restent dans le reste de la région 2 tel qu'elles occupent tous les symboles OFDMA restants et partagent les sous-canaux de la région 2.*
- *Sinon (cette combinaison se compose de 2 rafales)*

- *Insérer les 2 rafales dans la région 1 tel qu'elles occupent tous les sous-canaux et partagent les symboles OFDMA de la région 1.*
- *Choisir, parmi les 3 rafales restantes, la plus grande en l'insérant dans la région 2 tel qu'elle occupe tous les sous-canaux de la région 2,*
- *Insérer les 2 rafales qui restent dans le reste de la région 2 tel qu'elles occupent tous les symboles OFDMA restants et partagent les sous-canaux de la région 2.*

- *Fin Si.*

Solution B :

La solution B consiste à remplir la région 2 par la meilleure combinaison des rafales de données de la voie descendante, *DL Bursts*, de telle sorte à réduire l'espace qui restera vide. Le reste des rafales de données de la voie descendante sera inséré dans la région 1.

La solution B peut être représentée par l'une des 2 figures 3.5 ou 3.6 :

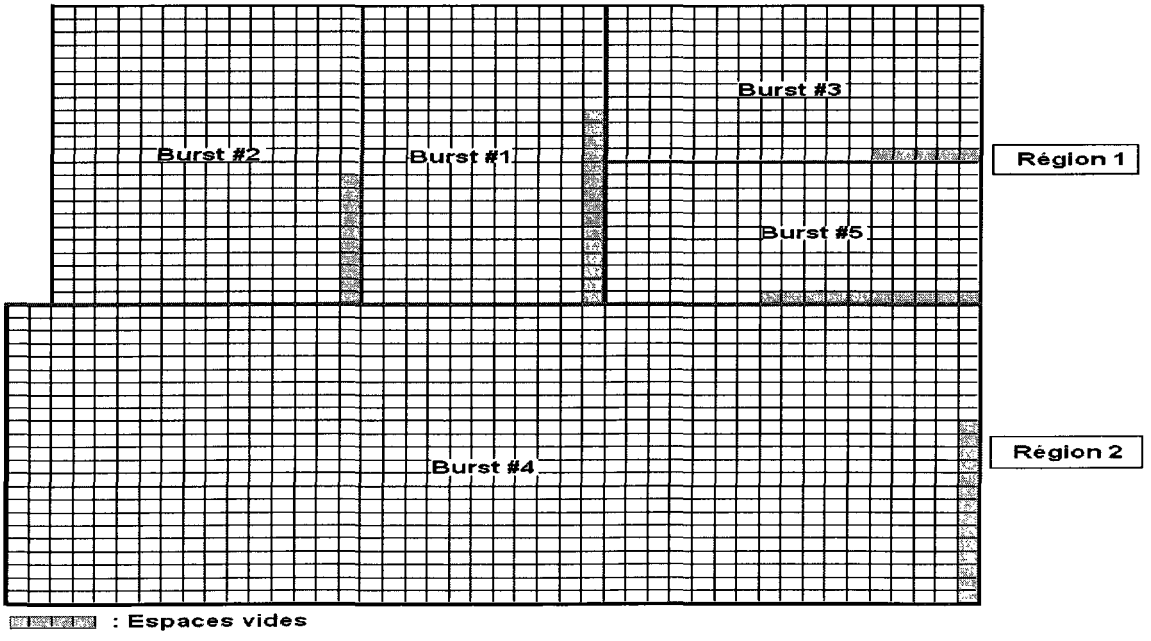

Figure 3.5 ***L'illustration de la sous trame de la voie descendante où la région 2 est occupée par une seule rafale de données.***

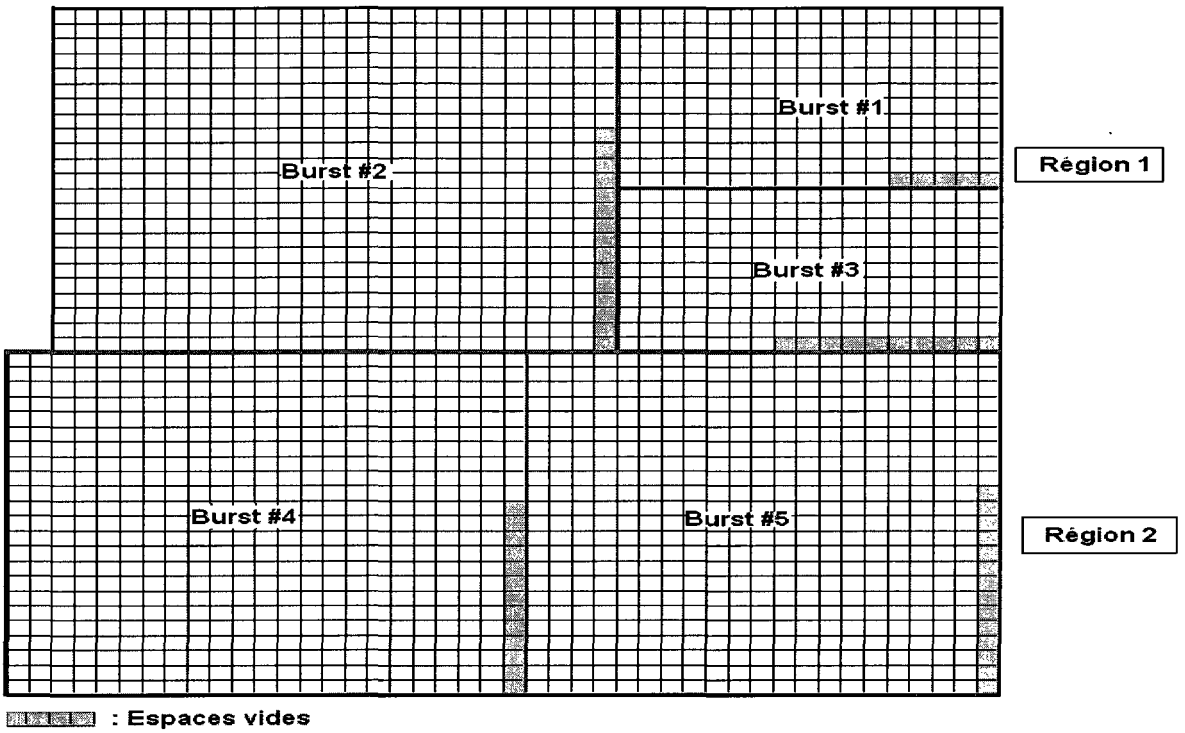

Figure 3.6 ***L'illustration de la sous trame de la voie descendante où la région 2 est occupée par 2 rafales de données.***

L'algorithme de la solution B est décrit comme suit :

- *Déterminer quelle combinaison peut être insérée dans la région 2,*
- *Si cette combinaison se compose d'une seule rafale,*
    - *Insérer cette rafale dans la région 2,*
    - *Choisir, parmi les 4 rafales restantes, la plus grande en l'insérant dans la région 1 tel qu'elle occupe tous les sous-canaux de cette région,*
    - *Choisir, parmi les 3 rafales restantes, la plus grande en l'insérant dans la région 1 tel qu'elle occupe tous les sous-canaux de cette région,*
    - *Insérer les 2 rafales qui restent dans le reste de la région 1 tel qu'elles occupent tous les symboles OFDMA restants et partagent les sous-canaux de la région 1.*

- *Sinon (cette combinaison se compose de 2 rafales)*
    - *Insérer les 2 rafales dans la région 2 tel qu'elles occupent tous les sous-canaux et partagent les symboles OFDMA de la région 2.*
    - *Choisir, parmi les 3 rafales restantes, la plus grande en l'insérant dans la région 1 tel qu'elle occupe tous les sous-canaux de la région 1,*
    - *Insérer les 2 rafales qui restent dans le reste de la région 1 tel qu'elles occupent tous les symboles OFDMA restants et partagent les sous-canaux de la région 1.*
- *Fin Si.*

Solution C :

La solution C consiste à augmenter la taille de la région 1 ce qui amènera une réduction de la taille de la région 2 par un sous canal jusqu'à ce qu'une combinaison des rafales de données de la voie descendante peuvent être insérées dans la région 2 de telle sorte à réduire l'espace qui restera vide. Le reste de DL Bursts sera inséré dans la région 1.

La solution C peut être représentée par l'une des 2 figures 3.7 ou 3.8 :

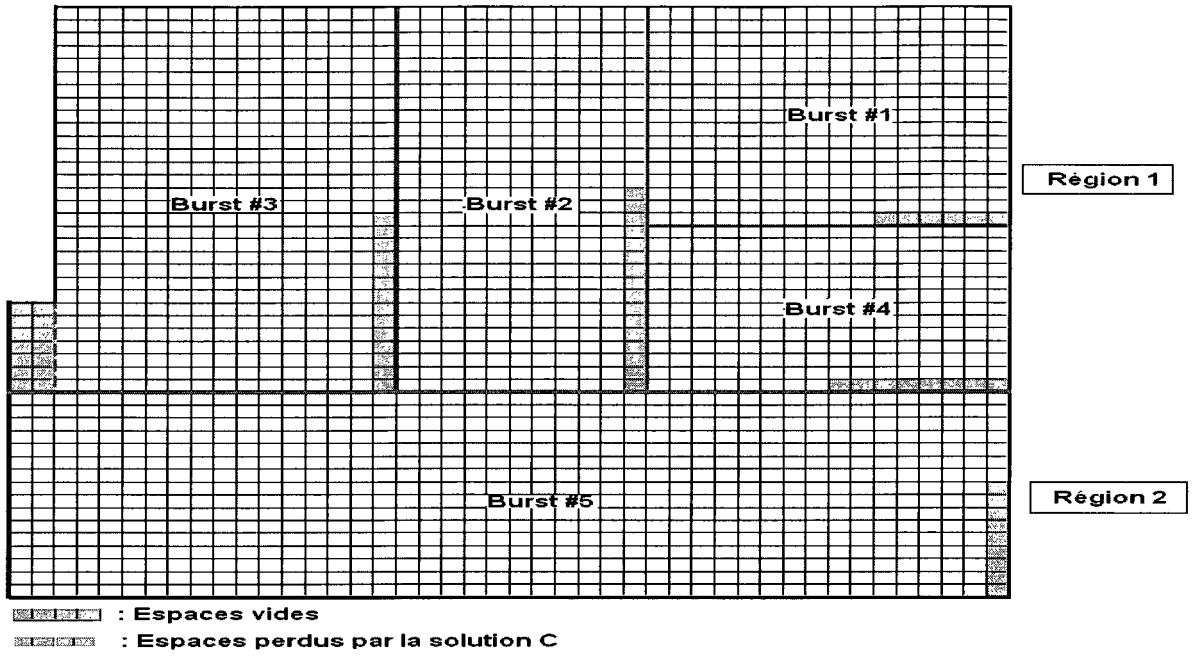

Figure 3.7 ***L'illustration de la sous trame de la voie descendante où la région 2 est occupée par une seule rafale de données.***

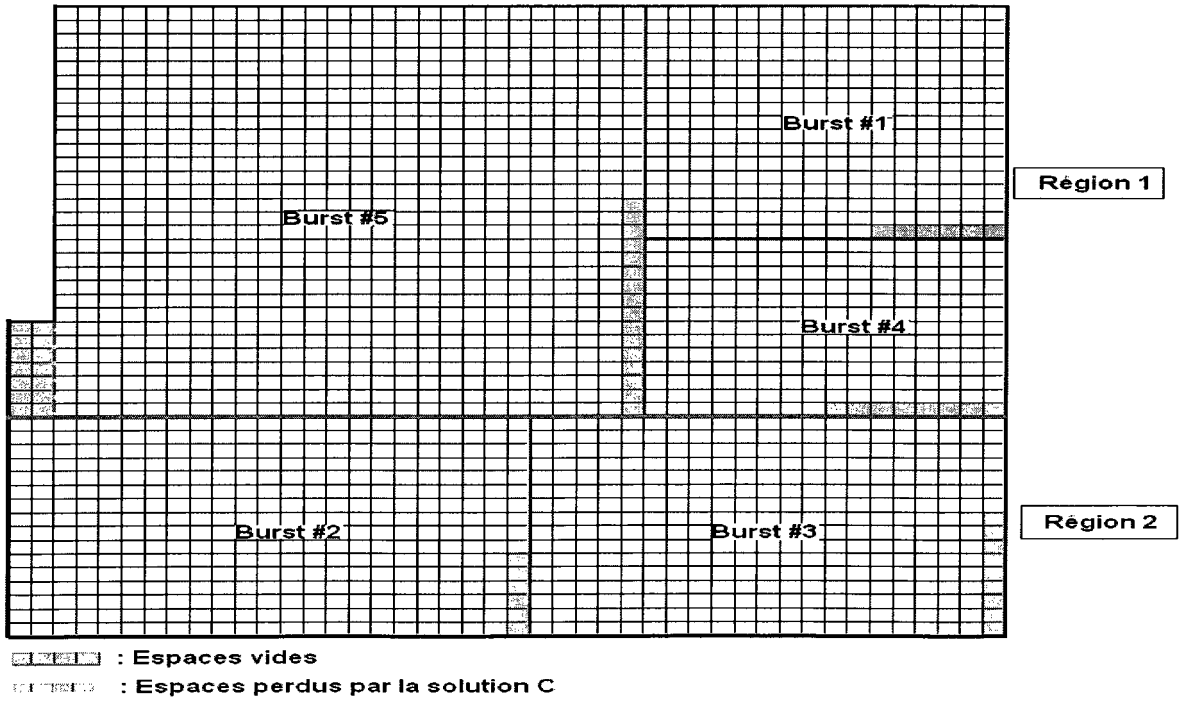

Figure 3.8 ***L'illustration de la sous trame de la voie descendante où la région 2 est occupée par 2 rafales de données.***

L'algorithme de la solution C est le suivant :

Faire :

- *Agrandir la région 1 par un sous-canal (ce qui diminuera la région 2 par un sous-canal),*
- *Si il existe une combinaison qui peut s'insérer dans la région 1*
  - *Déterminer quelle combinaison peut s'insérer dans la région 1,*
  - *Si cette combinaison se compose d'une seule rafale,*
    - *Insérer cette rafale dans la région 1,*
    - *Choisir, parmi les 4 rafales restantes, la plus grande en l'insérant dans la région 2 tel qu'elle occupe tous les sous-canaux de cette région,*
    - *Choisir, parmi les 3 rafales restantes, la plus grande en l'insérant dans la région 2 tel qu'elle occupe tous les sous-canaux de cette région,*
    - *Insérer les 2 rafales qui restent dans le reste de la région 2 tel qu'elles occupent tous les symboles OFDMA restants et partagent les sous-canaux de la région 2.*
    - ***Fin Faire.***
  - *Sinon (cette combinaison se compose de2 rafales)*
    - *Insérer les 2 rafales dans la région 1 tel qu'elles occupent tous les sous-canaux et partagent les symboles OFDMA de la région 1.*
    - *Choisir, parmi les 3 rafales restantes, la plus grande en l'insérant dans la région 2 tel qu'elle occupe tous les sous-canaux de la région 2,*
    - *Insérer les 2 rafales qui restent dans le reste de la région 2 tel qu'elles occupent tous les symboles OFDMA restants et partagent les sous-canaux de la région 2.*
    - ***Fin Faire.***
  - *Fin Si*

- *Sinon : Retourner à la boucle **Faire**.*

3.3.3 Conception de l'algorithme 2

Cet algorithme est basé sur le principe du plus grand en premier, c'est-à-dire que la trame sera remplie en premier lieu par la rafale de données qui occupera le plus grand nombre de tranches puis le suivant jusqu'au remplissage final de la trame.

À la figure 3.9, la rafale de données 3, *DL Burst 3*, occupera le plus grand nombre de tranches suivi par le DL Burst 2. Ensuite, DL Burst 5 occupera un nombre de tranches inférieur et ainsi de suite comme le montre la relation d'inégalité suivante :

DL Burst3 > DL Burst2 > DL Burst5 > DL Burst1 > DL Burst4

Alors, la trame sera remplie en premier lieu par le DL Burst3, ensuite par DL Burst2, puis par DL Burst5, ensuite par DL Burst1 et enfin par DL Burst4.

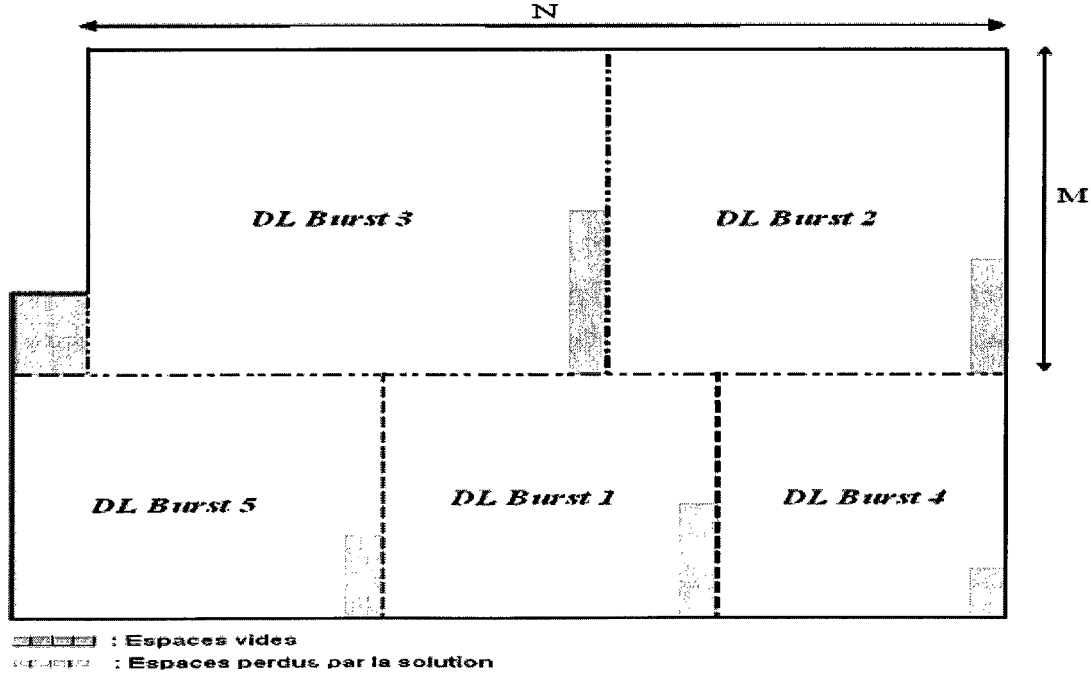

Figure 3.9 ***L'illustration de la sous trame de la voie descendante où l'algorithme 2 est utilisé.***

L'algorithme suivant montre la procédure à suivre pour remplir la trame :

Tant que ***(la somme des tailles des paquets reçus est inférieure à la taille de la région disponible pour les DL-Burst)***

Faire :

- *Recevoir les paquets classés par l'ordonnanceur.*
- *Classer ces paquets par ensemble de Burst #i (i = 1 à Np).*

Fin Tant que.

- *Mémoriser l'indice du dernier paquet reçu.*
- *Classer les DL Burst par ordre croissant.*
- *Insérer les 2 premiers de telle sorte qu'ils occupent tous les symboles OFDMA disponibles*
- *Insérer les 3 DL Bursts restants de telle sorte qu'ils occupent tous les symboles OFDMA disponibles et les sous-canaux restants.*

Faire :

- *Recevoir le paquet suivant (indice +1),*
- *Déterminer son profil* ***i****,*
- *Insérer le paquet dans la rafale #****i*** *correspondant,*
- *Si le paquet est plus grand que l'espace libre,*
  le paquet doit être fragmenté.
- *Fin Si.*
- *Si le dernier paquet ne s'insère pas dans son DL Burst #****i****,*
  Fin Faire. (Sortir de la boucle)
- *Sinon : indice ++*
-

3.4 Résultats

Dans cette section, les résultats de simulation des deux algorithmes proposés ainsi que l'algorithme conçu spécifiquement pour la trame de 1.25 MHz, FFT de 128 points. Les résultats concernent le taux de remplissage et le temps d'exécution de chaque algorithme.

Les simulations se feront pour un système OFDMA extensible de 20 MHz, 10 MHz, 5 MHz et 1.25 MHz. Aussi, pour des trames de 20 ms, 10 ms et 5 ms.

3.4.1 Environnement de simulation

La densité de probabilité des arrivées des paquets suit le modèle de Poisson et leurs tailles suivent une distribution exponentielle [15,16].

La distribution statistique des profils des rafales de données suit une distribution uniforme des nombres de 1 à 8. La fonction *randsrc(1000,1,[1 2 3 4 5 6 7 8])* sur Matlab a été utilisée pour cette fin.

Les paramètres numériques suivants ont été utilisés dans la simulation des deux algorithmes :

- La taille moyenne des paquets est de 200 octets.
- Le profil des rafales est distribué uniformément sur 1000 échantillons.
- Chaque algorithme est testé 500 fois pour extraire leurs performances en calculant la moyenne sur les 500 résultats obtenus.

Les profils de rafales «Bursts profil» sont :

- Profil 1 : QPSK-1/2.
- Profil 2 : QPSK-3/4.
- Profil 3 : 16QAM-1/2.
- Profil 4 : 16QAM-2/3.
- Profil 5 : 64QAM-2/3.
- Profil 6 : QPSK-2/3.
- Profil 7 : 16QAM-3/4.
- Profil 8 : 64QAM-1/2.

3.4.2 Algorithme 1

Le premier algorithme est basé sur la combinaison des différents *DL Bursts* possibles qui seront présents dans la trame pour déterminer de quelle manière chaque *DL Burst* sera inséré dans la trame.

Les simulations se feront pour un système OFDMA extensible de 20 MHz, 10 MHz et 5 MHz. Aussi, pour des trames de 20 ms, 10 ms et 5 ms. Les résultats seront présentés selon le rapport de la sous trame DL par rapport à la sous trame UL (*DL:UL*). Le rapport *DL:UL* représente le nombre de symboles OFDMA de la sous trame de la voie descendante par rapport au nombre de symboles OFDMA de la sous trame de la voie montante. Les tableaux 3.2 à 3.9 montrent les résultats associés à l'algorithme proposé.

Rapport DL:UL = 3:1 :

Tableau 3.2

Résultats obtenus pour une trame en mode PUSC

		5 ms	***10 ms***	***20 ms***
20 MHz	***Taux de remplissage (%)***	93.13	94.87	97.28
	Temps d'exécution (ms)	8.05	8.3	8.5
10 MHz	***Taux de remplissage (%)***	92.43	94.47	96.68
	Temps d'exécution (ms)	7.93	8.06	8.3
5 MHz	***Taux de remplissage (%)***	91.87	94.03	96.02
	Temps d'exécution (ms)	7.85	7.95	8.07

Tableau 3.3

Résultats obtenus pour une trame de 20 MHz en mode FUSC

	5 ms	***10 ms***	***20 ms***
Taux de remplissage (%)	92.57	94.48	96.66
Temps d'exécution (ms)	7.98	8.13	8.41

Rapport DL:UL = 2:1 :

Tableau 3.4

Résultats obtenus pour une trame en mode PUSC

		5 ms	*10 ms*	*20 ms*
20 MHz	*Taux de remplissage (%)*	92.34	94.42	96.57
	Temps d'exécution (ms)	8.03	8.24	8.45
10 MHz	*Taux de remplissage (%)*	92.23	94.29	96.45
	Temps d'exécution (ms)	7.79	8.01	8.27
5 MHz	*Taux de remplissage (%)*	91.76	93.78	96.01
	Temps d'exécution (ms)	7.61	7.78	8.08

Tableau 3.5

Résultats obtenus pour une trame de 20 MHz en mode FUSC

	5 ms	*10 ms*	*20 ms*
Taux de remplissage (%)	92.04	94.18	96.42
Temps d'exécution (ms)	7.89	8.17	8.35

Rapport DL:UL = 3:2 :

Tableau 3.6

Résultats obtenus pour une trame en mode PUSC

		5 ms	*10 ms*	*20 ms*
20 MHz	*Taux de remplissage (%)*	91.99	94.01	96.4
	Temps d'exécution (ms)	7.92	8.18	8.39
10 MHz	*Taux de remplissage (%)*	91.02	93.06	95.43
	Temps d'exécution (ms)	7.71	8.01	8.21
5 MHz	*Taux de remplissage (%)*	90.63	92.5	95.25
	Temps d'exécution (ms)	7.53	7.84	8.03

Tableau 3.7

Résultats obtenus pour une trame de 20 MHz en mode FUSC

	5 ms	*10 ms*	*20 ms*
Taux de remplissage (%)	90.17	93.42	95.21
Temps d'exécution (ms)	7.83	8.09	8.28

Rapport DL:UL = 1:1 :

Tableau 3.8

Résultats obtenus pour une trame en mode PUSC

		5 ms	*10 ms*	*20 ms*
20 MHz	*Taux de remplissage (%)*	90.8	92.51	95.27
	Temps d'exécution (ms)	7.8	8.11	8.31
10 MHz	*Taux de remplissage (%)*	90.45	92.2	95.12
	Temps d'exécution (ms)	7.65	7.95	8.09
5 MHz	*Taux de remplissage (%)*	89.91	91.71	94.75
	Temps d'exécution (ms)	7.51	7.82	7.95

Tableau 3.9

Résultats obtenus pour une trame de 20 MHz en mode FUSC

	5 ms	*10 ms*	*20 ms*
Taux de remplissage (%)	89.45	92.07	95.04
Temps d'exécution (ms)	7.71	8.03	8.16

Les figures 3.10 et 3.11 représentent le taux et le temps de remplissage en fonction du rapport DL:UL d'une trame de 20 MHz et 10 ms pour les deux modes PUSC et FUSC.

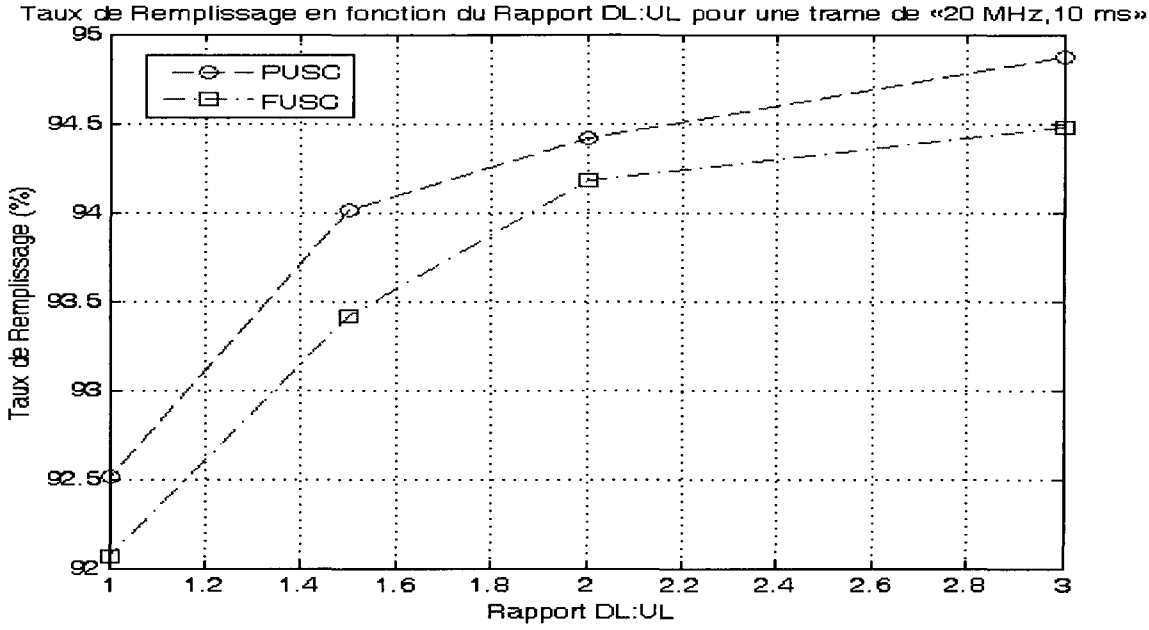

Figure 3.10 ***Taux de remplissage en fonction du rapport DL:UL.***

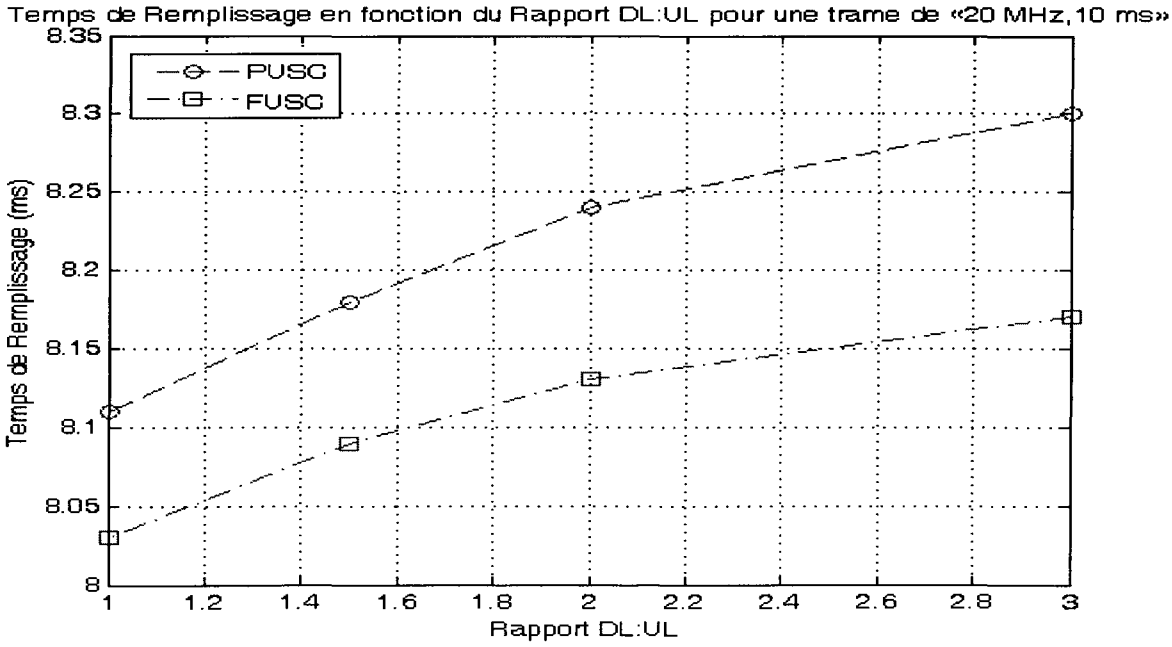

Figure 3.11 ***Temps de remplissage en fonction du rapport DL:UL.***

Les figures 3.12 et 3.13 représentent le taux et le temps de remplissage en fonction de la largeur de bande d'une trame de 10 ms.

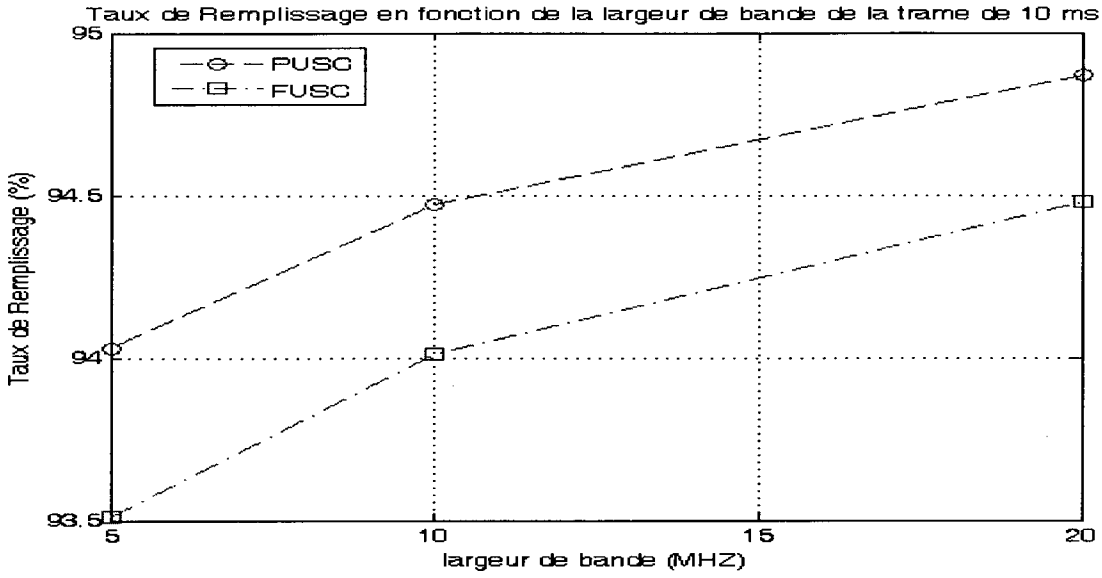

Figure 3.12 ***Taux de remplissage en fonction de la largeur de bande d'une trame de 10 ms.***

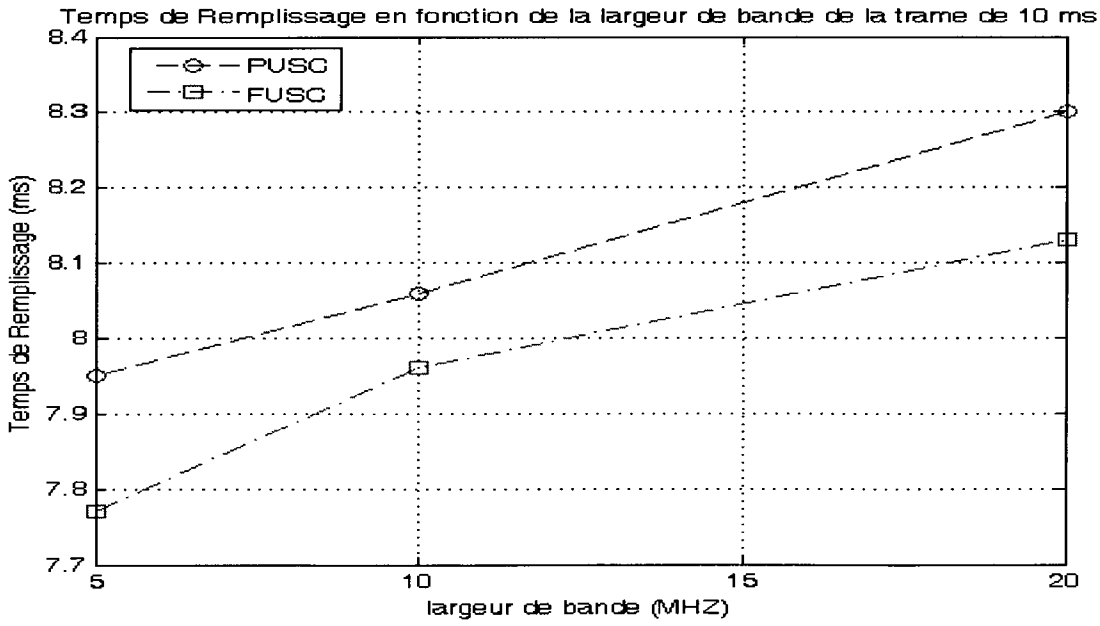

Figure 3.13 ***Temps de remplissage en fonction de la largeur de bande d'une trame de 10 ms.***

Les figures 3.14 et 3.15 représentent le taux et le temps de remplissage en fonction de la longueur d'une trame en ms.

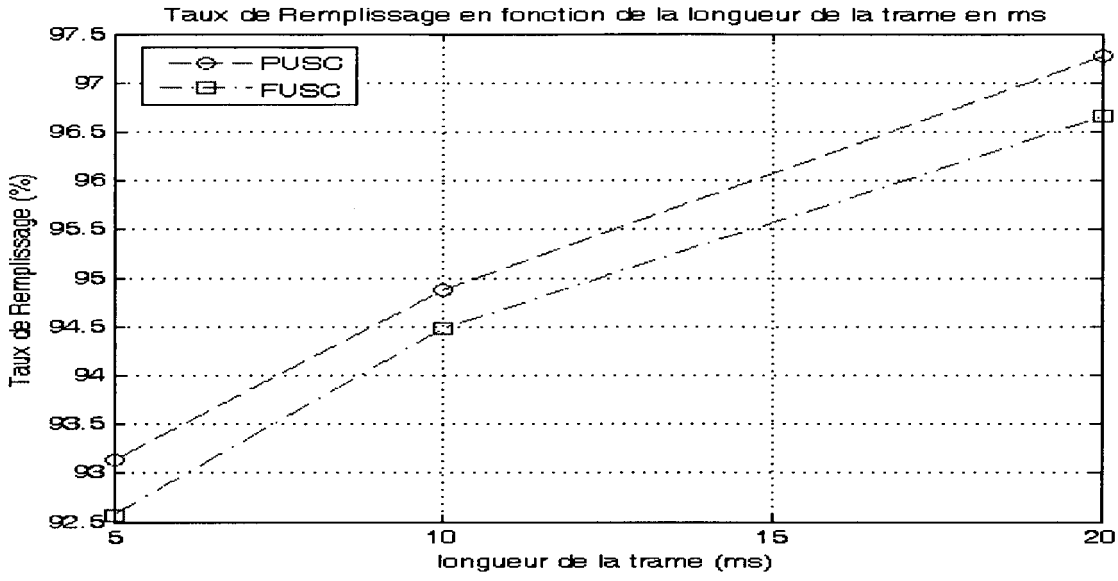

Figure 3.14 ***Taux de remplissage en fonction de la longueur d'une trame en ms.***

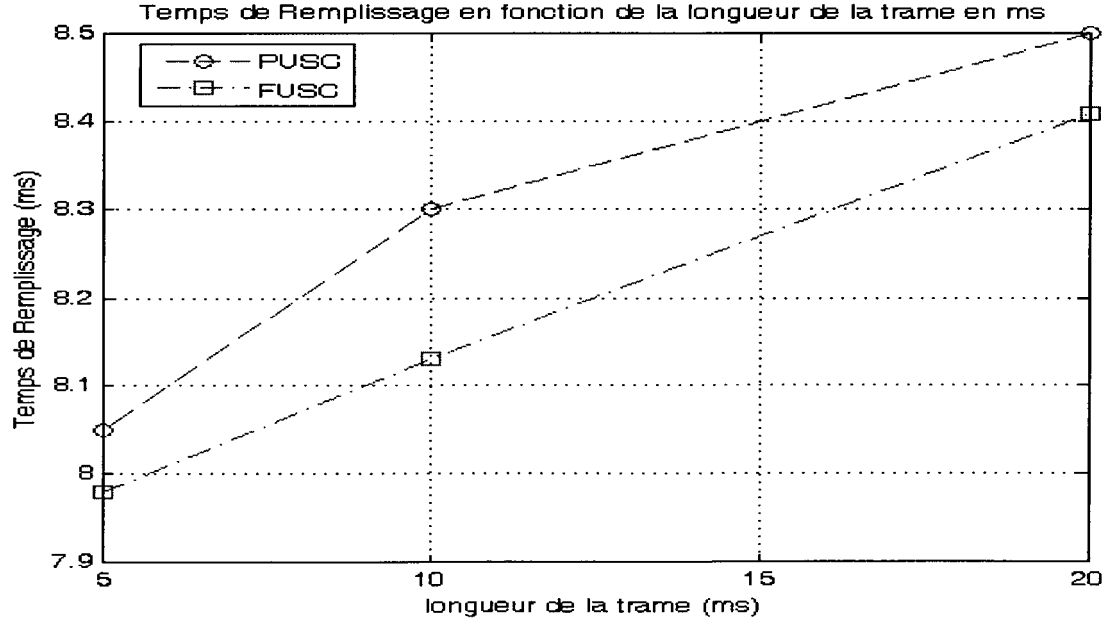

Figure 3.15 ***Temps de remplissage en fonction de la longueur d'une trame en ms.***

3.4.3 Algorithme 2

Le deuxième algorithme est basé sur le principe du plus grand en premier, c'est-à-dire que la trame sera remplie en premier lieu par le DL Burst qui occupera le plus grand nombre d'intervalles puis le suivant jusqu'au remplissage final de la trame.

Les simulations se feront pour un système OFDMA extensible de 20 MHz, 10 MHz et 5 MHz. Aussi, pour des trames de 20 ms, 10 ms et 5 ms. Les résultats seront présentés selon le rapport de la sous trame DL par rapport à la sous trame UL (*DL:UL*). Les tableaux 3.10 à 3.17 montrent les résultats associés à l'algorithme proposé.

Rapport DL:UL = 3:1 :

Tableau 3.10

Résultats obtenus pour une trame en mode PUSC

		5 ms	***10 ms***	***20 ms***
20 MHz	***Taux de remplissage (%)***	94.99	97.9	99.03
	Temps d'exécution (ms)	1.3	1.5	2
10 MHz	***Taux de remplissage (%)***	94.54	97.8	98.98
	Temps d'exécution (ms)	1.15	1.37	1.55
5 MHz	***Taux de remplissage (%)***	93.88	97.37	98.84
	Temps d'exécution (ms)	1.02	1.16	1.29

Tableau 3.11

Résultats obtenus pour une trame de 20 MHz en mode FUSC

	5 ms	***10 ms***	***20 ms***
Taux de remplissage (%)	94.41	97.7	98.97
Temps d'exécution (ms)	1.48	1.7	2.2

Rapport DL:UL = 2:1 :

Tableau 3.12

Résultats obtenus pour une trame en mode PUSC

		5 ms	***10 ms***	***20 ms***
20 MHz	***Taux de remplissage (%)***	94.22	97.59	98.91
	Temps d'exécution (ms)	1.15	1.4	1.7
10 MHz	***Taux de remplissage (%)***	93.61	97.42	98.81
	Temps d'exécution (ms)	1.06	1.3	1.48
5 MHz	***Taux de remplissage (%)***	92.85	97	98.68
	Temps d'exécution (ms)	0.87	1.02	1.16

Tableau 3.13

Résultats obtenus pour une trame de 20 MHz en mode FUSC

	5 ms	***10 ms***	***20 ms***
Taux de remplissage (%)	93.33	97.28	98.82
Temps d'exécution (ms)	1.23	1.59	2.03

Rapport DL:UL = 3:2 :

Tableau 3.14

Résultats obtenus pour une trame en mode PUSC

		5 ms	***10 ms***	***20 ms***
20 MHz	***Taux de remplissage (%)***	93.31	97.13	98.77
	Temps d'exécution (ms)	1.08	1.32	1.61
10 MHz	***Taux de remplissage (%)***	92.4	97.07	98.69
	Temps d'exécution (ms)	0.99	1.21	1.37
5 MHz	***Taux de remplissage (%)***	91.52	96.53	98.49
	Temps d'exécution (ms)	0.9	1.12	1.28

Tableau 3.15

Résultats obtenus pour une trame de 20 MHz en mode FUSC

	5 ms	***10 ms***	***20 ms***
Taux de remplissage (%)	90.30	96.66	98.34
Temps d'exécution (ms)	1.2	1.42	1.7

Rapport DL:UL = 1:1 :

Tableau 3.16

Résultats obtenus pour une trame en mode PUSC

		5 ms	***10 ms***	***20 ms***
20 MHz	***Taux de remplissage (%)***	92.22	96.5	98.50
	Temps d'exécution (ms)	1.01	1.23	1.5
10 MHz	***Taux de remplissage (%)***	91.71	96.26	98.39
	Temps d'exécution (ms)	0.92	1.18	1.28
5 MHz	***Taux de remplissage (%)***	90.91	95.65	98.15
	Temps d'exécution (ms)	0.83	1.09	1.17

Tableau 3.17

Résultats obtenus pour une trame de 20 MHz en mode FUSC

	5 ms	*10 ms*	*20 ms*
Taux de remplissage (%)	89.95	96.14	98.34
Temps d'exécution (ms)	1.1	1.29	1.58

Les figures 3.16 et 3.17 représentent le taux et le temps de remplissage en fonction du rapport DL:UL d'une trame de 20 MHz et 10 ms pour les deux modes PUSC et FUSC.

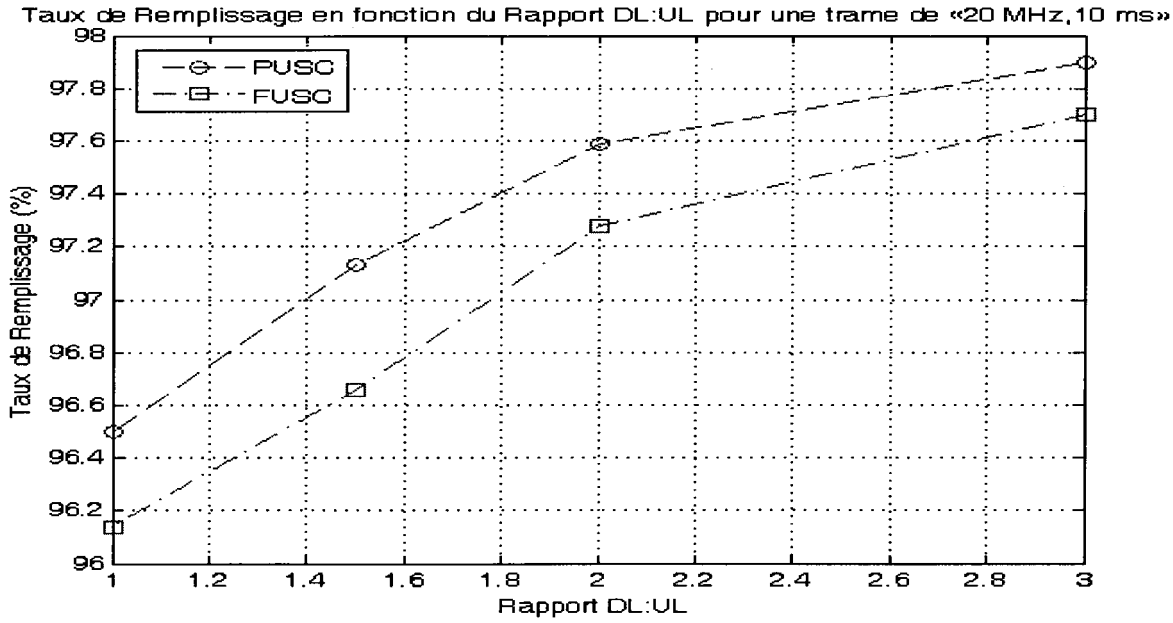

Figure 3.16 ***Taux de remplissage en fonction du rapport DL:UL.***

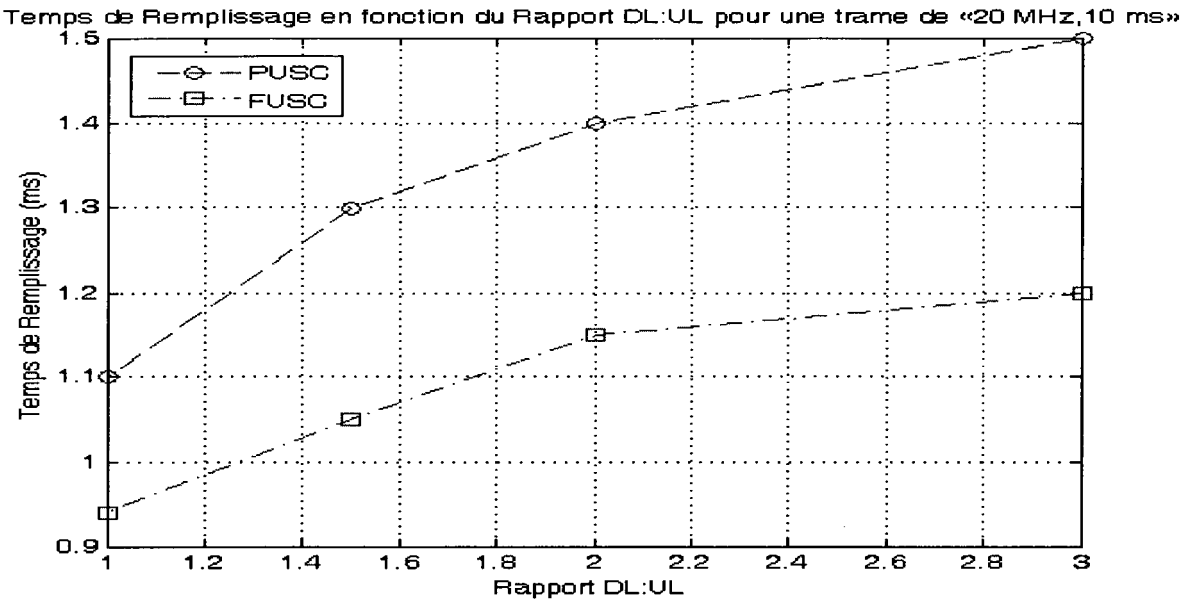

Figure 3.17 ***Temps de remplissage en fonction du rapport DL:UL.***

Les figures 3.18 et 3.19 représentent le taux et le temps de remplissage en fonction de la largeur de bande d'une trame de 10 ms.

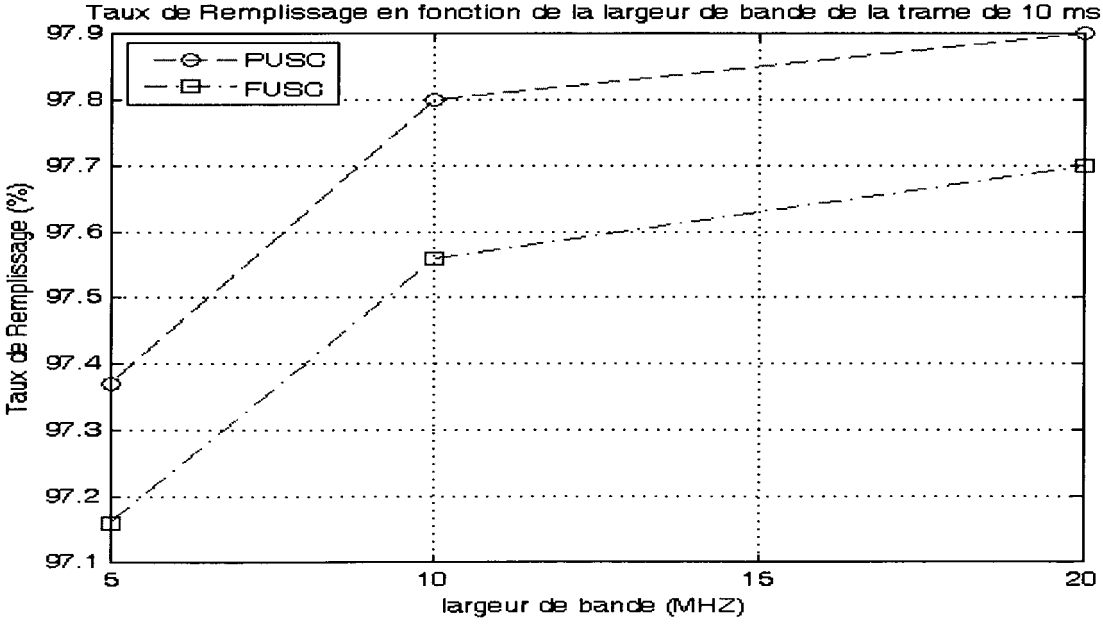

Figure 3.18 ***Taux de remplissage en fonction de la largeur de bande d'une trame de 10 ms.***

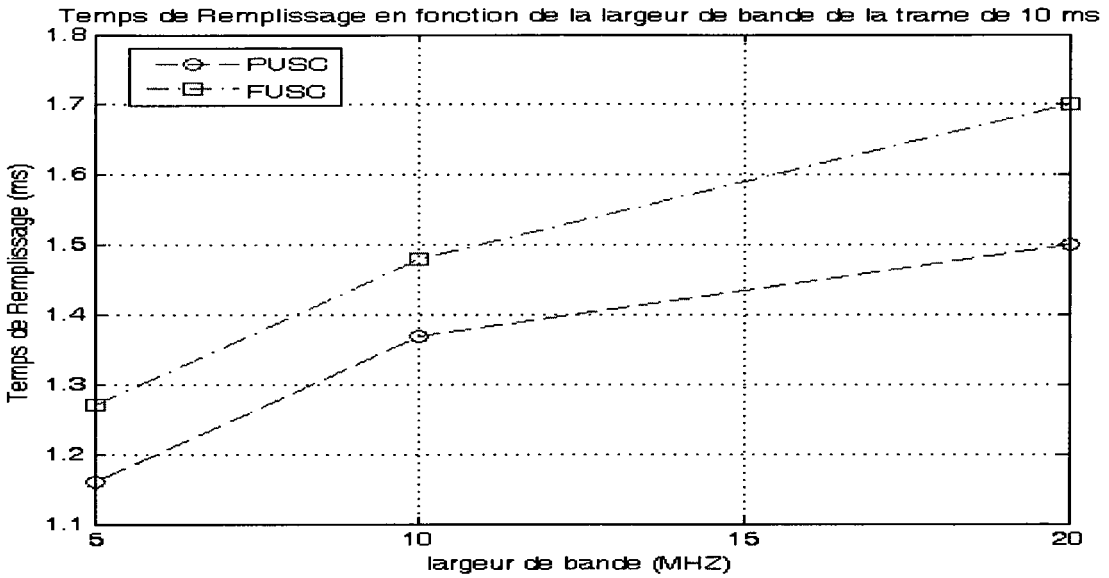

Figure 3.19 ***Temps de remplissage en fonction de la largeur de bande d'une trame de 10 ms.***

Les figures 3.20 et 3.21 représentent le taux et le temps de remplissage en fonction de la longueur d'une trame en ms.

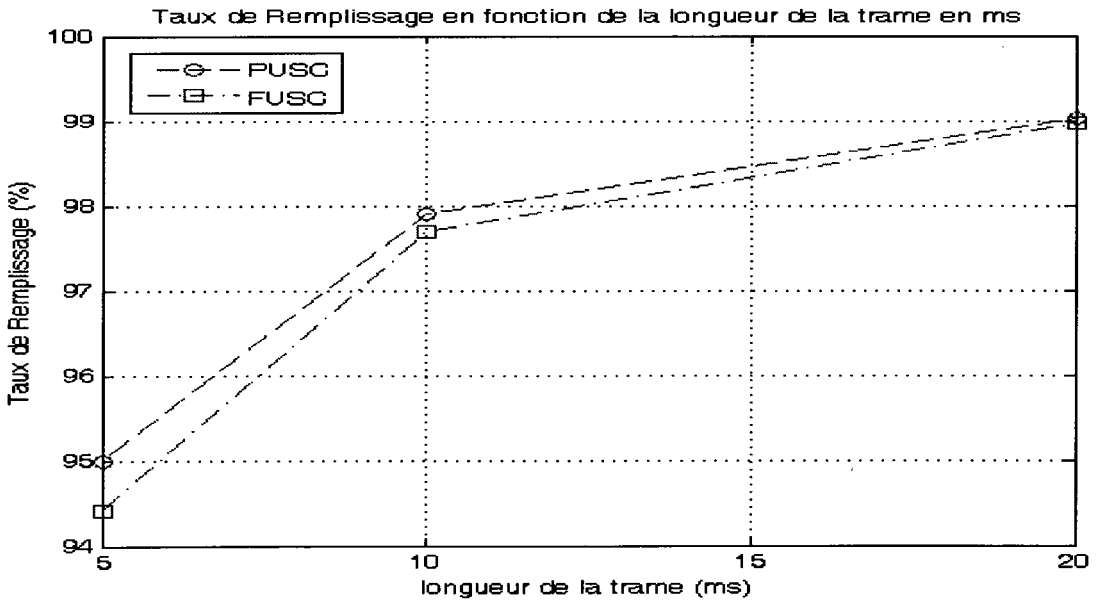

Figure 3.20 ***Taux de remplissage en fonction de la longueur d'une trame en ms.***

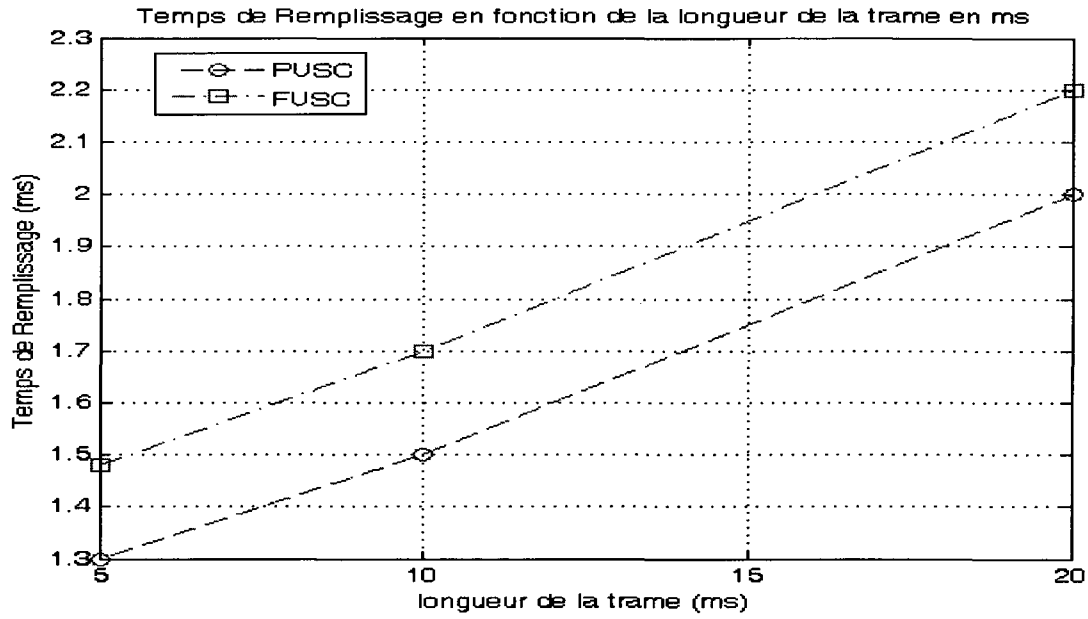

Figure 3.21 ***Temps de remplissage en fonction de la longueur d'une trame en ms.***

3.4.4 Algorithme de la trame de 1.25 MHz

Les deux algorithmes présentés précédemment peuvent bien être utiles pour déterminer de quelle manière chaque *DL Burst* sera inséré dans la trame 1.25 MHz. Or, puisque cette dernière n'a que deux ou trois sous canaux en mode FUSC et PUSC respectivement, l'utilisation de ces algorithmes n'est pas convenable.

Alors, un processus de mise en trame simple a été développé. Ce processus consiste à répartir les rafales de données horizontalement, c'est-à-dire selon l'axe des symboles OFDMA. Toutes les rafales de données vont occuper toutes la bande de fréquences de 1.25MHz (tous les sous canaux présents).

Les figures 3.10 et 3.11 montrent une sous trame de 1.25 MHz de la voie descendante en mode PUSC et FUSC respectivement. La trame en mode FUSC ne contient que 2 sous canaux tandis que la trame en mode PUSC en contient 3.

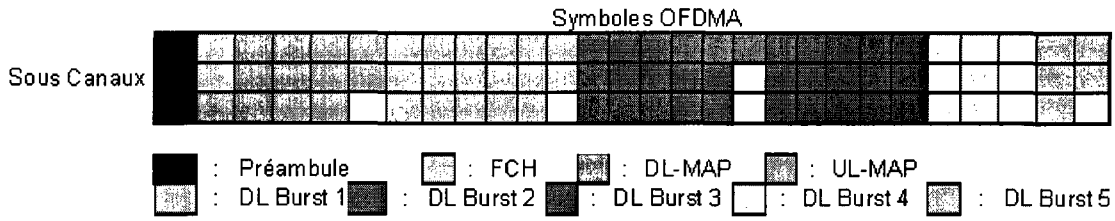

Figure 3.22 ***Sous trame 1.25 MHz de la voie descendante en mode PUSC.***

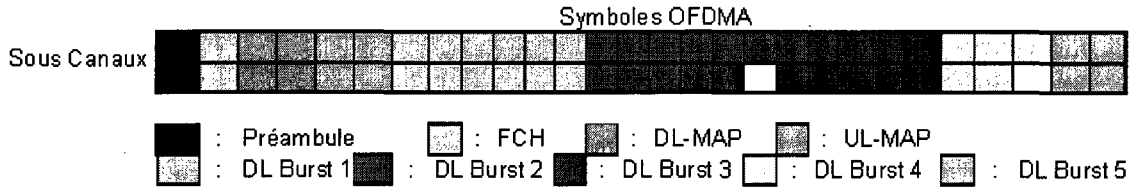

Figure 3.23 ***Sous trame 1.25 MHz de la voie descendante en mode FUSC.***

Les résultats seront présentés selon le rapport de la sous trame DL par rapport à la sous trame UL (*DL:UL*).

Les tableaux 3.18 à 3.21 montrent les résultats associés à l'algorithme proposé pour une trame en mode PUSC.

Tableau 3.18

Résultats obtenus pour une trame de 1.25 MHz pour un ratio DL:UL=3:1

	5 ms	***10 ms***	***20 ms***
Taux de remplissage (%)	83.75	90.50	94.60
Temps d'exécution (ms)	0.186	0.22	0.248

Tableau 3.19

Résultats obtenus pour une trame de 1.25 MHz pour un ratio DL:UL=2 :1

	5 ms	***10 ms***	***20 ms***
Taux de remplissage (%)	84.64	90.80	94.95
Temps d'exécution (ms)	0.252	0.316	0.376

Tableau 3.20

Résultats obtenus pour une trame de 1.25 MHz pour un ratio DL:UL=3:2

	5 ms	***10 ms***	***20 ms***
Taux de remplissage (%)	84.84	90.95	95.26
Temps d'exécution (ms)	0.316	0.35	0.385

Tableau 3.21

Résultats obtenus pour une trame de 1.25 MHz pour un ratio DL:UL=1:1

	5 ms	***10 ms***	***20 ms***
Taux de remplissage (%)	85.11	91.22	95.68
Temps d'exécution (ms)	0.318	0.362	0.391

Aussi, les tableaux 3.22 à 3.25 montrent les résultats associés à l'algorithme proposé pour une trame en mode FUSC.

Tableau 3.22

Résultats obtenus pour une trame de 1.25 MHz pour un ratio DL:UL=3:1

	5 ms	***10 ms***	***20 ms***
Taux de remplissage (%)	89.60	93.92	96.70
Temps d'exécution (ms)	0.222	0.25	0.284

Tableau 3.23

Résultats obtenus pour une trame de 1.25 MHz pour un ratio DL:UL=2 :1

	5 ms	***10 ms***	***20 ms***
Taux de remplissage (%)	90.10	94.15	96.82
Temps d'exécution (ms)	0.23	0.314	0.33

Tableau 3.24

Résultats obtenus pour une trame de 1.25 MHz pour un ratio DL:UL=3:2

	5 ms	***10 ms***	***20 ms***
Taux de remplissage (%)	90.65	94.52	96.98
Temps d'exécution (ms)	0.28	0.316	0.342

Tableau 3.25

Résultats obtenus pour une trame de 1.25 MHz pour un ratio DL:UL=1:1

	5 ms	*10 ms*	*20 ms*
Taux de remplissage (%)	91.23	94.86	97.15
Temps d'exécution (ms)	0.291	0.326	0.333

Les figures 3.24 et 3.25 représentent le taux et le temps de remplissage en fonction du rapport DL:UL d'une trame de 1.25 MHz et 10 ms pour les deux modes PUSC et FUSC.

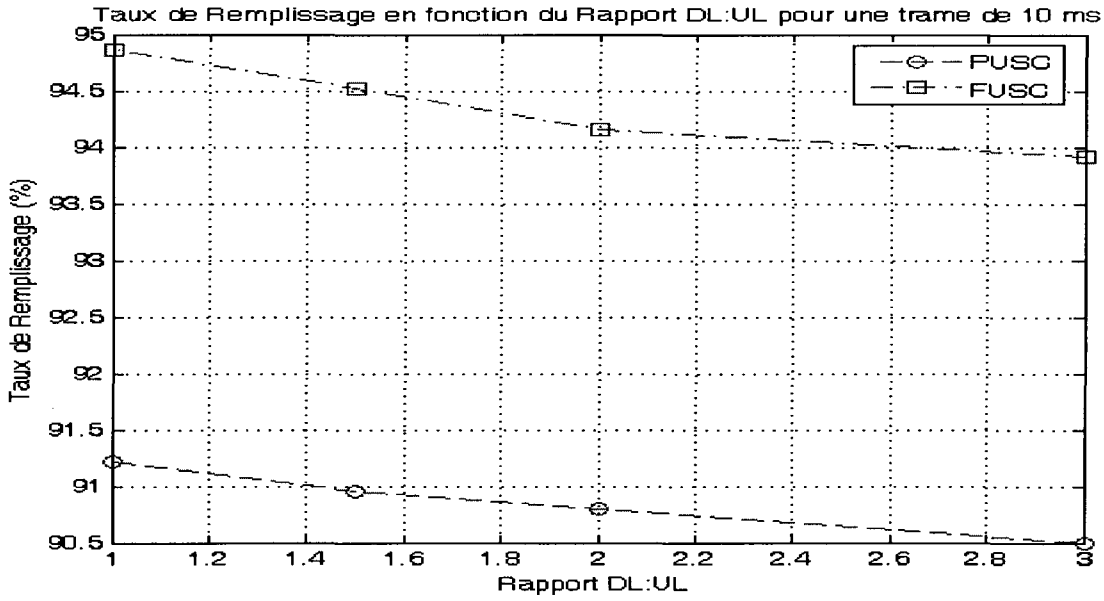

Figure 3.24 ***Taux de remplissage en fonction du rapport DL:UL.***

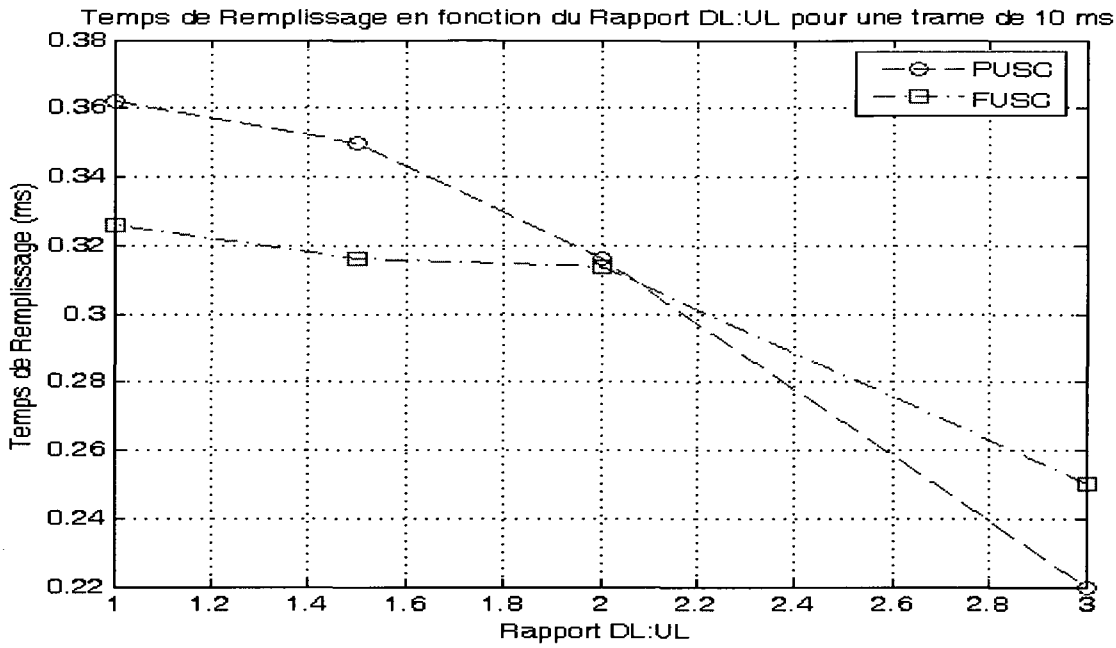

Figure 3.25 ***Temps de remplissage en fonction du rapport DL:UL.***

Les figures 3.26 et 3.27 représentent le taux et le temps de remplissage en fonction de la longueur d'une trame en ms.

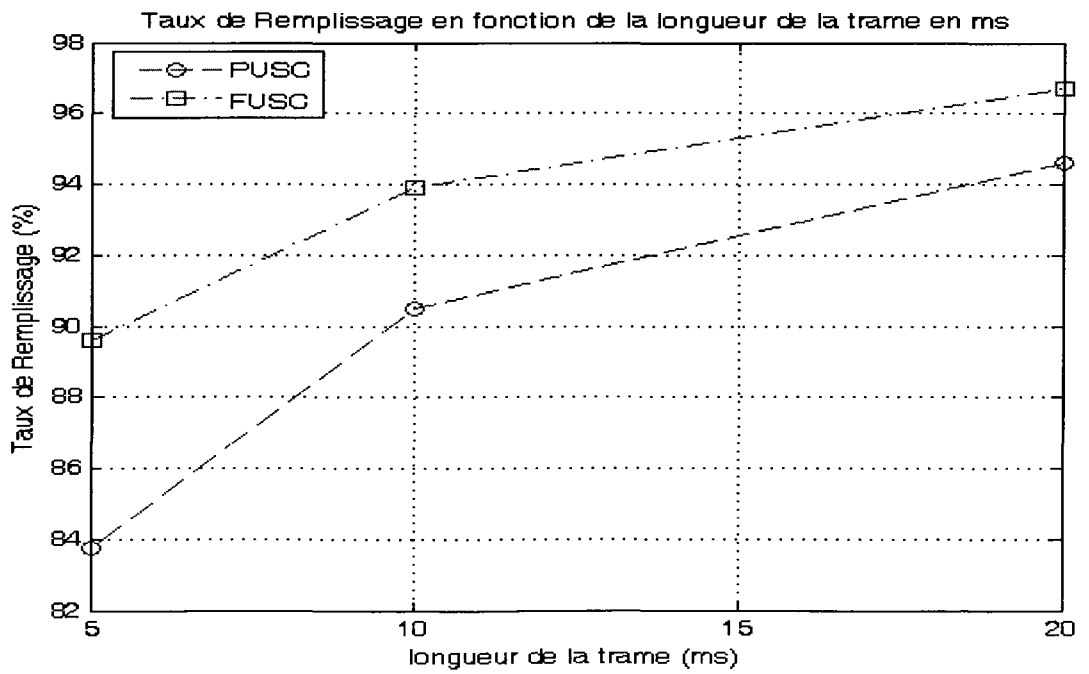

Figure 3.26 ***Taux de remplissage en fonction de la longueur d'une trame en ms.***

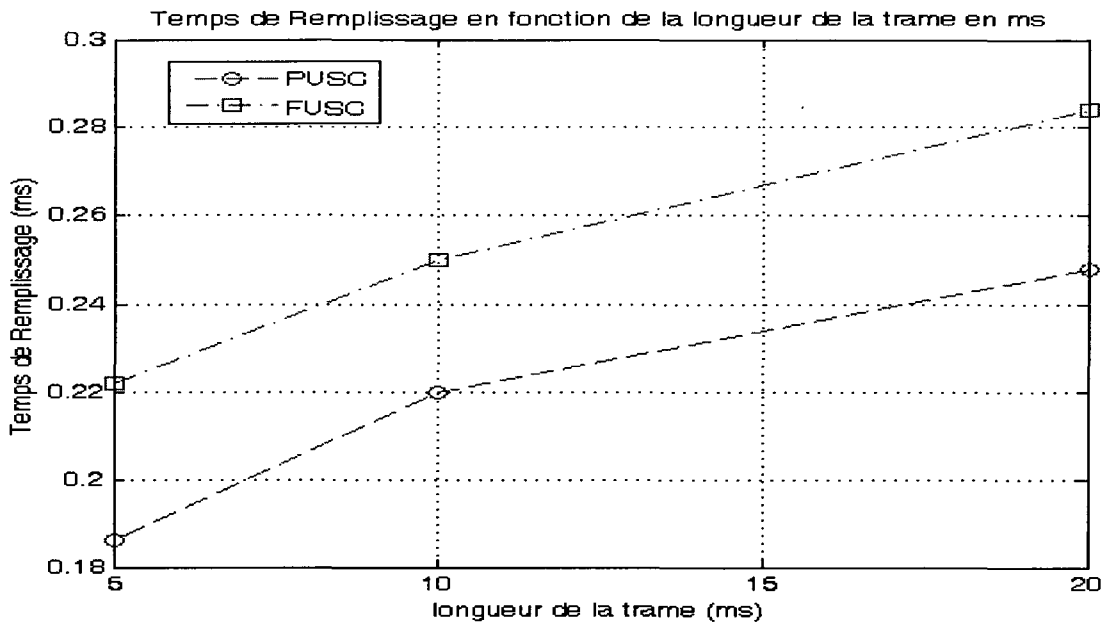

Figure 3.27 ***Temps de remplissage en fonction de la longueur d'une trame en ms.***

3.4.5 Simulation pour des cas extrêmes

Pour mieux analyser le comportement des deux algorithmes proposés, des simulations des deux algorithmes pour des cas extrêmes sont effectuées. Ces simulations se feront seulement pour des trames PUSC et FUSC et seulement pour un ratio DL:UL typique de 3:1 ainsi que pour :

- des paquets IP de 1500 octets,
- des paquets IP de 21 octets.

Lors de la simulation des paquets de 21 octets, le taux de remplissage a été considérablement amélioré mais le temps d'exécution a aussi augmenté.

Lors de la simulation des paquets IP de 1500 octets, pour une trame en mode PUSC ayant un rapport typique DL:UL de 3:1, il faut toujours segmenter ce type de paquets. La trame de 1.25 MHz n'a supporté qu'un ou deux paquets et le taux de remplissage a augmenté et le temps d'exécution a diminué puisqu'il n'y a pas beaucoup de paquets à manipuler. Les autres trames ont supporté plus qu'un paquet, mais la segmentation a été nécessaire pour remplir

d'une façon optimale les trames en question. Le taux de remplissage a diminué et le temps d'exécution a aussi diminué puisque le nombre de paquets à manipuler a diminué.

Le tableau 3.26 présente le nombre de paquets, d'une taille de 1500 octets, supporté par chaque type de trame ainsi que le taux de remplissage pour chaque cas. Par exemple un nombre de paquets de 0.28 représente un paquet de 420 octets et un nombre de paquets de 2.3 représente 2 paquets de 1500 octets et un troisième paquet de 450 octets.

Tableau 3.26

Le nombre de paquets de 1500 octets supporté par chaque trame ainsi que le taux de remplissage correspondent

	Nombre de paquets			**Taux de remplissage (%)**		
Trame	**5 ms**	**10 ms**	**20 ms**	**5 ms**	**10 ms**	**20 ms**
1.25 MHz	0.28	0.6	1.16	98.6	98.6	98.2
5 MHz	1.15	2.3	4.6	97.75	97.23	97.17
10 MHz	2.3	4.75	9.3	97.15	97.03	96.86
20 MHz	4.6	9.5	18.6	97.11	96.89	96.73

3.5 Comparaison des deux algorithmes

Dans cette section, une comparaison des deux algorithmes sera présentée. Les deux critères qui seront comparés sont le taux de remplissage de la trame ainsi que le temps d'exécution de chaque algorithme. Les résultats seront présentés sous forme de tableaux pour une trame typique possédant un rapport DL:UL de 3:1.

3.5.1 Taux de remplissage de chaque algorithme

Une présentation comparative du taux de remplissage de chaque algorithme pour une trame en mode PUSC et en mode FUSC sera faite.

Les tableaux 3.27 à 3.30 contiennent les résultats du taux de remplissage des deux algorithmes.

Tableau 3.27

Le taux de remplissage d'une trame en mode PUSC de 20 ms

	Algorithme 1	Algorithme 2
5 MHz	96.02	98.84
10 MHz	96.68	98.98
20 MHz	97.28	99.03

Tableau 3.28

Le taux de remplissage d'une trame en mode PUSC de 10 ms

	Algorithme 1	Algorithme 2
5 MHz	94.03	97.37
10 MHz	94.47	97.8
20 MHz	94.87	97.9

Tableau 3.29

Le taux de remplissage d'une trame en mode PUSC de 5 ms

	Algorithme 1	Algorithme 2
5 MHz	91.87	93.88
10 MHz	92.43	94.54
20 MHz	93.13	94.99

Tableau 3.30

Le taux de remplissage d'une trame en mode FUSC de 20 MHz

	Algorithme 1	Algorithme 2
5 ms	92.57	94.41
10 ms	94.48	97.7
20 ms	96.66	98.97

Les figures 3.28 et 3.29 illustrent la différence entre les deux algorithmes. La figure 3.28 représente le taux de remplissage des deux algorithmes en mode PUSC en fonction de la largeur de bande de la trame pour un rapport DL:UL de 3:1.

La figure 3.29 représente le taux de remplissage des deux algorithmes en mode FUSC en fonction de la longueur de la trame pour un rapport DL:UL de 3:1 et une largeur de bande de 20 MHZ.

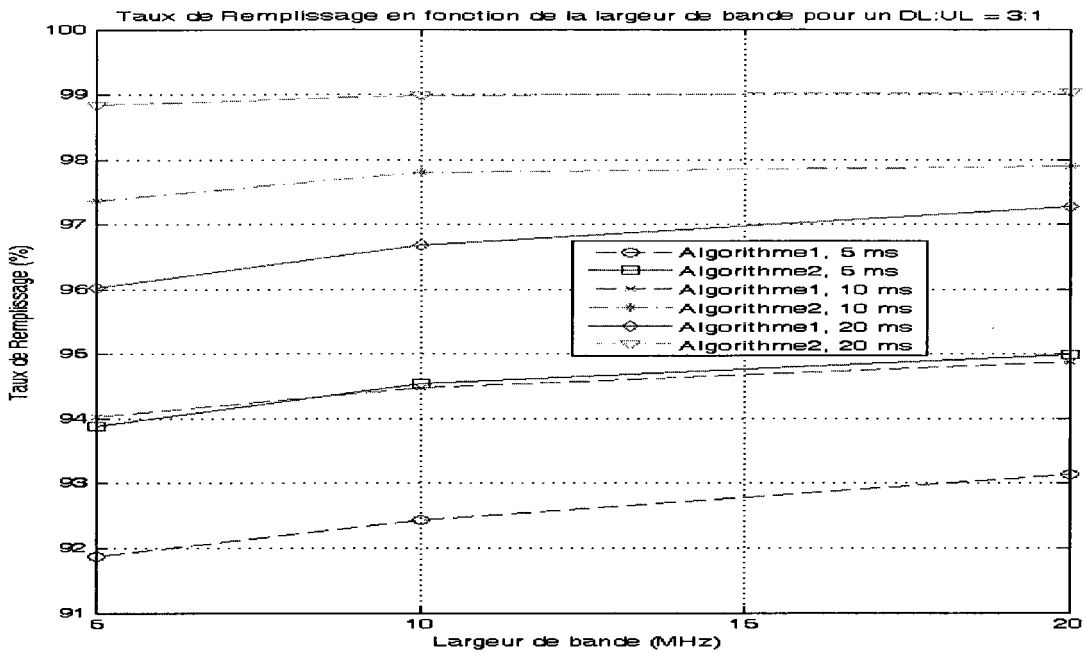

Figure 3.28 ***Le taux de remplissage des deux algorithmes en mode PUSC en fonction de largeur de bande de la trame.***

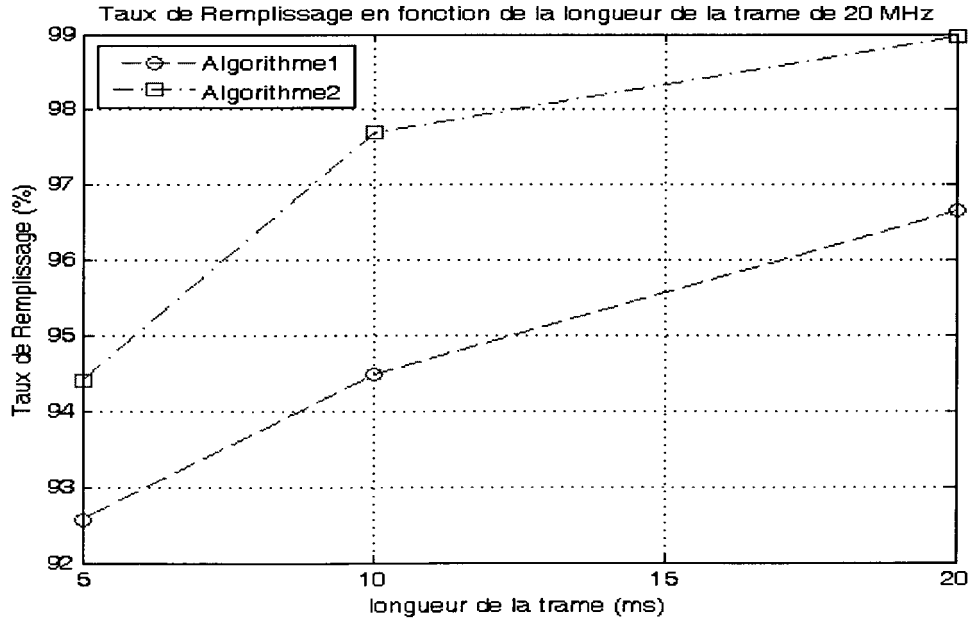

Figure 3.29 ***Le taux de remplissage des deux algorithmes en mode FUSC en fonction de la longueur de la trame de 20 MHz.***

3.5.2 Temps d'exécution de chaque algorithme

Afin de compléter l'évaluation des deux algorithmes, une présentation comparative du temps d'exécution de chaque algorithme pour une trame en mode PUSC et en mode FUSC sera faite.

Les tableaux 3.31 à 3.34 contiennent les résultats du temps d'exécution en ms des deux algorithmes.

Tableau 3.31

Le temps d'exécution d'une trame en mode PUSC 20 ms

	Algorithme 1	Algorithme 2
5 MHz	8.07	1.29
10 MHz	8.3	1.55
20 MHz	8.5	2

Tableau 3.32

Le temps d'exécution d'une trame en mode PUSC 10 ms

	Algorithme 1	Algorithme 2
5 MHz	7.95	1.16
10 MHz	8.06	1.37
20 MHz	8.3	1.5

Tableau 3.33

Le temps d'exécution d'une trame en mode PUSC 5 ms

	Algorithme 1	Algorithme 2
5 MHz	7.85	1.02
10 MHz	7.93	1.15
20 MHz	8.05	1.3

Tableau 3.34

Le temps d'exécution d'une trame en mode FUSC de 20 MHz

	Algorithme 1	Algorithme 2
5 ms	7.98	1.48
10 ms	8.13	1.7
20 ms	8.41	2.2

Les figures 3.30 et 3.31 illustrent la différence entre les deux algorithmes. La figure 3.30 représente le temps de remplissage des deux algorithmes en mode PUSC en fonction de la largeur de bande de la trame pour un rapport DL:UL de 3:1.

La figure 3.31 représente le temps de remplissage des deux algorithmes en mode FUSC en fonction de la longueur de la trame pour un rapport DL:UL de 3:1 et une largeur de bande de 20 MHZ.

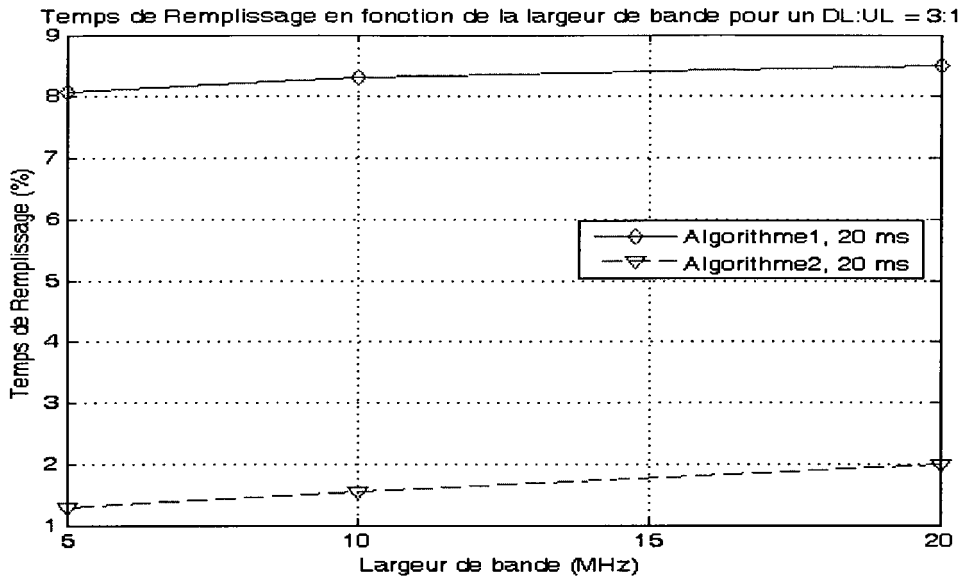

Figure 3.30 ***Le temps de remplissage des deux algorithmes en mode PUSC en fonction de largeur de bande de la trame.***

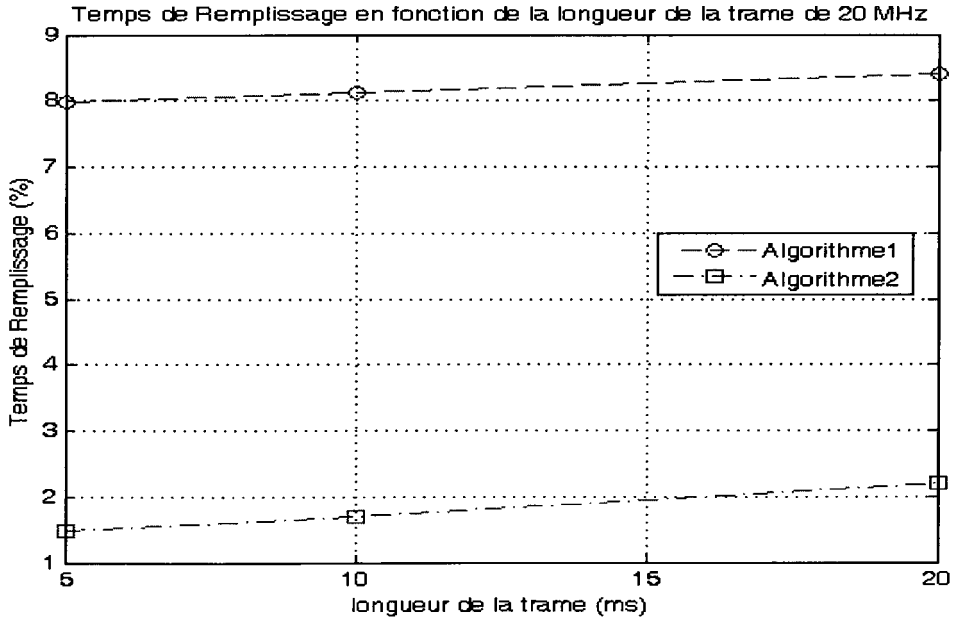

Figure 3.31 ***Le temps de remplissage des deux algorithmes en mode FUSC en fonction de la longueur de la trame de 20 MHz.***

3.6 Conclusion

En conclusion, la simulation des deux algorithmes permet de bien évaluer leurs performances ainsi que leur impact sur la trame WiMAX, IEEE 802.16e. Ces deux algorithmes ont été évalués pour différentes types de trames.

L'évaluation a été faite selon le mode d'utilisation partielle des sous-canaux «PUSC», d'utilisation entière des sous-canaux «FUSC», le rapport DL:UL, la largeur de bande de la trame et la longueur de la trame en nombre de symboles OFDMA.

Bien que la simulation des deux algorithmes se soit faite seulement pour une trame en mode temporel, TDD, l'extensibilité pour le mode fréquentiel, FDD est possible avec un minimum d'adaptation de ces algorithmes.

Le mode PUSC présente l'avantage de la granularité plus grande que celle du mode FUSC, ce qui avantage le premier mode. Au niveau des deux algorithmes, le deuxième algorithme est plus performant que le premier pour les deux modes en question.

Lors de l'évaluation des deux algorithmes tout en changeant le rapport DL:UL, lorsque la portion de la trame UL augmente, le taux de remplissage de la trame s'améliore pour les différents types de la trame. Par contre, lorsque la portion de la trame DL augmente, le taux de remplissage de la trame diminue pour les différents types de la trame. En effet, puisque la sous trame de la voie montante ne requiert pas que les rafales soient insérées dans des régions rectangulaires, les seules pertes possibles sont ceux des ajouts de bits de bourrage «Padding bits». Ce qui diminuera les espaces vides de la trame.

La variation de la largeur de bande de la trame n'a pas un impact trop important sur le temps de remplissage ou de traitement de chaque algorithme. Par contre, elle a un effet sur le taux de remplissage de la trame. Aussi, le deuxième algorithme est plus avantageux que le premier algorithme.

La taille de la trame en nombre de symboles OFDMA, en ms, a un effet sur les performances de chaque algorithme. En effet, lorsque la taille de la trame est grande, le taux de remplissage

s'améliore tandis que le temps d'exécution ou de traitement de chaque algorithme ne change pas d'une façon significative. Par contre, le deuxième algorithme est toujours plus performant que le premier.

Le taux de remplissage des deux algorithmes dépasse en général 92%. Ce qui est à première vue semble très intéressant. Or, une perte de 8% pour chaque algorithme intégré dans une station de base a un impact direct de 8% sur les revenus générés lorsque la demande est forte. Compte tenu de cet impacte financier direct, toute amélioration d'efficacité est souhaitable.

Le temps de remplissage ou de traitement a été présenté seulement pour comparer les deux algorithmes. En effet, le taux de remplissage dépend essentiellement du processeur intégré dans la station de base. Mais l'algorithme 2 est toujours plus rapide que l'algorithme 1 quelque soit la vitesse du processeur utilisé.

L'évaluation des ces algorithmes par rapport à d'autres algorithmes dans le même domaine n'est pas évidente à faire surtout avec l'absence de recherches dans le domaine des systèmes WiMAX mobile portant sur la mise en trame. Les algorithmes présentés peuvent être utilisés comme une base dans les recherches futures qui portent sur la mise en trame selon la norme IEEE 802.16e.

CONCLUSION

La norme IEEE 802.16e et particulièrement le WiMAX mobile est une solution large bande qui permet la convergence des réseaux fixe et mobile large bande à travers une technologie commune d'accès par radio large bande et une architecture de réseau flexible. L'interface sans-fil du WiMAX mobile adopte la technique de l'accès multiple par division orthogonale de la fréquence, OFDMA, pour une performance des multi trajets améliorée dans des environnements obstrués, NLOS.

Dans le chapitre 1, une revue de la norme IEEE 802.16e a été présentée. La couche physique de cette norme est basée sur la technique de modulation et le mode d'accès OFDMA. La couche MAC est définie pour supporter un très grand trafic de données avec une demande crête de débits très élevée. La trame de la norme IEEE 802.16e peut être en mode TDD ou FDD. Cependant, Le mode TDD est plus répandu grâce aux nombreux avantages qu'il représente.

Dans le chapitre 2, une présentation de la structure de la trame de la norme IEEE 802.16e ou WiMAX mobile a été introduite. Cette trame, divisée en deux sous-trames, est construite selon un repère bidimensionnel. Ce repère est caractérisé par deux axes : axe fréquentiel et axe temporel. L'axe fréquentiel est représenté par les sous-canaux et l'axe temporel est représenté par les symboles OFDMA. La sous-trame de la voie descendante contient des messages de contrôle ainsi que les données utiles de la voie descendante. Ces données sont regroupées en rafales et chaque rafale doit occuper une région rectangulaire de la sous-trame de la voie descendante. La sous-trame de la voie montante contient des messages d'information ainsi que les données utiles de la voie montante.

Dans le chapitre 3, deux algorithmes sont proposés pour la mise en trame des paquets de données utiles ainsi que les données de contrôle. Le premier algorithme est basé sur la combinaison des différents *DL Bursts* possibles qui seront présents dans la trame pour déterminer de quelle manière chaque *DL Burst* sera inséré dans la trame. Le deuxième algorithme est basé sur le principe du plus grand en premier, c'est-à-dire que la trame sera

remplie en premier lieu par le DL Burst qui occupera le plus grand nombre d'intervalles puis le suivant jusqu'au remplissage final de la trame.

En conclusion, la simulation des deux algorithmes permet de bien évaluer leurs performances ainsi que leur impact sur la trame WiMAX, IEEE 802.16e. Ces deux algorithmes ont été évalués pour différentes types de trames. L'évaluation a été faite selon le mode d'utilisation partielle des sous-canaux «PUSC», d'utilisation entière des sous-canaux «FUSC», le rapport DL:UL, la largeur de bande de la trame et la longueur de la trame en nombre de symboles OFDMA.

Bien que la simulation des deux algorithmes soit faite seulement pour une trame en mode temporel, TDD, l'extensibilité pour le mode fréquentiel, FDD est possible avec un minimum d'adaptation de ces algorithmes.

Le mode PUSC présente l'avantage de la granularité plus grande que celle du mode FUSC ce qui avantage le premier mode. Au niveau des deux algorithmes, le deuxième algorithme est plus performant que le premier pour les deux modes en question.

Lors de l'évaluation des deux algorithmes tout en changeant le rapport DL:UL, lorsque la portion de la trame UL augmente, le taux de remplissage de la trame s'améliore pour les différents types de la trame. Par contre, lorsque la portion de la trame DL augmente, le taux de remplissage de la trame diminue pour les différents types de la trame. En effet, puisque la sous trame de la voie montante ne requiert pas que les rafales soient insérées dans des régions rectangulaires, les seules pertes possibles sont ceux des ajouts de bits de bourrage «Padding bits». Ce qui diminuera les espaces qui resteront vides dans la trame.

La variation de la largeur de bande de la trame n'a pas un impact trop important sur le taux de remplissage de la trame. Par contre, elle a un effet sur le temps d'exécution ou de traitement de chaque algorithme. Aussi, le deuxième algorithme est plus avantageux que le premier algorithme.

La taille de la trame en nombre de symboles OFDMA a un effet sur les performances de chaque algorithme. En effet, lorsque la taille de la trame est grande, le taux de remplissage

s'améliore et le temps d'exécution ou de traitement de chaque algorithme augmente légèrement. Par contre, le deuxième algorithme est toujours plus performant que le premier.

RECOMMANDATIONS

Dans ce mémoire, le principal travail est de trouver des algorithmes de mise en trame des paquets d'un système de télécommunications WiMAX. De plus, une simulation des algorithmes proposés a été faite pour déterminer leurs performances.

La simulation a été faite selon les conditions théoriques d'un système WiMAX. Une autre étape qui est nécessaire pour valider les résultats obtenus est celle de la simulation sur un simulateur de réseaux comme par exemple le logiciel OPNET. Une étape subséquente consistera à l'implémentation et à la validation de ces deux algorithmes sur un système réel dans son environnement d'utilisation.

Pour utiliser efficacement toutes les tranches de la trame, une autre proposition peut être envisagée. Cette proposition consiste à concevoir l'ordonnanceur, *Scheduler*, de telle sorte qu'il intègre dans son allocation un algorithme de mise en trame des paquets. Cette solution peut, probablement, alourdir l'ordonnanceur au niveau de la puissance du calcul, mais les tranches de la trame seront utilisées d'une façon optimale.

Une solution médiane consistera à fournir une méthode qui combinera les deux solutions citées précédemment. Cette solution sera de concevoir un ordonnanceur et un algorithme de mise en trame séparément, mais chacun des deux communique avec l'autre pour remplir efficacement la trame.

ANNEXE I

Algorithme 1

```
%##########################################################################
#
%initialiser tout
clc;
clear all;
%##########################################################################
#
    % PUSC : Data=24
    % FUSC : Data=48
    mode = 3; %3=PUSC,  2=FUSC
    Nombre_profil = 5;
    FFT = 2048;
    Ratio_DL_UL = 3/1;
    T_frame = 0.01;

    if mode == 3
        DataCarrier = 24;
        if FFT == 2048
            Nsubchannel = 60;
        elseif FFT == 1024
            Nsubchannel = 30;
        elseif FFT == 512
            Nsubchannel = 15;
        end
    elseif mode == 2
        DataCarrier = 48;
        if FFT == 2048
            Nsubchannel = 32;
        elseif FFT == 1024
            Nsubchannel = 16;
        elseif FFT == 512
            Nsubchannel = 8;
        end
    end

    if T_frame == 0.005
        N_OFDMA_Symb = 48; %Trame de 5 ms
    elseif T_frame == 0.01
        N_OFDMA_Symb = 96; %Trame de 10 ms
    elseif T_frame == 0.02
        N_OFDMA_Symb = 192; %Trame de 20 ms
    end
    if Ratio_DL_UL == 3/2
        DL_OFDMA_Symb = floor(N_OFDMA_Symb*3/5) - 2;
    elseif Ratio_DL_UL == 1/1
        DL_OFDMA_Symb = floor(N_OFDMA_Symb*0.5) - 2;
    elseif Ratio_DL_UL == 2/3
        DL_OFDMA_Symb = floor(N_OFDMA_Symb*2/5) - 2;
    elseif Ratio_DL_UL == 2/1
        DL_OFDMA_Symb = floor(N_OFDMA_Symb*2/3) - 2;
    elseif Ratio_DL_UL == 1/2
        DL_OFDMA_Symb = floor(N_OFDMA_Symb*1/3) - 2;
    elseif Ratio_DL_UL == 3/1
        DL_OFDMA_Symb = floor(N_OFDMA_Symb*3/4) - 2;
    elseif Ratio_DL_UL == 1/3
```

```
        DL_OFDMA_Symb = floor(N_OFDMA_Symb*1/4) - 2;
    end
    UL_OFDMA_Symb = N_OFDMA_Symb - DL_OFDMA_Symb  - 4;
    N_ulmapsubchannel = floor(Nsubchannel/2);
%##########################################################################
#
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Tableau des CID
for k=1:1:1000
    CID(k)=k;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Tableau des Burst profil

for iii = 1:1:500
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Tableau des Burst profil
BurstProfil = randsrc(1000,1,[1 2 3 4 5 6 7 8]);
TaillePaquet = poissrnd(200,1,1000);

%##########################################################################
#
%##########################################################################
#
%Construction de la trame

    %initialisation de la trame
    DL_subFrame = zeros(Nsubchannel,DL_OFDMA_Symb);
    UL_subFrame = zeros(Nsubchannel,UL_OFDMA_Symb);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
tic;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% DL_subframe
    %Preamble
    for k = 1:1:60
        DL_subFrame(k,1) = 0.5*10;
    end

    %FCH
    for k = 1:1:2
        for m = 2:1:3
            DL_subFrame(k,m) = 1*10;
        end
    end

    %DL_MAP
```

```
    for k = 3:1:60
        for m = 2:1:3
            DL_subFrame(k,m) = 1*10;
        end
    end

    %UL_MAP
    N_ulmapsubchannel = 25;
    for k = 1:1:N_ulmapsubchannel
        for m = 4:1:5
            DL_subFrame(k,m) = 1*10;
        end
    end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% UL_subframe
    Total_UL = 0;
    slot_tot = UL_OFDMA_Symb*Nsubchannel;
    % Ranging
    for m = 1:1:UL_OFDMA_Symb
        for k = 1:1:2
            UL_subFrame(k,m) = 5;
            Total_UL = Total_UL + 2*DataCarrier;
        end
    end
    %Burst1
    for m = 1:1:UL_OFDMA_Symb
        for k = 3:1:floor(Nsubchannel/5)
            UL_subFrame(k,m) = 1*10;
            Total_UL = Total_UL + 2*DataCarrier;
        end
    end
    for m = 1:1:floor(UL_OFDMA_Symb/2)
        UL_subFrame(floor(Nsubchannel/5)+1,m) = 1*10;
        Total_UL = Total_UL + 2*DataCarrier;
    end

    %Burst2
    for m = floor(UL_OFDMA_Symb/2+0.5)+1:1:UL_OFDMA_Symb
        UL_subFrame(floor(Nsubchannel/5)+1,m) = 2*10;
        Total_UL = Total_UL + 2*DataCarrier;
    end
    for m = 1:1:UL_OFDMA_Symb
        for k = floor(Nsubchannel/5)+2:1:2*floor(Nsubchannel/5)
            UL_subFrame(k,m) = 2*10;
            Total_UL = Total_UL + 2*DataCarrier;
        end
    end

    %Burst3
    for m = 1:1:UL_OFDMA_Symb
```

```
        for k = 2*floor(Nsubchannel/5)+1:1:3*floor(Nsubchannel/5)
            UL_subFrame(k,m) = 3*10;
            Total_UL = Total_UL + 4*DataCarrier;
        end
    end
    for m = 1:1:floor(UL_OFDMA_Symb/2)
        UL_subFrame(3*floor(Nsubchannel/5)+1,m) = 3*10;
        Total_UL = Total_UL + 4*DataCarrier;
    end

    %Burst4
    for m = floor(UL_OFDMA_Symb/2+0.5)+1:1:UL_OFDMA_Symb
        UL_subFrame(3*floor(Nsubchannel/5)+1,m) = 4*10;
        Total_UL = Total_UL + 4*DataCarrier;
    end
    for m = 1:1:UL_OFDMA_Symb
        for k = 3*floor(Nsubchannel/5)+2:1:4*floor(Nsubchannel/5)
            UL_subFrame(k,m) = 4*10;
            Total_UL = Total_UL + 4*DataCarrier;
        end
    end


    %Burst5
   for m = 1:1:UL_OFDMA_Symb
        for k = 4*floor(Nsubchannel/5):1:Nsubchannel
            UL_subFrame(k,m) = 5*10;
            Total_UL = Total_UL + 6*DataCarrier;
        end
    end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% DL_subframe (LD-MAP & DL burst #1 à #5)

  % Time slot disponibles et les 2 regions
    Nofdmasymbol = DL_OFDMA_Symb;

  %DL_MAP
    for k = 3:1:60
        for m = 2:1:3
            DL_subFrame(k,m) = 1*10;
        end
    end


  %DL burst #1 à #5
    %reception des paquets et traitement(mapping)
    Burst_1 = 0;Burst_2 = 0;Burst_3 = 0;Burst_4 = 0;Burst_5 = 0;
    k=0;
```

```
    while Burst_1 + Burst_2 + Burst_3 + Burst_4 + Burst_5 <
Nsubchannel*DataCarrier*Nofdmasymbol - (2*N_ulmapsubchannel +
3*Nsubchannel + 2*N_ulmapsubchannel + (Nsubchannel-N_ulmapsubchannel) +
34)*DataCarrier
    k = k + 1;
        if BurstProfil(k)==1
            Burst_1= Burst_1 + TaillePaquet(k)*8;

        elseif BurstProfil(k)==2
            Burst_2= Burst_2 + 0.5*TaillePaquet(k)*8*4/3;

        elseif BurstProfil(k)==3
            Burst_3= Burst_3 + 0.25*TaillePaquet(k)*8*2;

         elseif BurstProfil(k)==4
            Burst_4= Burst_4 + 0.25*TaillePaquet(k)*8*4/3;

        elseif BurstProfil(k)==5
            Burst_5= Burst_5 + 1/6*TaillePaquet(k)*8*3/2;

        end

    end

    lastPacket = k;
    Subch_Reg1=Nsubchannel- N_ulmapsubchannel;
    DL_s=DL_OFDMA_Symb;
    Reg1 = DataCarrier* N_ulmapsubchannel;
%#########################################################################
%############################ SOLUTION A: ################################
%#########################################################################
    CombinaisonA1 = -(Burst_1 - Subch_Reg1*DataCarrier*DL_s);
    CombinaisonA12 = -((Burst_1 + Burst_2) - Subch_Reg1*DataCarrier*DL_s);
    CombinaisonA13 = -((Burst_1 + Burst_3) - Subch_Reg1*DataCarrier*DL_s);
    CombinaisonA14 = -((Burst_1 + Burst_4) - Subch_Reg1*DataCarrier*DL_s);
    CombinaisonA15 = -((Burst_1 + Burst_5) - Subch_Reg1*DataCarrier*DL_s);

    CombinaisonA2 = -(Burst_2 - Subch_Reg1*DataCarrier*DL_s);
    CombinaisonA23 = -((Burst_2 + Burst_3) - Subch_Reg1*DataCarrier*DL_s);
    CombinaisonA24 = -((Burst_2 + Burst_4) - Subch_Reg1*DataCarrier*DL_s);
    CombinaisonA25 = -((Burst_2 + Burst_5) - Subch_Reg1*DataCarrier*DL_s);

    CombinaisonA3 = -(Burst_3 - Subch_Reg1*DataCarrier*DL_s);
    CombinaisonA34 = -((Burst_3 + Burst_4) - Subch_Reg1*DataCarrier*DL_s);
    CombinaisonA35 = -((Burst_3 + Burst_5) - Subch_Reg1*DataCarrier*DL_s);

    CombinaisonA4 = -(Burst_4 - Subch_Reg1*DataCarrier*DL_s);
    CombinaisonA45 = -((Burst_4 + Burst_5) - Subch_Reg1*DataCarrier*DL_s);

    CombinaisonA5 = -(Burst_5 - Subch_Reg1*DataCarrier*DL_s);

    CombinaisonA = [CombinaisonA1 CombinaisonA12 CombinaisonA13
CombinaisonA14 CombinaisonA15 CombinaisonA2 CombinaisonA23 CombinaisonA24
```

```
CombinaisonA25 CombinaisonA3 CombinaisonA34 CombinaisonA35 CombinaisonA4
CombinaisonA45 CombinaisonA5];

    Comb = (CombinaisonA1 < Reg1 & CombinaisonA1 > 0)|(CombinaisonA12 <
2*Reg1  & CombinaisonA12 > Reg1)|(CombinaisonA13 < 2*Reg1  &
CombinaisonA13 > Reg1) |(CombinaisonA14 < 2*Reg1  & CombinaisonA14 >
Reg1)| (CombinaisonA15 < 2*Reg1  & CombinaisonA15 > Reg1)| (CombinaisonA2
< Reg1  & CombinaisonA2 > 0) | (CombinaisonA23 < 2*Reg1  & CombinaisonA23
> Reg1 ) |(CombinaisonA24 < 2*Reg1  & CombinaisonA24 > Reg1)
|(CombinaisonA25 < 2*Reg1  & CombinaisonA25 > Reg1) |(CombinaisonA3 < Reg1
& CombinaisonA3 > 0) |(CombinaisonA34 < 2*Reg1  & CombinaisonA34 > Reg1)
|(CombinaisonA35 < 2*Reg1  & CombinaisonA35 > Reg1)| (CombinaisonA4 < Reg1
& CombinaisonA4 > 0) |(CombinaisonA45 < 2*Reg1  & CombinaisonA45 > Reg1)
|(CombinaisonA5 < Reg1  & CombinaisonA5 > 0);

    if ((Comb) == 1)
        sprintf('%s','SOLUTION A !!!!!')

        for j = 1:1:15
            a(j) = CombinaisonA(j);
            if (CombinaisonA(j) < 0)
                a(j) = 1000000; % maximiser les négatifs pour qu'ils
n'influencent pas sur la recherche du min
            end
        end

        [A,I] =
min([a(1),a(2),a(3),a(4),a(5),a(6),a(7),a(8),a(9),a(10),a(11),a(12),a(13),
a(14),a(15)]);

        if (CombinaisonA1 < Reg1 & CombinaisonA1 > 0) % CombinaisonA 1
            sprintf('%s','Combinaison 1 .....')

            a(1) = 2000000;
            N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole1_reste = mod(Burst_1,(N_ulmapsubchannel));
            for k = 1:1:N_ulmapsubchannel
                for m = 6:1:6 + N_ofdmaSymbole1
                    DL_subFrame(k,m) = 10*1;
                end
            end
            for k = 1:1:floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,6+N_ofdmaSymbole1+1) = 10*1;
            end

            Burst = [0,Burst_2,Burst_3,Burst_4,Burst_5];
            [brst,i] = max(Burst);

            N_ofdmaSymbole1 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
            N_ofdmaSymbole1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
```

```
                for m = 4:1:3 + N_ofdmaSymbole1
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,4+N_ofdmaSymbole2) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_ofdmaSymbole_1 = N_ofdmaSymbole1 + 1 +
floor(Burst(i)/(DataCarrier*(Nsubchannel-N_ulmapsubchannel)));
            N_ofdmaSymbole_1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4+N_ofdmaSymbole1+1:1:3 + N_ofdmaSymbole_1
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k = N_ulmapsubchannel+1:1:floor(N_ofdmaSymbole_1_reste)
                DL_subFrame(k,3+N_ofdmaSymbole_1+1) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole_1-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole_1-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole_1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole_1+1:1:4+N_ofdmaSymbole_1+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole_1-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole_1-1-3));
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
                for m = 4+N_ofdmaSymbole_1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
```

```
                end
            end
            for m =
4+N_ofdmaSymbole_1+1:1:4+N_ofdmaSymbole_1+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
            end

        elseif (CombinaisonA2 < Reg1  & CombinaisonA2 > 0) % CombinaisonA
2
            sprintf('%s','Combinaison 2 .....')

            a(6) = 2000000;
            N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole2_reste = mod(Burst_2,(N_ulmapsubchannel));
            for k = 1:1:N_ulmapsubchannel
                for m = 6:1:6 + N_ofdmaSymbole2
                    DL_subFrame(k,m) = 10*2;
                end
            end
            for k = 1:1:floor(N_ofdmaSymbole2_reste)
                DL_subFrame(k,6+N_ofdmaSymbole2+1) = 10*2;
            end

            Burst = [Burst_1,0,Burst_3,Burst_4,Burst_5];
            [brst,i] = max(Burst);

            N_ofdmaSymbole1 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
            N_ofdmaSymbole1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4:1:3 + N_ofdmaSymbole1
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,4+N_ofdmaSymbole2) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_ofdmaSymbole_1 = N_ofdmaSymbole1 + 1 +
floor(Burst(i)/(DataCarrier*(Nsubchannel-N_ulmapsubchannel)));
            N_ofdmaSymbole_1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4+N_ofdmaSymbole1+1:1:3 + N_ofdmaSymbole_1
                    DL_subFrame(k,m) = 10*i;
                end
            end
```

```
            for k = N_ulmapsubchannel+1:1:floor(N_ofdmaSymbole_1_reste)
                DL_subFrame(k,3+N_ofdmaSymbole_1+1) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole_1-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole_1-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole_1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole_1+1:1:4+N_ofdmaSymbole_1+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole_1-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole_1-1-3));
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
                for m = 4+N_ofdmaSymbole_1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole_1+1:1:4+N_ofdmaSymbole_1+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
            end



        elseif (CombinaisonA3 < Reg1  & CombinaisonA3 > 0) % CombinaisonA
3
            sprintf('%s','Combinaison 3 .....')

            a(10) = 2000000;
            N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole3_reste = mod(Burst_3,(N_ulmapsubchannel));
            for k = 1:1:N_ulmapsubchannel
                for m = 6:1:6 + N_ofdmaSymbole3
                    DL_subFrame(k,m) = 10*3;
```

```
                end
            end
            for k = 1:1:floor(N_ofdmaSymbole2_reste)
                DL_subFrame(k,6+N_ofdmaSymbole3+1) = 10*3;
            end

            Burst = [Burst_1,Burst_2,0,Burst_4,Burst_5];
            [brst,i] = max(Burst);

            N_ofdmaSymbole1 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
            N_ofdmaSymbole1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4:1:3 + N_ofdmaSymbole1
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,4+N_ofdmaSymbole2) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_ofdmaSymbole_1 = N_ofdmaSymbole1 + 1 +
floor(Burst(i)/(DataCarrier*(Nsubchannel-N_ulmapsubchannel)));
            N_ofdmaSymbole_1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4+N_ofdmaSymbole1+1:1:3 + N_ofdmaSymbole_1
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k = N_ulmapsubchannel+1:1:floor(N_ofdmaSymbole_1_reste)
                DL_subFrame(k,3+N_ofdmaSymbole_1+1) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole_1-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole_1-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole_1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole_1+1:1:4+N_ofdmaSymbole_1+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
```

```
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole_1-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole_1-1-3));
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
                for m = 4+N_ofdmaSymbole_1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole_1+1:1:4+N_ofdmaSymbole_1+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
            end

        elseif (CombinaisonA4 < Reg1  & CombinaisonA4 > 0) % CombinaisonA
4
            sprintf('%s','Combinaison 4 !!!!!')

            a(13) = 2000000;
            N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole4_reste = mod(Burst_4,(N_ulmapsubchannel));
            for k = 1:1:N_ulmapsubchannel
                for m = 6:1:6 + N_ofdmaSymbole4
                    DL_subFrame(k,m) = 10*4;
                end
            end
            for k = 1:1:floor(N_ofdmaSymbole2_reste)
                DL_subFrame(k,6+N_ofdmaSymbole2+1) = 10*4;
            end

            Burst = [Burst_1,Burst_2,Burst_3,0,Burst_5];
            [brst,i] = max(Burst);

            N_ofdmaSymbole1 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
            N_ofdmaSymbole1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4:1:3 + N_ofdmaSymbole1
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,4+N_ofdmaSymbole2) = 10*i;
```

```
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_ofdmaSymbole_1 = N_ofdmaSymbole1 + 1 +
floor(Burst(i)/(DataCarrier*(Nsubchannel-N_ulmapsubchannel)));
            N_ofdmaSymbole_1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4+N_ofdmaSymbole1+1:1:3 + N_ofdmaSymbole_1
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k = N_ulmapsubchannel+1:1:floor(N_ofdmaSymbole_1_reste)
                DL_subFrame(k,3+N_ofdmaSymbole_1+1) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole_1-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole_1-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole_1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole_1+1:1:4+N_ofdmaSymbole_1+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole_1-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole_1-1-3));
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
                for m = 4+N_ofdmaSymbole_1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole_1+1:1:4+N_ofdmaSymbole_1+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
            end
```

```
        elseif (CombinaisonA5 < Reg1  & CombinaisonA5 > 0) % CombinaisonA
5
            sprintf('%s','Combinaison 5 .....')

            a(15) = 2000000;
            N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole5_reste = mod(Burst_5,(N_ulmapsubchannel));
            for k = 1:1:N_ulmapsubchannel
                for m = 6:1:6 + N_ofdmaSymbole5
                    DL_subFrame(k,m) = 10*5;
                end
            end
            for k = 1:1:floor(N_ofdmaSymbole2_reste)
                DL_subFrame(k,6+N_ofdmaSymbole5+1) = 10*5;
            end

            Burst = [Burst_1,Burst_2,Burst_3,Burst_4,0];
            [brst,i] = max(Burst);

            N_ofdmaSymbole1 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
            N_ofdmaSymbole1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4:1:3 + N_ofdmaSymbole1
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,4+N_ofdmaSymbole2) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_ofdmaSymbole_1 = N_ofdmaSymbole1 + 1 +
floor(Burst(i)/(DataCarrier*(Nsubchannel-N_ulmapsubchannel)));
            N_ofdmaSymbole_1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4+N_ofdmaSymbole1+1:1:3 + N_ofdmaSymbole_1
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k = N_ulmapsubchannel+1:1:floor(N_ofdmaSymbole_1_reste)
                DL_subFrame(k,3+N_ofdmaSymbole_1+1) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);
```

```
            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole_1-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole_1-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole_1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole_1+1:1:4+N_ofdmaSymbole_1+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole_1-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole_1-1-3));
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
                for m = 4+N_ofdmaSymbole_1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole_1+1:1:4+N_ofdmaSymbole_1+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
            end

        elseif (CombinaisonA12 < 2*Reg1  & CombinaisonA12 > Reg1) %
CombinaisonA 12
            sprintf('%s','Combinaison 12 .....')

            a(2) = 2000000;
            N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole1_reste = mod(Burst_1,(N_ulmapsubchannel));
            N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole2_reste = mod(Burst_2,(N_ulmapsubchannel));
            for k = 1:1:N_ulmapsubchannel
                for m = 6:1:6 + N_ofdmaSymbole1
                    DL_subFrame(k,m) = 10*1;
                end
                for m = 6 + N_ofdmaSymbole1+2:1:6 + N_ofdmaSymbole2 +
N_ofdmaSymbole1+1
                    DL_subFrame(k,m) = 10*2;
                end
            end
```

```
            for k = 1:1:floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole1 + 1) = 10*1;
            end
            for k = 1:1:floor(N_ofdmaSymbole2_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole2 + N_ofdmaSymbole1+2) =
10*2;
            end

            Burst = [0,0,Burst_3,Burst_4,Burst_5];
            [brst,i] = max(Burst);

            N_ofdmaSymbole2 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
            N_ofdmaSymbole2_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4:1:3 + N_ofdmaSymbole2
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole2_reste)
                DL_subFrame(k,4+N_ofdmaSymbole2) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole2-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole2-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole2+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole2+1:1:4+N_ofdmaSymbole2+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole2-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole2-1-3));
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
                for m = 4+N_ofdmaSymbole2+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
```

```
            end
        end
        for m =
4+N_ofdmaSymbole2+1:1:4+N_ofdmaSymbole2+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
        end

    elseif (CombinaisonA13 < 2*Reg1  & CombinaisonA13 > Reg1) %
CombinaisonA 13
        sprintf('%s','Combinaison 13 .....')

        a(3) = 2000000;
        N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*N_ulmapsubchannel));
        N_ofdmaSymbole1_reste = mod(Burst_1,(N_ulmapsubchannel));
        N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*N_ulmapsubchannel));
        N_ofdmaSymbole3_reste = mod(Burst_3,(N_ulmapsubchannel));
        for k = 1:1:N_ulmapsubchannel
            for m = 6:1:6 + N_ofdmaSymbole1
                DL_subFrame(k,m) = 10*1;
            end
            for m = 6 + N_ofdmaSymbole1+2:1:6 + N_ofdmaSymbole3 +
N_ofdmaSymbole1+1
                DL_subFrame(k,m) = 10*3;
            end
        end
        for k = 1:1:floor(N_ofdmaSymbole1_reste)
            DL_subFrame(k,6 + N_ofdmaSymbole1 + 1) = 10*1;
        end
        for k = 1:1:floor(N_ofdmaSymbole3_reste)
            DL_subFrame(k,6 + N_ofdmaSymbole3 + N_ofdmaSymbole1+2) =
10*3;
        end

        Burst = [0,Burst_2,0,Burst_4,Burst_5];
        [brst,i] = max(Burst);

        N_ofdmaSymbole2 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
        N_ofdmaSymbole2_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
        for k = N_ulmapsubchannel+1:1:Nsubchannel
            for m = 4:1:3 + N_ofdmaSymbole2
                DL_subFrame(k,m) = 10*i;
            end
        end
        for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole2_reste)
            DL_subFrame(k,4+N_ofdmaSymbole2) = 10*i;
        end

        Burst(i) = 0;
```

```
            [brst,i] = max(Burst);

            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole2-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole2-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole2+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole2+1:1:4+N_ofdmaSymbole2+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole2-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole2-1-3));
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
                for m = 4+N_ofdmaSymbole2+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole2+1:1:4+N_ofdmaSymbole2+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
            end

        elseif (CombinaisonA14 < 2*Reg1  & CombinaisonA14 > Reg1) %
CombinaisonA 14
            sprintf('%s','Combinaison 14 .....')

            a(4) = 2000000;
            N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole1_reste = mod(Burst_1,(N_ulmapsubchannel));
            N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole4_reste = mod(Burst_4,(N_ulmapsubchannel));
            for k = 1:1:N_ulmapsubchannel
                for m = 6:1:6 + N_ofdmaSymbole1
                    DL_subFrame(k,m) = 10*1;
                end
                for m = 6 + N_ofdmaSymbole1+2:1:6 + N_ofdmaSymbole4 +
N_ofdmaSymbole1+1
                    DL_subFrame(k,m) = 10*4;
```

```
                end
            end
            for k = 1:1:floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole1 + 1) = 10*1;
            end
            for k = 1:1:floor(N_ofdmaSymbole4_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole4 + N_ofdmaSymbole1+2) =
10*4;
            end

            Burst = [0,Burst_2,Burst_3,0,Burst_5];
            [brst,i] = max(Burst);

            N_ofdmaSymbole2 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
            N_ofdmaSymbole2_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4:1:3 + N_ofdmaSymbole2
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole2_reste)
                DL_subFrame(k,4+N_ofdmaSymbole2) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole2-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole2-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole2+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole2+1:1:4+N_ofdmaSymbole2+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole2-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole2-1-3));
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
```

```
                for m = 4+N_ofdmaSymbole2+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole2+1:1:4+N_ofdmaSymbole2+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
            end

        elseif (CombinaisonA15 < 2*Reg1  & CombinaisonA15 > Reg1) %
CombinaisonA 15
            sprintf('%s','Combinaison 15 !.....')

            a(5) = 2000000;
            N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole1_reste = mod(Burst_1,(N_ulmapsubchannel));
            N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole5_reste = mod(Burst_5,(N_ulmapsubchannel));
            for k = 1:1:N_ulmapsubchannel
                for m = 6:1:6 + N_ofdmaSymbole1
                    DL_subFrame(k,m) = 10*1;
                end
                for m = 6 + N_ofdmaSymbole1+2:1:6 + N_ofdmaSymbole5 +
N_ofdmaSymbole1+1
                    DL_subFrame(k,m) = 10*5;
                end
            end
            for k = 1:1:floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole1 + 1) = 10*1;
            end
            for k = 1:1:floor(N_ofdmaSymbole5_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole5 + N_ofdmaSymbole1+2) =
10*5;
            end

            Burst = [0,Burst_2,Burst_3,Burst_4,0];
            [brst,i] = max(Burst);

            N_ofdmaSymbole2 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
            N_ofdmaSymbole2_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4:1:3 + N_ofdmaSymbole2
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole2_reste)
                DL_subFrame(k,4+N_ofdmaSymbole2) = 10*i;
            end
```

```
            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole2-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole2-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole2+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole2+1:1:4+N_ofdmaSymbole2+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole2-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole2-1-3));
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
                for m = 4+N_ofdmaSymbole2+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole2+1:1:4+N_ofdmaSymbole2+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
            end

        elseif (CombinaisonA23 < 2*Reg1  & CombinaisonA23 > Reg1 ) %
CombinaisonA 23
            sprintf('%s','Combinaison 23 .....')

            a(7) = 2000000;
            N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole2_reste = mod(Burst_2,(N_ulmapsubchannel));
            N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole3_reste = mod(Burst_3,(N_ulmapsubchannel));
            for k = 1:1:N_ulmapsubchannel
                for m = 6:1:5 + N_ofdmaSymbole2
                    DL_subFrame(k,m) = 10*2;
                end
```

```
                for m = 5 + N_ofdmaSymbole2+2:1:5 + N_ofdmaSymbole3 +
N_ofdmaSymbole2+1
                    DL_subFrame(k,m) = 10*3;
                end
            end
            for k = 1:1:floor(N_ofdmaSymbole2_reste)
                DL_subFrame(k,5 + N_ofdmaSymbole2 + 1) = 10*2;
            end
            for k = 1:1:floor(N_ofdmaSymbole3_reste)
                DL_subFrame(k,5 + N_ofdmaSymbole3 + N_ofdmaSymbole2+2) =
10*3;
            end

            Burst = [Burst_1,0,0,Burst_4,Burst_5];
            [brst,i] = max(Burst);

            N_ofdmaSymbole1 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
            N_ofdmaSymbole1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4:1:3 + N_ofdmaSymbole1
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,4+N_ofdmaSymbole1) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole1-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole1-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole1+1:1:4+N_ofdmaSymbole1+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole1-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole1-1-3));
```

```
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
                for m = 4+N_ofdmaSymbole1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole1+1:1:4+N_ofdmaSymbole1+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
            end



        elseif (CombinaisonA24 < 2*Reg1  & CombinaisonA24 > Reg1) %
CombinaisonA 24
            sprintf('%s','Combinaison 24 .....')

            a(8) = 2000000;
            N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole2_reste = mod(Burst_2,(N_ulmapsubchannel));
            N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole4_reste = mod(Burst_4,(N_ulmapsubchannel));
            for k = 1:1:N_ulmapsubchannel
                for m = 6:1:6 + N_ofdmaSymbole2
                    DL_subFrame(k,m) = 10*2;
                end
                for m = 6 + N_ofdmaSymbole2+2:1:6 + N_ofdmaSymbole4 +
N_ofdmaSymbole2+1
                    DL_subFrame(k,m) = 10*4;
                end
            end
            for k = 1:1:floor(N_ofdmaSymbole2_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole2 + 1) = 10*2;
            end
            for k = 1:1:floor(N_ofdmaSymbole4_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole2 + N_ofdmaSymbole4+2) =
10*4;
            end

            Burst = [Burst_1,0,Burst_3,0,Burst_5];
            [brst,i] = max(Burst);

            N_ofdmaSymbole1 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
            N_ofdmaSymbole1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4:1:3 + N_ofdmaSymbole1
                    DL_subFrame(k,m) = 10*i;
                end
```

```
            end
            for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,4+N_ofdmaSymbole1) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole1-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole1-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole1+1:1:4+N_ofdmaSymbole1+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole1-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole1-1-3));
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
                for m = 4+N_ofdmaSymbole1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole1+1:1:4+N_ofdmaSymbole1+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
            end

        elseif (CombinaisonA25 < 2*Reg1  & CombinaisonA25 > Reg1) %
CombinaisonA 25
            sprintf('%s','Combinaison 25 .....')

            a(9) = 2000000;
            N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole2_reste = mod(Burst_2,(N_ulmapsubchannel));
            N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole5_reste = mod(Burst_5,(N_ulmapsubchannel));
```

```
            for k = 1:1:N_ulmapsubchannel
                for m = 6:1:6 + N_ofdmaSymbole2
                    DL_subFrame(k,m) = 10*2;
                end
                for m = 6 + N_ofdmaSymbole2+2:1:6 + N_ofdmaSymbole5 +
N_ofdmaSymbole2+1
                    DL_subFrame(k,m) = 10*5;
                end
            end
            for k = 1:1:floor(N_ofdmaSymbole2_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole2 + 1) = 10*2;
            end
            for k = 1:1:floor(N_ofdmaSymbole5_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole2 + N_ofdmaSymbole5+2) =
10*5;
            end

            Burst = [Burst_1,0,Burst_3,Burst_4,0];
            [brst,i] = max(Burst);

            N_ofdmaSymbole1 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
            N_ofdmaSymbole1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4:1:3 + N_ofdmaSymbole1
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,4+N_ofdmaSymbole1) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole1-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole1-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole1+1:1:4+N_ofdmaSymbole1+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);
```

```
            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole1-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole1-1-3));
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
                for m = 4+N_ofdmaSymbole1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole1+1:1:4+N_ofdmaSymbole1+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
            end

        elseif (CombinaisonA34 < 2*Reg1  & CombinaisonA34 > Reg1) %
CombinaisonA 34
            sprintf('%s','Combinaison 34 .....')

            a(11) = 2000000;
            N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole3_reste = mod(Burst_3,(N_ulmapsubchannel));
            N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole4_reste = mod(Burst_4,(N_ulmapsubchannel));
            for k = 1:1:N_ulmapsubchannel
                for m = 6:1:6 + N_ofdmaSymbole3
                    DL_subFrame(k,m) = 10*3;
                end
                for m = 6 + N_ofdmaSymbole3+2:1:6 + N_ofdmaSymbole3 +
N_ofdmaSymbole4+1
                    DL_subFrame(k,m) = 10*4;
                end
            end
            for k = 1:1:floor(N_ofdmaSymbole3_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole3 + 1) = 10*3;
            end
            for k = 1:1:floor(N_ofdmaSymbole4_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole3 + N_ofdmaSymbole4+2) =
10*4;
            end

            Burst = [Burst_1,Burst_2,0,0,Burst_5];
            [brst,i] = max(Burst);

            N_ofdmaSymbole1 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
            N_ofdmaSymbole1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4:1:3 + N_ofdmaSymbole1
```

```
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,4+N_ofdmaSymbole1) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole1-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole1-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole1+1:1:4+N_ofdmaSymbole1+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole1-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole1-1-3));
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
                for m = 4+N_ofdmaSymbole1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole1+1:1:4+N_ofdmaSymbole1+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
            end

        elseif (CombinaisonA35 < 2*Reg1  & CombinaisonA35 > Reg1) %
CombinaisonA 35
            sprintf('%s','Combinaison 35 .....')

            a(12) = 2000000;
            N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole3_reste = mod(Burst_3,(N_ulmapsubchannel));
```

```
            N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole5_reste = mod(Burst_5,(N_ulmapsubchannel));
            for k = 1:1:N_ulmapsubchannel
                for m = 6:1:6 + N_ofdmaSymbole3
                    DL_subFrame(k,m) = 10*3;
                end
                for m = 6 + N_ofdmaSymbole3+2:1:6 + N_ofdmaSymbole3 +
N_ofdmaSymbole5+1
                    DL_subFrame(k,m) = 10*5;
                end
            end
            for k = 1:1:floor(N_ofdmaSymbole3_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole3 + 1) = 10*3;
            end
            for k = 1:1:floor(N_ofdmaSymbole5_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole3 + N_ofdmaSymbole5+2) =
10*5;
            end

            Burst = [Burst_1,Burst_2,0,Burst_4,0];
            [brst,i] = max(Burst);

            N_ofdmaSymbole1 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
            N_ofdmaSymbole1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4:1:3 + N_ofdmaSymbole1
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,4+N_ofdmaSymbole1) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole1-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole1-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole1+1:1:4+N_ofdmaSymbole1+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
            end
```

```
            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole1-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole1-1-3));
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
                for m = 4+N_ofdmaSymbole1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole1+1:1:4+N_ofdmaSymbole1+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
            end

        elseif (CombinaisonA45 < 2*Reg1  & CombinaisonA45 > Reg1) %
CombinaisonA 45
            sprintf('%s','Combinaison 45 .....')

            a(14) = 2000000;
            N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole4_reste = mod(Burst_4,(N_ulmapsubchannel));
            N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*N_ulmapsubchannel));
            N_ofdmaSymbole5_reste = mod(Burst_5,(N_ulmapsubchannel));
            for k = 1:1:N_ulmapsubchannel
                for m = 6:1:6 + N_ofdmaSymbole4
                    DL_subFrame(k,m) = 10*4;
                end
                for m = 6 + N_ofdmaSymbole4+2:1:6 + N_ofdmaSymbole4 +
N_ofdmaSymbole5+1
                    DL_subFrame(k,m) = 10*5;
                end
            end
            for k = 1:1:floor(N_ofdmaSymbole4_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole4 + 1) = 10*4;
            end
            for k = 1:1:floor(N_ofdmaSymbole5_reste)
                DL_subFrame(k,6 + N_ofdmaSymbole4 + N_ofdmaSymbole5+2) =
10*5;
            end

            Burst = [Burst_1,Burst_2,Burst_3,0,0];
            [brst,i] = max(Burst);

            N_ofdmaSymbole1 = floor(Burst(i)/(DataCarrier*(Nsubchannel-
N_ulmapsubchannel)));
```

```
            N_ofdmaSymbole1_reste = mod(Burst(i),(Nsubchannel-
N_ulmapsubchannel));
            for k = N_ulmapsubchannel+1:1:Nsubchannel
                for m = 4:1:3 + N_ofdmaSymbole1
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                DL_subFrame(k,4+N_ofdmaSymbole1) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole1-1-3)));
            N_subchannel1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole1-1-3));
            for k = N_ulmapsubchannel+1:1:N_ulmapsubchannel+N_subchannel1
                for m = 4+N_ofdmaSymbole1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole1+1:1:4+N_ofdmaSymbole1+N_subchannel1_reste
                DL_subFrame(N_ulmapsubchannel+N_subchannel1+1,m) = 10*i;
            end

            Burst(i) = 0;
            [brst,i] = max(Burst);

            N_subchannel_1 = floor(Burst(i)/(DataCarrier*(Nofdmasymbol-
N_ofdmaSymbole1-1-3)));
            N_subchannel_1_reste = mod(Burst(i),(Nofdmasymbol-
N_ofdmaSymbole1-1-3));
            for k =
N_ulmapsubchannel+N_subchannel1+2:1:N_ulmapsubchannel+N_subchannel1+N_subc
hannel_1+1
                for m = 4+N_ofdmaSymbole1+1:1:Nofdmasymbol
                    DL_subFrame(k,m) = 10*i;
                end
            end
            for m =
4+N_ofdmaSymbole1+1:1:4+N_ofdmaSymbole1+N_subchannel_1_reste

DL_subFrame(N_ulmapsubchannel+N_subchannel1+N_subchannel_1+2,m) = 10*i;
            end
        end

%###################  Fin de l'allocation des DL Burst_i  ##############

 SecondRemplissage = 0;
```

```
 while (1)
        lastPacket = lastPacket + 1;;
        Profil = BurstProfil(lastPacket);
        Taille = TaillePaquet(lastPacket);

        if Profil == 1
            if Taille*8/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
Taille*8/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
Taille*8/DataCarrier;
            else
                break;
            end
         elseif Profil == 2
             if 0.5*Taille*8*(4/3)/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
0.5*Taille*8*(4/3)/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
0.5*Taille*8*(4/3)/DataCarrier;
             else
                break;
            end
         elseif Profil == 3
             if 0.25*Taille*16/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
0.25*Taille*16/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
0.25*Taille*16/DataCarrier;
             else
                break;
             end
         elseif Profil == 4
             if 0.25*Taille*8*(4/3)/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
0.25*Taille*8*(4/3)/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
0.25*Taille*8*(4/3)/DataCarrier;
             else
                break;
            end
         elseif Profil == 5
             if 1/6*Taille*8*(3/2)/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
1/6*Taille*8*(3/2)/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
1/6*Taille*8*(3/2)/DataCarrier;
             else
                break;
            end
         end
    end


%#########################################################################
```

```
%########################## SOLUTION B: ################################
%#############################################################################
    else
        CombinaisonB1 = -(Burst_1 - 35*DataCarrier*34);
        CombinaisonB12 = -((Burst_1 + Burst_2) - 35*DataCarrier*34);
        CombinaisonB13 = -((Burst_1 + Burst_3) - 35*DataCarrier*34);
        CombinaisonB14 = -((Burst_1 + Burst_4) - 35*DataCarrier*34);
        CombinaisonB15 = -((Burst_1 + Burst_5) - 35*DataCarrier*34);

        CombinaisonB2 = -(Burst_2 - 35*DataCarrier*34);
        CombinaisonB23 = -((Burst_2 + Burst_3) - 35*DataCarrier*34);
        CombinaisonB24 = -((Burst_2 + Burst_4) - 35*DataCarrier*34);
        CombinaisonB25 = -((Burst_2 + Burst_5) - 35*DataCarrier*34);

        CombinaisonB3 = -(Burst_3 - 35*DataCarrier*34);
        CombinaisonB34 = -((Burst_3 + Burst_4) - 35*DataCarrier*34);
        CombinaisonB35 = -((Burst_3 + Burst_5) - 35*DataCarrier*34);

        CombinaisonB4 = -(Burst_4 - 35*DataCarrier*34);
        CombinaisonB45 = -((Burst_4 + Burst_5) - 35*DataCarrier*34);

        CombinaisonB5 = -(Burst_5 - 35*DataCarrier*34);

        CombinaisonB = [CombinaisonB1 CombinaisonB12 CombinaisonB13
CombinaisonB14 CombinaisonB15 CombinaisonB2 CombinaisonB23 CombinaisonB24
CombinaisonB25 CombinaisonB3 CombinaisonB34 CombinaisonB35 CombinaisonB4
CombinaisonB45 CombinaisonB5];

        if (CombinaisonB1 < 840 & CombinaisonB1 > 0)| (CombinaisonB12 <
2*840  & CombinaisonB12 > 840) |(CombinaisonB13 < 2*840  & CombinaisonB13
> 840) |(CombinaisonB14 < 2*840  & CombinaisonB14 > 840)| (CombinaisonB15
< 2*840  & CombinaisonB15 > 840)| (CombinaisonB2 < 840  & CombinaisonB2 >
0) | (CombinaisonB23 < 2*840 & CombinaisonB23 > 840 ) |(CombinaisonB24 <
2*840  & CombinaisonB24 > 840) |(CombinaisonB25 < 2*840  & CombinaisonB25
> 840) |(CombinaisonB3 < 840  & CombinaisonB3 > 0) |(CombinaisonB34 <
2*840  & CombinaisonB34 > 840) (CombinaisonB35 < 2*840  & CombinaisonB35 >
840)| (CombinaisonB4 < 840  & CombinaisonB4 > 0) |(CombinaisonB45 < 2*840
& CombinaisonB45 > 840) |(CombinaisonB5 < 840  & CombinaisonB5 > 0)
            % 840 = 35*24

            sprintf('%s','SOLUTION B !!!!!')

            for j = 1:1:15
                b(j) = CombinaisonB(j);
                if (CombinaisonB(j) < 0)
                    b(j) = 1000000; % maximiser les négatifs pour qu'ils
n'influencent pas sur la recherche du min
                end
            end

            [B,I] =
min([b(1),b(2),b(3),b(4),b(5),b(6),b(7),b(8),b(9),b(10),b(11),b(12),b(13),
b(14),b(15)]);
```

```
                if (I == 1) % CombinaisonB 1
                    sprintf('%s','Combinaison 1 .....')

                    b(1) = 2000000;
                    N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole1_reste = mod(Burst_1,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole1
                            DL_subFrame(k,m) = 10*1;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                        DL_subFrame(k,4+N_ofdmaSymbole1+1) = 10*1;
                    end

                elseif (I == 6) % CombinaisonB 2
                    sprintf('%s','Combinaison 2 .....')

                    b(6) = 2000000
                    N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole2_reste = mod(Burst_2,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole2
                            DL_subFrame(k,m) = 10*2;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole2_reste)
                        DL_subFrame(k,4+N_ofdmaSymbole2+1) = 10*2;
                    end

                elseif (I == 10) % CombinaisonB 3
                    sprintf('%s','Combinaison 3 .....')

                    b(10) = 2000000;
                    N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole3_reste = mod(Burst_3,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole3
                            DL_subFrame(k,m) = 10*3;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole3_reste)
                        DL_subFrame(k,4+N_ofdmaSymbole3+1) = 10*3;
                    end
```

```
                elseif (I == 13) % CombinaisonB 4
                    sprintf('%s','Combinaison 4 .....')

                    b(13) = 2000000;
                    N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole4_reste = mod(Burst_4,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole4
                            DL_subFrame(k,m) = 10*4;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole4_reste)
                        DL_subFrame(k,4+N_ofdmaSymbole4+1) = 10*4;
                    end

                elseif (I == 15) % CombinaisonB 5
                    sprintf('%s','Combinaison 5 .....')

                    b(15) = 2000000;
                    N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole5_reste = mod(Burst_5,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole5
                            DL_subFrame(k,m) = 10*5;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole4_reste)
                        DL_subFrame(k,4+N_ofdmaSymbole5+1) = 10*5;
                    end

                elseif (I == 2) % CombinaisonB 12
                    sprintf('%s','Combinaison 12 .....')

                    b(2) = 2000000;
                    N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole1_reste = mod(Burst_1,((Nsubchannel -
N_ulmapsubchannel)));
                    N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole2_reste = mod(Burst_2,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole1
                            DL_subFrame(k,m) = 10*1;
                        end
                        for m = 4 + N_ofdmaSymbole1+2:1:4 +
N_ofdmaSymbole2 + N_ofdmaSymbole1+1
```

```
                            DL_subFrame(k,m) = 10*2;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole1 + 1) = 10*1;
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole2_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole2 +
N_ofdmaSymbole1+2) = 10*2;
                    end

                elseif (I == 3) % CombinaisonB 13
                    sprintf('%s','Combinaison 13 .....')

                    b(3) = 2000000;
                    N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole1_reste = mod(Burst_1,((Nsubchannel -
N_ulmapsubchannel)));
                    N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole3_reste = mod(Burst_3,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole1
                            DL_subFrame(k,m) = 10*1;
                        end
                        for m = 4 + N_ofdmaSymbole1+2:1:4 +
N_ofdmaSymbole3 + N_ofdmaSymbole1+1
                            DL_subFrame(k,m) = 10*3;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole1 + 1) = 10*1;
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole3_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole3 +
N_ofdmaSymbole1+2) = 10*3;
                    end

                elseif (I == 4) % CombinaisonB 14
                    sprintf('%s','Combinaison 14 .....')

                    b(4) = 2000000;
                    N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole1_reste = mod(Burst_1,((Nsubchannel -
N_ulmapsubchannel)));
                    N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
```

```
                    N_ofdmaSymbole4_reste = mod(Burst_4,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole1
                            DL_subFrame(k,m) = 10*1;
                        end
                        for m = 4 + N_ofdmaSymbole1+2:1:4 +
N_ofdmaSymbole4 + N_ofdmaSymbole1+1
                            DL_subFrame(k,m) = 10*4;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole1 + 1) = 10*1;
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole4_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole4 +
N_ofdmaSymbole1+2) = 10*4;
                    end

                elseif (I == 5) % CombinaisonB 15
                    sprintf('%s','Combinaison 15 .....')

                    b(5) = 2000000;
                    N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole1_reste = mod(Burst_1,((Nsubchannel -
N_ulmapsubchannel)));
                    N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole5_reste = mod(Burst_5,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole1
                            DL_subFrame(k,m) = 10*1;
                        end
                        for m = 4 + N_ofdmaSymbole1+2:1:4 +
N_ofdmaSymbole5 + N_ofdmaSymbole1+1
                            DL_subFrame(k,m) = 10*5;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole1_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole1 + 1) = 10*1;
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole5_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole5 +
N_ofdmaSymbole1+2) = 10*5;
                    end

                elseif (I == 7) % CombinaisonB 23
                    sprintf('%s','Combinaison 23 .....')
```

```
                    b(7) = 2000000;
                    N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole2_reste = mod(Burst_2,((Nsubchannel -
N_ulmapsubchannel)));
                    N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole3_reste = mod(Burst_3,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole2
                            DL_subFrame(k,m) = 10*2;
                        end
                        for m = 4 + N_ofdmaSymbole2+2:1:4 +
N_ofdmaSymbole3 + N_ofdmaSymbole2+1
                            DL_subFrame(k,m) = 10*3;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole2_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole2 + 1) = 10*2;
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole3_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole3 +
N_ofdmaSymbole2+2) = 10*3;
                    end

                elseif (I == 8) % CombinaisonB 24
                    sprintf('%s','Combinaison 24 .....')

                    b(8) = 2000000;
                    N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole2_reste = mod(Burst_2,((Nsubchannel -
N_ulmapsubchannel)));
                    N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole4_reste = mod(Burst_4,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole2
                            DL_subFrame(k,m) = 10*2;
                        end
                        for m = 4 + N_ofdmaSymbole2+2:1:4 +
N_ofdmaSymbole4 + N_ofdmaSymbole2+1
                            DL_subFrame(k,m) = 10*4;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole2_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole2 + 1) = 10*2;
                    end
```

```
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole4_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole4 +
N_ofdmaSymbole2+2) = 10*4;
                    end

                elseif (I == 9) % CombinaisonB 25
                    sprintf('%s','Combinaison 25 .....')

                    b(9) = 2000000;
                    N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole2_reste = mod(Burst_2,((Nsubchannel -
N_ulmapsubchannel)));
                    N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole5_reste = mod(Burst_5,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole2
                            DL_subFrame(k,m) = 10*2;
                        end
                        for m = 4 + N_ofdmaSymbole2+2:1:4 +
N_ofdmaSymbole5 + N_ofdmaSymbole2+1
                            DL_subFrame(k,m) = 10*5;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole2_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole2 + 1) = 10*2;
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole5_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole5 +
N_ofdmaSymbole2+2) = 10*5;
                    end

                elseif (I == 11) % CombinaisonB 34
                    sprintf('%s','Combinaison 34 .....')

                    b(11) = 2000000;
                    N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole3_reste = mod(Burst_3,((Nsubchannel -
N_ulmapsubchannel)));
                    N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole4_reste = mod(Burst_4,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole3
                            DL_subFrame(k,m) = 10*3;
                        end
```

```
                        for m = 4 + N_ofdmaSymbole3+2:1:4 +
N_ofdmaSymbole4 + N_ofdmaSymbole3+1
                            DL_subFrame(k,m) = 10*4;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole3_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole3 + 1) = 10*3;
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole4_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole4 +
N_ofdmaSymbole3+2) = 10*4;
                    end

                elseif (I == 12) % CombinaisonB 35
                    sprintf('%s','Combinaison 35 .....')

                    b(12) = 2000000;
                    N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole3_reste = mod(Burst_3,((Nsubchannel -
N_ulmapsubchannel)));
                    N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole5_reste = mod(Burst_5,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole3
                            DL_subFrame(k,m) = 10*3;
                        end
                        for m = 4 + N_ofdmaSymbole3+2:1:4 +
N_ofdmaSymbole5 + N_ofdmaSymbole3+1
                            DL_subFrame(k,m) = 10*5;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole3_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole3 + 1) = 10*3;
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole5_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole5 +
N_ofdmaSymbole3+2) = 10*5;
                    end

                    elseif (I == 14) % CombinaisonB 45
                        sprintf('%s','Combinaison 45 .....')

                    b(14) = 2000000;
                    N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole4_reste = mod(Burst_4,((Nsubchannel -
N_ulmapsubchannel)));
```

```
                    N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*(Nsubchannel - N_ulmapsubchannel)));
                    N_ofdmaSymbole5_reste = mod(Burst_5,((Nsubchannel -
N_ulmapsubchannel)));
                    for k = N_ulmapsubchannel+1:1:Nsubchannel
                        for m = 4:1:4 + N_ofdmaSymbole4
                            DL_subFrame(k,m) = 10*4;
                        end
                        for m = 4 + N_ofdmaSymbole4+2:1:4 +
N_ofdmaSymbole5 + N_ofdmaSymbole4+1
                            DL_subFrame(k,m) = 10*5;
                        end
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole4_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole4 + 1) = 10*4;
                    end
                    for k =
N_ulmapsubchannel+1:1:N_ulmapsubchannel+floor(N_ofdmaSymbole5_reste)
                        DL_subFrame(k,4 + N_ofdmaSymbole5 +
N_ofdmaSymbole4+2) = 10*5;
                    end
                end

            SecondRemplissage = 0;

    while (1)
        lastPacket = lastPacket + 1;;
        Profil = BurstProfil(lastPacket);
        Taille = TaillePaquet(lastPacket);

        if Profil == 1
            if Taille*8/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
Taille*8/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
Taille*8/DataCarrier;
            else
                break;
            end
         elseif Profil == 2
             if 0.5*Taille*8*(4/3)/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
0.5*Taille*8*(4/3)/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
0.5*Taille*8*(4/3)/DataCarrier;
             else
                break;
            end
         elseif Profil == 3
             if 0.25*Taille*16/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
0.25*Taille*16/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
0.25*Taille*16/DataCarrier;
```

```
            else
               break;
            end
         elseif Profil == 4
            if 0.25*Taille*8*(4/3)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
0.25*Taille*8*(4/3)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
0.25*Taille*8*(4/3)/DataCarrier;
            else
               break;
            end
         elseif Profil == 5
            if 1/6*Taille*8*(3/2)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
1/6*Taille*8*(3/2)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
1/6*Taille*8*(3/2)/DataCarrier;
            else
               break;
            end
         end
    end

            else


%###################  Fin de l'allocation des DL Burst_i  ###############

 SecondRemplissage = 0;

 while (1)
        lastPacket = lastPacket + 1;;
        Profil = BurstProfil(lastPacket);
        Taille = TaillePaquet(lastPacket);

        if Profil == 1
            if Taille*8/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
Taille*8/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
Taille*8/DataCarrier;
            else
               break;
            end
         elseif Profil == 2
            if 0.5*Taille*8*(4/3)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
0.5*Taille*8*(4/3)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
0.5*Taille*8*(4/3)/DataCarrier;
            else
               break;
            end
```

```
        elseif Profil == 3
            if 0.25*Taille*16/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
0.25*Taille*16/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
0.25*Taille*16/DataCarrier;
            else
               break;
            end
        elseif Profil == 4
            if 0.25*Taille*8*(4/3)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
0.25*Taille*8*(4/3)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
0.25*Taille*8*(4/3)/DataCarrier;
            else
               break;
           end
        elseif Profil == 5
            if 1/6*Taille*8*(3/2)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
1/6*Taille*8*(3/2)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
1/6*Taille*8*(3/2)/DataCarrier;
            else
               break;
           end
        end
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% SOLUTION C: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            for i=1:1:10
                CombinaisonC1 = -(Burst_1 -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);
                CombinaisonC12 = -((Burst_1 + Burst_2) -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);
                CombinaisonC13 = -((Burst_1 + Burst_3) -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);
                CombinaisonC14 = -((Burst_1 + Burst_4) -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);
                CombinaisonC15 = -((Burst_1 + Burst_5) -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);

                CombinaisonC2 = -(Burst_2 -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);
                CombinaisonC23 = -((Burst_2 + Burst_3) -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);
                CombinaisonC24 = -((Burst_2 + Burst_4) -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);
```

```
                    CombinaisonC25 = -((Burst_2 + Burst_5) -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);

                    CombinaisonC3 = -(Burst_3 -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);
                    CombinaisonC34 = -((Burst_3 + Burst_4) -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);
                    CombinaisonC35 = -((Burst_3 + Burst_5) -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);

                    CombinaisonC4 = -(Burst_4 -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);
                    CombinaisonC45 = -((Burst_4 + Burst_5) -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);

                    CombinaisonC5 = -(Burst_5 -
(N_ulmapsubchannel+i)*DataCarrier*DL_s);

                    CombinaisonC = [CombinaisonC1 CombinaisonC12
CombinaisonC13 CombinaisonC14 CombinaisonC15 CombinaisonC2 CombinaisonC23
CombinaisonC24 CombinaisonC25 CombinaisonC3 CombinaisonC34 CombinaisonC35
CombinaisonC4 CombinaisonC45 CombinaisonC5];

                    if (CombinaisonC1 < Reg1+DataCarrier*i & CombinaisonC1
> 0)| (CombinaisonC12 < 2*Reg1+DataCarrier*i  & CombinaisonC12 >
Reg1+DataCarrier*i) |(CombinaisonC13 < 2*Reg1+DataCarrier*i  &
CombinaisonC13 > Reg1+DataCarrier*i) |(CombinaisonC14 <
2*Reg1+DataCarrier*i  & CombinaisonC14 > Reg1+DataCarrier*i)|
(CombinaisonC15 < 2*Reg1+DataCarrier*i  & CombinaisonC15 >
Reg1+DataCarrier*i)| (CombinaisonC2 < Reg1+DataCarrier*i  & CombinaisonC2
> 0) | (CombinaisonC23 < 2*Reg1+DataCarrier*i  & CombinaisonC23 >
Reg1+DataCarrier*i ) |(CombinaisonC24 < 2*Reg1+DataCarrier*i  &
CombinaisonC24 > Reg1+DataCarrier*i) |(CombinaisonC25 <
2*Reg1+DataCarrier*i  & CombinaisonC25 > Reg1+DataCarrier*i)
|(CombinaisonC3 < Reg1+DataCarrier*i  & CombinaisonC3 > 0)
|(CombinaisonC34 < 2*Reg1+DataCarrier*i  & CombinaisonC34 >
Reg1+DataCarrier*i) (CombinaisonC35 < 2*Reg1+DataCarrier*i  &
CombinaisonC35 > Reg1+DataCarrier*i)| (CombinaisonC4 < Reg1+DataCarrier*i
& CombinaisonC4 > 0) |(CombinaisonC45 < 2*Reg1+DataCarrier*i  &
CombinaisonC45 > Reg1+DataCarrier*i) |(CombinaisonC5 < Reg1+DataCarrier*i
& CombinaisonC5 > 0)


                        sprintf('%s','SOLUTION C !!!!!')

                        for j = 1:1:15
                            c(j) = CombinaisonC(j);
                            if (CombinaisonC(j) < 0)
                                c(j) = 1000000; % maximiser les négatifs
pour qu'ils n'influencent pas sur la recherche du min
                            end
                        end
```

```
                        [C,I] =
min([c(1),c(2),c(3),c(4),c(5),c(6),c(7),c(8),c(9),c(10),c(11),c(12),c(13),
c(14),c(15)]);

                        if (I == 1) % CombinaisonC 1
                            sprintf('%s','Combinaison 1 .....')

                            c(1) = 2000000;
                            N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole1_reste =
mod(Burst_1,((N_ulmapsubchannel+i)));
                            for k = 1:1:(N_ulmapsubchannel+i)
                                for m = 6:1:6 + N_ofdmaSymbole1
                                    DL_subFrame(k,m) = 10*1;
                                end
                            end
                            for k = 1:1:floor(N_ofdmaSymbole1_reste)
                                DL_subFrame(k,6+N_ofdmaSymbole1+1) = 10*1;
                            end

                            break;

                        elseif (I == 6) % CombinaisonC 2
                            sprintf('%s','Combinaison 2 .....')

                            c(6) = 2000000;
                            N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole2_reste =
mod(Burst_2,((N_ulmapsubchannel+i)));
                            for k = 1:1:(N_ulmapsubchannel+i)
                                for m = 6:1:6 + N_ofdmaSymbole2
                                    DL_subFrame(k,m) = 10*2;
                                end
                            end
                            for k = 1:1:floor(N_ofdmaSymbole2_reste)
                                DL_subFrame(k,6+N_ofdmaSymbole2+1) = 10*2;
                            end

                            break;

                        elseif (I == 10) % CombinaisonC 3
                            sprintf('%s','Combinaison 3 .....')

                            c(10) = 2000000;
                            N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole3_reste =
mod(Burst_3,((N_ulmapsubchannel+i)));
                            for k = 1:1:(N_ulmapsubchannel+i)
                                for m = 6:1:6 + N_ofdmaSymbole3
                                    DL_subFrame(k,m) = 10*3;
                                end
```

```
                    end
                    for k = 1:1:floor(N_ofdmaSymbole2_reste)
                        DL_subFrame(k,6+N_ofdmaSymbole3+1) = 10*3;
                    end

                    break;

                elseif (I == 13) % CombinaisonC 4
                    sprintf('%s','Combinaison 4 !!!!!')

                    c(13) = 2000000;
                    N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*(N_ulmapsubchannel+i)));
                    N_ofdmaSymbole4_reste =
mod(Burst_4,((N_ulmapsubchannel+i)));
                    for k = 1:1:(N_ulmapsubchannel+i)
                        for m = 6:1:6 + N_ofdmaSymbole4
                            DL_subFrame(k,m) = 10*4;
                        end
                    end
                    for k = 1:1:floor(N_ofdmaSymbole2_reste)
                        DL_subFrame(k,6+N_ofdmaSymbole2+1) = 10*4;
                    end

                    break;

                elseif (I == 15) % CombinaisonC 5
                    sprintf('%s','Combinaison 5 .....')

                    c(15) = 2000000;
                    N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*(N_ulmapsubchannel+i)));
                    N_ofdmaSymbole5_reste =
mod(Burst_5,((N_ulmapsubchannel+i)));
                    for k = 1:1:(N_ulmapsubchannel+i)
                        for m = 6:1:6 + N_ofdmaSymbole5
                            DL_subFrame(k,m) = 10*5;
                        end
                    end
                    for k = 1:1:floor(N_ofdmaSymbole2_reste)
                        DL_subFrame(k,6+N_ofdmaSymbole5+1) = 10*5;
                    end

                    break;

                elseif (I == 2) % CombinaisonC 12
                    sprintf('%s','Combinaison 12 .....')

                    c(2) = 2000000;
                    N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*(N_ulmapsubchannel+i)));
                    N_ofdmaSymbole1_reste =
mod(Burst_1,((N_ulmapsubchannel+i)));
```

```
                            N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole2_reste =
mod(Burst_2,((N_ulmapsubchannel+i)));
                            for k = 1:1:(N_ulmapsubchannel+i)
                                for m = 6:1:6 + N_ofdmaSymbole1
                                    DL_subFrame(k,m) = 10*1;
                                end
                                for m = 6 + N_ofdmaSymbole1+2:1:6 +
N_ofdmaSymbole2 + N_ofdmaSymbole1+1
                                    DL_subFrame(k,m) = 10*2;
                                end
                            end
                            for k = 1:1:floor(N_ofdmaSymbole1_reste)
                                DL_subFrame(k,6 + N_ofdmaSymbole1 + 1) =
10*1;
                            end
                            for k = 1:1:floor(N_ofdmaSymbole2_reste)
                                DL_subFrame(k,6 + N_ofdmaSymbole2 +
N_ofdmaSymbole1+2) = 10*2;
                            end

                            break;

                        elseif (I == 3) % CombinaisonC 13
                            sprintf('%s','Combinaison 13 .....')

                            c(3) = 2000000;
                            N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole1_reste =
mod(Burst_1,((N_ulmapsubchannel+i)));
                            N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole3_reste =
mod(Burst_3,((N_ulmapsubchannel+i)));
                            for k = 1:1:(N_ulmapsubchannel+i)
                                for m = 6:1:6 + N_ofdmaSymbole1
                                    DL_subFrame(k,m) = 10*1;
                                end
                                for m = 6 + N_ofdmaSymbole1+2:1:6 +
N_ofdmaSymbole3 + N_ofdmaSymbole1+1
                                    DL_subFrame(k,m) = 10*3;
                                end
                            end
                            for k = 1:1:floor(N_ofdmaSymbole1_reste)
                                DL_subFrame(k,6 + N_ofdmaSymbole1 + 1) =
10*1;
                            end
                            for k = 1:1:floor(N_ofdmaSymbole3_reste)
                                DL_subFrame(k,6 + N_ofdmaSymbole3 +
N_ofdmaSymbole1+2) = 10*3;
                            end
```

```
                    break;

                elseif (I == 4) % CombinaisonC 14
                    sprintf('%s','Combinaison 14 .....')

                    c(4) = 2000000;
                    N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*(N_ulmapsubchannel+i)));
                    N_ofdmaSymbole1_reste =
mod(Burst_1,((N_ulmapsubchannel+i)));
                    N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*(N_ulmapsubchannel+i)));
                    N_ofdmaSymbole4_reste =
mod(Burst_4,((N_ulmapsubchannel+i)));
                    for k = 1:1:(N_ulmapsubchannel+i)
                        for m = 6:1:6 + N_ofdmaSymbole1
                            DL_subFrame(k,m) = 10*1;
                        end
                        for m = 6 + N_ofdmaSymbole1+2:1:6 +
N_ofdmaSymbole4 + N_ofdmaSymbole1+1
                            DL_subFrame(k,m) = 10*4;
                        end
                    end
                    for k = 1:1:floor(N_ofdmaSymbole1_reste)
                        DL_subFrame(k,6 + N_ofdmaSymbole1 + 1) =
10*1;
                    end
                    for k = 1:1:floor(N_ofdmaSymbole4_reste)
                        DL_subFrame(k,6 + N_ofdmaSymbole4 +
N_ofdmaSymbole1+2) = 10*4;
                    end

                    break;

                elseif (I == 5) % CombinaisonC 15
                    sprintf('%s','Combinaison 15 !.....')

                    c(5) = 2000000;
                    N_ofdmaSymbole1 =
floor(Burst_1/(DataCarrier*(N_ulmapsubchannel+i)));
                    N_ofdmaSymbole1_reste =
mod(Burst_1,((N_ulmapsubchannel+i)));
                    N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*(N_ulmapsubchannel+i)));
                    N_ofdmaSymbole5_reste =
mod(Burst_5,((N_ulmapsubchannel+i)));
                    for k = 1:1:(N_ulmapsubchannel+i)
                        for m = 6:1:6 + N_ofdmaSymbole1
                            DL_subFrame(k,m) = 10*1;
                        end
                        for m = 6 + N_ofdmaSymbole1+2:1:6 +
N_ofdmaSymbole5 + N_ofdmaSymbole1+1
                            DL_subFrame(k,m) = 10*5;
                        end
```

```
                            end
                            for k = 1:1:floor(N_ofdmaSymbole1_reste)
                                DL_subFrame(k,6 + N_ofdmaSymbole1 + 1) =
10*1;
                            end
                            for k = 1:1:floor(N_ofdmaSymbole5_reste)
                                DL_subFrame(k,6 + N_ofdmaSymbole5 +
N_ofdmaSymbole1+2) = 10*5;
                            end

                            break;

                        elseif (I == 7) % CombinaisonC 23
                            sprintf('%s','Combinaison 23 .....')

                            c(7) = 2000000;
                            N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole2_reste =
mod(Burst_2,((N_ulmapsubchannel+i)));
                            N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole3_reste =
mod(Burst_3,((N_ulmapsubchannel+i)));
                            for k = 1:1:(N_ulmapsubchannel+i)
                                for m = 6:1:6 + N_ofdmaSymbole2
                                    DL_subFrame(k,m) = 10*2;
                                end
                                for m = 6 + N_ofdmaSymbole2+2:1:6 +
N_ofdmaSymbole3 + N_ofdmaSymbole2+1
                                    DL_subFrame(k,m) = 10*3;
                                end
                            end
                            for k = 1:1:floor(N_ofdmaSymbole2_reste)
                                DL_subFrame(k,6 + N_ofdmaSymbole2 + 1) =
10*2;
                            end
                            for k = 1:1:floor(N_ofdmaSymbole3_reste)
                                DL_subFrame(k,6 + N_ofdmaSymbole3 +
N_ofdmaSymbole2+2) = 10*3;
                            end

                            break;

                        elseif (I == 8) % CombinaisonC 24
                            sprintf('%s','Combinaison 24 .....')

                            c(8) = 2000000;
                            N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole2_reste =
mod(Burst_2,((N_ulmapsubchannel+i)));
                            N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*(N_ulmapsubchannel+i)));
```

```
                            N_ofdmaSymbole4_reste =
mod(Burst_4,((N_ulmapsubchannel+i)));
                            for k = 1:1:(N_ulmapsubchannel+i)
                                for m = 6:1:6 + N_ofdmaSymbole2
                                    DL_subFrame(k,m) = 10*2;
                                end
                                for m = 6 + N_ofdmaSymbole2+2:1:6 +
N_ofdmaSymbole4 + N_ofdmaSymbole2+1
                                    DL_subFrame(k,m) = 10*4;
                                end
                            end
                            for k = 1:1:floor(N_ofdmaSymbole2_reste)
                                DL_subFrame(k,6 + N_ofdmaSymbole2 + 1) =
10*2;
                            end
                            for k = 1:1:floor(N_ofdmaSymbole4_reste)
                                DL_subFrame(k,6 + N_ofdmaSymbole2 +
N_ofdmaSymbole4+2) = 10*4;
                            end

                            break;

                        elseif (I == 9) % CombinaisonC 25
                            sprintf('%s','Combinaison 25 .....')

                            c(9) = 2000000;
                            N_ofdmaSymbole2 =
floor(Burst_2/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole2_reste =
mod(Burst_2,((N_ulmapsubchannel+i)));
                            N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole5_reste =
mod(Burst_5,((N_ulmapsubchannel+i)));
                            for k = 1:1:(N_ulmapsubchannel+i)
                                for m = 6:1:6 + N_ofdmaSymbole2
                                    DL_subFrame(k,m) = 10*2;
                                end
                                for m = 6 + N_ofdmaSymbole2+2:1:6 +
N_ofdmaSymbole5 + N_ofdmaSymbole2+1
                                    DL_subFrame(k,m) = 10*5;
                                end
                            end
                            for k = 1:1:floor(N_ofdmaSymbole2_reste)
                                DL_subFrame(k,6 + N_ofdmaSymbole2 + 1) =
10*2;
                            end
                            for k = 1:1:floor(N_ofdmaSymbole5_reste)
                                DL_subFrame(k,6 + N_ofdmaSymbole2 +
N_ofdmaSymbole5+2) = 10*5;
                            end

                            break;
```

```
                        elseif (I == 11) % CombinaisonC 34
                            sprintf('%s','Combinaison 34 .....')

                            c(11) = 2000000;
                            N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole3_reste =
mod(Burst_3,((N_ulmapsubchannel+i)));
                            N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole4_reste =
mod(Burst_4,((N_ulmapsubchannel+i)));
                            for k = 1:1:(N_ulmapsubchannel+i)
                                for m = 6:1:6 + N_ofdmaSymbole3
                                    DL_subFrame(k,m) = 10*3;
                                end
                                for m = 6 + N_ofdmaSymbole3+2:1:6 +
N_ofdmaSymbole3 + N_ofdmaSymbole4+1
                                    DL_subFrame(k,m) = 10*4;
                                end
                            end
                            for k = 1:1:floor(N_ofdmaSymbole3_reste)
                                DL_subFrame(k,6 + N_ofdmaSymbole3 + 1) =
10*3;
                            end
                            for k = 1:1:floor(N_ofdmaSymbole4_reste)
                                DL_subFrame(k,6 + N_ofdmaSymbole3 +
N_ofdmaSymbole4+2) = 10*4;
                            end

                            break;

                        elseif (I == 12) % CombinaisonC 35
                            sprintf('%s','Combinaison 35 .....')

                            c(12) = 2000000;
                            N_ofdmaSymbole3 =
floor(Burst_3/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole3_reste =
mod(Burst_3,((N_ulmapsubchannel+i)));
                            N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*(N_ulmapsubchannel+i)));
                            N_ofdmaSymbole5_reste =
mod(Burst_5,((N_ulmapsubchannel+i)));
                            for k = 1:1:(N_ulmapsubchannel+i)
                                for m = 6:1:6 + N_ofdmaSymbole3
                                    DL_subFrame(k,m) = 10*3;
                                end
                                for m = 6 + N_ofdmaSymbole3+2:1:6 +
N_ofdmaSymbole3 + N_ofdmaSymbole5+1
                                    DL_subFrame(k,m) = 10*5;
                                end
                            end
                            for k = 1:1:floor(N_ofdmaSymbole3_reste)
```

```
                                    DL_subFrame(k,6 + N_ofdmaSymbole3 + 1) =
10*3;
                                end
                                for k = 1:1:floor(N_ofdmaSymbole5_reste)
                                    DL_subFrame(k,6 + N_ofdmaSymbole3 +
N_ofdmaSymbole5+2) = 10*5;
                                end

                                break;

                            elseif (I == 14) % CombinaisonC 45
                                sprintf('%s','Combinaison 45 .....')

                                c(14) = 2000000;
                                N_ofdmaSymbole4 =
floor(Burst_4/(DataCarrier*(N_ulmapsubchannel+i)));
                                N_ofdmaSymbole4_reste =
mod(Burst_4,((N_ulmapsubchannel+i)));
                                N_ofdmaSymbole5 =
floor(Burst_5/(DataCarrier*(N_ulmapsubchannel+i)));
                                N_ofdmaSymbole5_reste =
mod(Burst_5,((N_ulmapsubchannel+i)));
                                for k = 1:1:(N_ulmapsubchannel+i)
                                    for m = 6:1:6 + N_ofdmaSymbole4
                                        DL_subFrame(k,m) = 10*4;
                                    end
                                    for m = 6 + N_ofdmaSymbole4+2:1:6 +
N_ofdmaSymbole4 + N_ofdmaSymbole5+1
                                        DL_subFrame(k,m) = 10*5;
                                    end
                                end
                                for k = 1:1:floor(N_ofdmaSymbole4_reste)
                                    DL_subFrame(k,6 + N_ofdmaSymbole4 + 1) =
10*4;
                                end
                                for k = 1:1:floor(N_ofdmaSymbole5_reste)
                                    DL_subFrame(k,6 + N_ofdmaSymbole3 +
N_ofdmaSymbole5+2) = 10*5;
                                end

                                break;
                            end

                        end
                    end

%####################  Fin de l'allocation des DL Burst_i  ###############

SecondRemplissage = 0;
while (1)
        lastPacket = lastPacket + 1;;
        Profil = BurstProfil(lastPacket);
        Taille = TaillePaquet(lastPacket);
```

```
        if Profil == 1
            if Taille*8/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
Taille*8/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
Taille*8/DataCarrier;
            else
                break;
            end
         elseif Profil == 2
             if 0.5*Taille*8*(4/3)/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
0.5*Taille*8*(4/3)/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
0.5*Taille*8*(4/3)/DataCarrier;
             else
                break;
            end
         elseif Profil == 3
             if 0.25*Taille*16/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
0.25*Taille*16/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
0.25*Taille*16/DataCarrier;
             else
                break;
             end
         elseif Profil == 4
             if 0.25*Taille*8*(4/3)/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
0.25*Taille*8*(4/3)/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
0.25*Taille*8*(4/3)/DataCarrier;
             else
                break;
            end
         elseif Profil == 5
             if 1/6*Taille*8*(3/2)/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
1/6*Taille*8*(3/2)/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
1/6*Taille*8*(3/2)/DataCarrier;
             else
                break;
            end
         end
    end
        end

    end
%##########################################################################

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
 tps = toc;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%#########################################################################
vide = 0;
iiiii = 1;
for i = 1:1:Nsubchannel
    for j = 1:1:Nofdmasymbol
        if DL_subFrame(i,j) == 0
            vide = vide + 1;
        end
    end
end

    TauxDeRemplissage = (Nsubchannel*(Nofdmasymbol+UL_OFDMA_Symb) - vide -
0.5*Nombre_profil)/(Nsubchannel*(Nofdmasymbol+UL_OFDMA_Symb))*100;
    Taux(iii)=TauxDeRemplissage;
end
sprintf('le taux de remplissage de la trame est : %0.5g pourcent',
mean(Taux));
mean(Taux)
iii=1:1:500;
mean(Temps)

%#########################################################################
```

ANNEXE II

Algorithme 2

```
%##########################################################################
%initialiser tout
clc;
clear all;
%##########################################################################
    % PUSC : Data=24
    % FUSC : Data=48
    mode = 3; %3=PUSC,  2=FUSC
    Nombre_profil = 8; % 5, 6, 7 ou 8
    FFT = 1024; % 512, 1024 ou 2048
    Ratio_DL_UL = 3/1; % 3/2, 1/1, 2/3, 2/1, 1/2, 3/1 ou 1/3
    T_frame = 0.005; % 0.05 = 5 ms, 0.01 = 10 ms ou 0.02 = 20 ms

    if mode == 3
        DataCarrier = 24;
        if FFT == 2048
            Nsubchannel = 60;
        elseif FFT == 1024
            Nsubchannel = 30;
        elseif FFT == 512
            Nsubchannel = 15;
        end
    elseif mode == 2
        DataCarrier = 48;
        if FFT == 2048
            Nsubchannel = 32;
        elseif FFT == 1024
            Nsubchannel = 16;
        elseif FFT == 512
            Nsubchannel = 8;
        end
    end

    if T_frame == 0.005
        N_OFDMA_Symb = 48; %Trame de 5 ms
    elseif T_frame == 0.01
        N_OFDMA_Symb = 96; %Trame de 10 ms
    elseif T_frame == 0.02
        N_OFDMA_Symb = 192; %Trame de 20 ms
    end
    if Ratio_DL_UL == 3/2
        DL_OFDMA_Symb = floor(N_OFDMA_Symb*3/5) - 2;
    elseif Ratio_DL_UL == 1/1
        DL_OFDMA_Symb = floor(N_OFDMA_Symb*0.5) - 2;
    elseif Ratio_DL_UL == 2/3
        DL_OFDMA_Symb = floor(N_OFDMA_Symb*2/5) - 2;
    elseif Ratio_DL_UL == 2/1
        DL_OFDMA_Symb = floor(N_OFDMA_Symb*2/3) - 2;
    elseif Ratio_DL_UL == 1/2
        DL_OFDMA_Symb = floor(N_OFDMA_Symb*1/3) - 2;
    elseif Ratio_DL_UL == 3/1
        DL_OFDMA_Symb = floor(N_OFDMA_Symb*3/4) - 2;
    elseif Ratio_DL_UL == 1/3
        DL_OFDMA_Symb = floor(N_OFDMA_Symb*1/4) - 2;
    end
```

```
    UL_OFDMA_Symb = N_OFDMA_Symb - DL_OFDMA_Symb  - 4;
    N_ulmapsubchannel = floor(Nsubchannel/2);
%##########################################################################
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tableau des CID
for k=1:1:1000
    CID(k)=k;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tableau des Burst profil

for iii = 1:1:500
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tableau des Burst profil
BurstProfil = randsrc(1000,1,[1 2 3 4 5 6 7 8]);
TaillePaquet = poissrnd(200,1,1000);

%##########################################################################
%Construction de la trame

    %initialisation de la trame
    DL_subFrame = zeros(Nsubchannel,DL_OFDMA_Symb);
    UL_subFrame = zeros(Nsubchannel,UL_OFDMA_Symb);

    if Nombre_profil == 5
        a6 = 0;a7 = 0;a8 = 0;
    elseif Nombre_profil == 6
        a6 = 1;a7 = 0;a8 = 0;
    elseif Nombre_profil == 7
        a6 = 1;a7 = 1;a8 = 0;
    elseif Nombre_profil == 8
        a6 = 1;a7 = 0;a8 = 1;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DL_subframe
    %Preamble
    for k = 1:1:Nsubchannel
        DL_subFrame(k,1) = 0.5*10;
    end

    %FCH
    for k = 1:1:2
        for m = 2:1:3
            DL_subFrame(k,m) = 1*10;
        end
    end

    %DL_MAP
    for k = 3:1:Nsubchannel
        for m = 2:1:3
            DL_subFrame(k,m) = 1*10;
```

```
        end
    end

    %UL_MAP

    for k = 1:1:N_ulmapsubchannel
        for m = 4:1:5
            DL_subFrame(k,m) = 1*10;
        end
    end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% UL_subframe
    Total_UL = 0;
    slot_tot = UL_OFDMA_Symb*Nsubchannel;
    % Ranging
    for m = 1:1:UL_OFDMA_Symb
        for k = 1:1:2
            UL_subFrame(k,m) = 5;
            Total_UL = Total_UL + 2*DataCarrier;
        end
    end
    %Burst1
    for m = 1:1:UL_OFDMA_Symb
        for k = 3:1:floor(Nsubchannel/5)
            UL_subFrame(k,m) = 1*10;
            Total_UL = Total_UL + 2*DataCarrier;
        end
    end
    for m = 1:1:floor(UL_OFDMA_Symb/2)
        UL_subFrame(floor(Nsubchannel/5)+1,m) = 1*10;
        Total_UL = Total_UL + 2*DataCarrier;
    end

    %Burst2
    for m = floor(UL_OFDMA_Symb/2+0.5)+1:1:UL_OFDMA_Symb
        UL_subFrame(floor(Nsubchannel/5)+1,m) = 2*10;
        Total_UL = Total_UL + 2*DataCarrier;
    end
    for m = 1:1:UL_OFDMA_Symb
        for k = floor(Nsubchannel/5)+2:1:2*floor(Nsubchannel/5)
            UL_subFrame(k,m) = 2*10;
            Total_UL = Total_UL + 2*DataCarrier;
        end
    end

    %Burst3
    for m = 1:1:UL_OFDMA_Symb
        for k = 2*floor(Nsubchannel/5)+1:1:3*floor(Nsubchannel/5)
            UL_subFrame(k,m) = 3*10;
            Total_UL = Total_UL + 4*DataCarrier;
        end
```

```
    end
    for m = 1:1:floor(UL_OFDMA_Symb/2)
        UL_subFrame(3*floor(Nsubchannel/5)+1,m) = 3*10;
        Total_UL = Total_UL + 4*DataCarrier;
    end

    %Burst4
    for m = floor(UL_OFDMA_Symb/2+0.5)+1:1:UL_OFDMA_Symb
        UL_subFrame(3*floor(Nsubchannel/5)+1,m) = 4*10;
        Total_UL = Total_UL + 4*DataCarrier;
    end
    for m = 1:1:UL_OFDMA_Symb
        for k = 3*floor(Nsubchannel/5)+2:1:4*floor(Nsubchannel/5)
            UL_subFrame(k,m) = 4*10;
            Total_UL = Total_UL + 4*DataCarrier;
        end
    end


    %Burst5
   for m = 1:1:UL_OFDMA_Symb
        for k = 4*floor(Nsubchannel/5):1:Nsubchannel
            UL_subFrame(k,m) = 5*10;
            Total_UL = Total_UL + 6*DataCarrier;
        end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % TTG
    for k = 1:1:Nsubchannel
        for m = 1:1:2
            TTG(k,m) = 0;
        end
    end
% RTG
    for k = 1:1:Nsubchannel
        for m = 1:1:3
            RTG(k,m) = 0;
        end
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DL_subframe (LD-MAP & DL burst #1 à #Np)

  % Time slot disponibles et les 2 regions
    Nofdmasymbol = DL_OFDMA_Symb;


  %DL_MAP
    for k = 3:1:Nsubchannel
        for m = 2:1:3
            DL_subFrame(k,m) = 1*10;
```

```
        end
    end


  %DL burst #1 à #5
    %reception des paquets et traitement(mapping)
    Burst_1 = 0;Burst_2 = 0;Burst_3 = 0;Burst_4 = 0;Burst_5 = 0;Burst_6 =
0;Burst_7 = 0;Burst_8 = 0;
    k=0;
    while Burst_1 + Burst_2 + Burst_3 + Burst_4 + Burst_5 + a6*Burst_6 +
a7*Burst_7 + a8*Burst_8 < Nsubchannel*DataCarrier*Nofdmasymbol -
(2*N_ulmapsubchannel + 3*Nsubchannel + 2*N_ulmapsubchannel + (Nsubchannel-
N_ulmapsubchannel) + 34)*DataCarrier
        k = k + 1;
        if BurstProfil(k)==1
            Burst_1= Burst_1 + TaillePaquet(k)*8;

        elseif BurstProfil(k)==2
            Burst_2= Burst_2 + 0.5*TaillePaquet(k)*8*4/3;

        elseif BurstProfil(k)==3
            Burst_3= Burst_3 + 0.25*TaillePaquet(k)*8*2;

         elseif BurstProfil(k)==4
            Burst_4= Burst_4 + 0.25*TaillePaquet(k)*8*4/3;

        elseif BurstProfil(k)==5
            Burst_5= Burst_5 + 1/6*TaillePaquet(k)*8*3/2;

        elseif BurstProfil(k)==6
            Burst_6= Burst_6 + a6*0.5*TaillePaquet(k)*8*3/2;

        elseif BurstProfil(k)==7
            Burst_7= Burst_7 + a7*1/4*TaillePaquet(k)*8*4/3;

        elseif BurstProfil(k)==8
            Burst_8= Burst_8 + a8*1/6*TaillePaquet(k)*8*2;

        end

    end

    EsapceLibre = [0 0 0 0 0 0 0 0];
    lastPacket = k;


    %################  Début de l'allocation des DL Burst_i  ############

    Burst =
[Burst_1,Burst_2,Burst_3,Burst_4,Burst_5,Burst_6,Burst_7,Burst_8];
    Burst_tmp =
[Burst_1,Burst_2,Burst_3,Burst_4,Burst_5,Burst_6,Burst_7,Burst_8];
```

```
    [C,I1] = max(Burst_tmp);
    Burst_tmp(I1) = 0;
    [C,I2] = max(Burst_tmp);
    Burst_tmp(I2) = 0;
    [C,I3] = max(Burst_tmp);
    Burst_tmp(I3) = 0;
    [C,I4] = max(Burst_tmp);
    Burst_tmp(I4) = 0;
    [C,I5] = max(Burst_tmp);
    Burst_tmp(I5) = 0;
    [C,I6] = max(Burst_tmp);
    Burst_tmp(I6) = 0;
    [C,I7] = max(Burst_tmp);
    Burst_tmp(I7) = 0;
    [C,I8] = max(Burst_tmp);
    Burst_tmp(I8) = 0;

    %------------------- Commencer par Region 1 -------------------------
    M = ceil((Burst(I1) + Burst(I2))/(DataCarrier*(Nofdmasymbol-5)));
% Trouver la largeur requise pour les deux plus gros Dl_Burst dans la
région 1
    if(M >= N_ulmapsubchannel)
        p = ceil((Nofdmasymbol-5)*(Burst(I1)/(M*DataCarrier*(Nofdmasymbol-
5))));  %  Trouver La longueur du DL_Burst_1
        q = ceil((Nofdmasymbol-5)*(Burst(I2)/(M*DataCarrier*(Nofdmasymbol-
5))));  % p + q = 1
        if( p + q <= Nofdmasymbol-5 )
            sprintf('Region 1    sans ajout de subch')
            % Burst(I1)  et Burst(I2)
            reste_Burst_I1 = p*M*DataCarrier - Burst(I1);
            reste_Burst_I2 = q*M*DataCarrier - Burst(I2);
            reste_I1 = floor(reste_Burst_I1/DataCarrier);
            reste_I2 = floor(reste_Burst_I2/DataCarrier);
            EspaceLibre(I1) = (M - reste_I1);
            EsapceLibre(I2) = (M - reste_I2);
            for k = 1:1:M
                for m = 6:1:6 + p-1
                    DL_subFrame(k,m) = 10*I1;
                end
                for m = 6+p+1:1:Nofdmasymbol-1
                    DL_subFrame(k,m) = 10*I2;
                end
            end
            for k = 1:1:reste_I1
                DL_subFrame(k,p+6) = 10*I1;
            end
            for k = 1:1:reste_I2
                DL_subFrame(k,Nofdmasymbol) = 10*I2;
            end
            % Burst(I3)
            Nofdmasymbol_I3 = ceil(Burst(I3)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I3 = Nofdmasymbol_I3*(Nsubchannel-M)*DataCarrier -
Burst(I3);
```

```
            reste_I3 = floor(reste_Burst_I3/DataCarrier);
            EspaceLibre(I3) = (Nsubchannel-M - reste_I3);
            for k = M+1:1:Nsubchannel
                for m = 4:1:4 + Nofdmasymbol_I3-1
                    DL_subFrame(k,m) = 10*I3;
                end
            end
            for k = M+1:1:M+1+reste_I3
                DL_subFrame(k,4 + Nofdmasymbol_I3) = 10*I3;
            end
            % Burst(I4)
            Nofdmasymbol_I4 = ceil(Burst(I4)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I4 = Nofdmasymbol_I4*(Nsubchannel-M)*DataCarrier -
Burst(I4);
            reste_I4 = floor(reste_Burst_I4/DataCarrier);
            EspaceLibre(I4) = (Nsubchannel-M - reste_Burst_I4);
            for k = M+1:1:Nsubchannel
                for m = 4 + Nofdmasymbol_I3+1:1:4 + Nofdmasymbol_I3+1 +
Nofdmasymbol_I4-1
                    DL_subFrame(k,m) = 10*I4;
                end
            end
            for k = M+1:1:M+1+reste_I4
                DL_subFrame(k,4 + Nofdmasymbol_I3+1 + Nofdmasymbol_I4) =
10*I4;
            end
             % Burst(I5)
            Nofdmasymbol_I5 = ceil(Burst(I5)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I5 = Nofdmasymbol_I5*(Nsubchannel-M)*DataCarrier -
Burst(I5);
            reste_I5 = floor(reste_Burst_I5/DataCarrier);
            EspaceLibre(I5) = (Nsubchannel-M - reste_I5);
            for k = M+1:1:Nsubchannel
                for m = 4 + Nofdmasymbol_I3+Nofdmasymbol_I4+2:1:4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5-1
                    DL_subFrame(k,m) = 10*I5;
                end
            end
            for k = M+1:1:M+1+reste_I5
                DL_subFrame(k,4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5) = 10*I5;
            end
             % Burst(I6)
            Nofdmasymbol_I6 = ceil(Burst(I6)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I6 = Nofdmasymbol_I6*(Nsubchannel-M)*DataCarrier -
Burst(I6);
            reste_I6 = floor(reste_Burst_I6/DataCarrier);
            EspaceLibre(I6) = (Nsubchannel-M - reste_I6);
            for k = M+1:1:Nsubchannel
                for m = 4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+2:1:4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6-1
```

```
                    DL_subFrame(k,m) = 10*I6;
                end
            end
            for k = M+1:1:M+1+reste_I6
                DL_subFrame(k,4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6) = 10*I6;
            end
             % Burst(I7)
            Nofdmasymbol_I7 = ceil(Burst(I7)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I7 = Nofdmasymbol_I7*(Nsubchannel-M)*DataCarrier -
Burst(I7);
            reste_I7 = floor(reste_Burst_I7/DataCarrier);
            EspaceLibre(I7) = (Nsubchannel-M - reste_I7);
            for k = M+1:1:Nsubchannel
                for m = 4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+2:1:4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7-1
                    DL_subFrame(k,m) = 10*I7;
                end
            end
            for k = M+1:1:M+1+reste_I7
                DL_subFrame(k,4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7) = 10*I7;
            end
            % Burst(I8)
            Nofdmasymbol_I8 = ceil(Burst(I8)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I8 = Nofdmasymbol_I8*(Nsubchannel-M)*DataCarrier -
Burst(I8);
            reste_I8 = floor(reste_Burst_I8/DataCarrier);
            EspaceLibre(I8) = (Nsubchannel-M - reste_I8);
            for k = M+1:1:Nsubchannel
                for m = 4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7+2:1:4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7+Nofdmasymbol_I8-1
                    DL_subFrame(k,m) = 10*I8;
                end
            end
            for k = M+1:1:M+1+reste_I8
                DL_subFrame(k,4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7+Nofdmasymbol_I8) = 10*I8;
            end

        else
            sprintf('Region 1    avec ajout de subch')
            M=M+1;
            p=ceil((Nofdmasymbol-
5)*(Burst(I1)/((M)*DataCarrier*(Nofdmasymbol-5))));
```

```
            q = ceil((Nofdmasymbol-
5)*(Burst(I2)/(M*DataCarrier*(Nofdmasymbol-5))));  % p + q = 1
            % Burst(I1)  et Burst(I2)
            reste_Burst_I1 = p*M*DataCarrier - Burst(I1);
            reste_Burst_I2 = q*M*DataCarrier - Burst(I2);
            reste_I1 = floor(reste_Burst_I1/DataCarrier);
            reste_I2 = floor(reste_Burst_I2/DataCarrier);
            EspaceLibre(I1) = (M - reste_I1);
            EsapceLibre(I2) = (M - reste_I2);
            for k = 1:1:M
                for m = 6:1:6 + p-1
                    DL_subFrame(k,m) = 10*I1;
                end
                for m = 6+p+1:1:Nofdmasymbol-1
                    DL_subFrame(k,m) = 10*I2;
                end
            end
            for k = 1:1:reste_I1
                DL_subFrame(k,p+6) = 10*I1;
            end
            for k = 1:1:reste_I2
                DL_subFrame(k,Nofdmasymbol) = 10*I2;
            end
            % Burst(I3)
            Nofdmasymbol_I3 = ceil(Burst(I3)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I3 = Nofdmasymbol_I3*(Nsubchannel-M)*DataCarrier -
Burst(I3);
            reste_I3 = floor(reste_Burst_I3/DataCarrier);
            EspaceLibre(I3) = (Nsubchannel-M - reste_I3);
            for k = M+1:1:Nsubchannel
                for m = 4:1:4 + Nofdmasymbol_I3-1
                    DL_subFrame(k,m) = 10*I3;
                end
            end
            for k = M+1:1:M+1+reste_I3
                DL_subFrame(k,4 + Nofdmasymbol_I3) = 10*I3;
            end
            % Burst(I4)
            Nofdmasymbol_I4 = ceil(Burst(I4)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I4 = Nofdmasymbol_I4*(Nsubchannel-M)*DataCarrier -
Burst(I4);
            reste_I4 = floor(reste_Burst_I4/DataCarrier);
            EspaceLibre(I4) = (Nsubchannel-M - reste_I4);
            for k = M+1:1:Nsubchannel
                for m = 4 + Nofdmasymbol_I3+1:1:4 + Nofdmasymbol_I3+1 +
Nofdmasymbol_I4-1
                    DL_subFrame(k,m) = 10*I4;
                end
            end
            for k = M+1:1:M+1+reste_I4
                DL_subFrame(k,4 + Nofdmasymbol_I3+1 + Nofdmasymbol_I4) =
10*I4;
            end
```

```
            % Burst(I5)
            Nofdmasymbol_I5 = ceil(Burst(I5)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I5 = Nofdmasymbol_I5*(Nsubchannel-M)*DataCarrier -
Burst(I5);
            reste_I5 = floor(reste_Burst_I5/DataCarrier);
            EspaceLibre(I5) = (Nsubchannel-M - reste_I5);
            for k = M+1:1:Nsubchannel
                for m = 4 + Nofdmasymbol_I3+Nofdmasymbol_I4+2:1:4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5-1
                    DL_subFrame(k,m) = 10*I5;
                end
            end
            for k = M+1:1:M+1+reste_I5
                DL_subFrame(k,4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5) = 10*I5;
            end
            % Burst(I6)
            Nofdmasymbol_I6 = ceil(Burst(I6)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I6 = Nofdmasymbol_I6*(Nsubchannel-M)*DataCarrier -
Burst(I6);
            reste_I6 = floor(reste_Burst_I6/DataCarrier);
            EspaceLibre(I6) = (Nsubchannel-M - reste_I6);
            for k = M+1:1:Nsubchannel
                for m = 4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+2:1:4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6-1
                    DL_subFrame(k,m) = 10*I6;
                end
            end
            for k = M+1:1:M+1+reste_I6
                DL_subFrame(k,4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6) = 10*I6;
            end
            % Burst(I7)
            Nofdmasymbol_I7 = ceil(Burst(I7)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I7 = Nofdmasymbol_I7*(Nsubchannel-M)*DataCarrier -
Burst(I7);
            reste_I7 = floor(reste_Burst_I7/DataCarrier);
            EspaceLibre(I7) = (Nsubchannel-M - reste_I7);
            for k = M+1:1:Nsubchannel
                for m = 4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+2:1:4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7-1
                    DL_subFrame(k,m) = 10*I7;
                end
            end
            for k = M+1:1:M+1+reste_I7
                DL_subFrame(k,4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7) = 10*I7;
            end
```

```
            % Burst(I8)
            Nofdmasymbol_I8 = ceil(Burst(I8)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I8 = Nofdmasymbol_I8*(Nsubchannel-M)*DataCarrier -
Burst(I8);
            reste_I8 = floor(reste_Burst_I8/DataCarrier);
            EspaceLibre(I8) = (Nsubchannel-M - reste_I8);
            for k = M+1:1:Nsubchannel
                for m = 4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7+2:1:4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7+Nofdmasymbol_I8-1
                    DL_subFrame(k,m) = 10*I8;
                end
            end
            for k = M+1:1:M+1+reste_I8
                DL_subFrame(k,4 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7+Nofdmasymbol_I8) = 10*I8;
            end
        end



    %-------------------- Commencer par Region 2 -------------------------
    else
       M = ceil((Burst(I1) + Burst(I2))/(DataCarrier*(Nofdmasymbol-3)));
% Trouver la largeur requise pour les deux plus gros Dl_Burst dans la
région 2
       p = ceil((Nofdmasymbol-3)*(Burst(I1)/(M*DataCarrier*(Nofdmasymbol-
3))));  %  Trouver La longueur du DL_Burst_1
       q = ceil((Nofdmasymbol-3)*(Burst(I2)/(M*DataCarrier*(Nofdmasymbol-
3))));  % p + q = 1
       if( p + q <= Nofdmasymbol-5 )
           sprintf('Region 2    sans ajout de subch')
            % Burst(I1)  et Burst(I2)
            reste_Burst_I1 = p*M*DataCarrier - Burst(I1);
            reste_Burst_I2 = q*M*DataCarrier - Burst(I2);
            reste_I1 = floor(reste_Burst_I1/DataCarrier);
            reste_I2 = floor(reste_Burst_I2/DataCarrier);
            EspaceLibre(I1) = (M - reste_I1);
            EsapceLibre(I2) = (M - reste_I2)
            for k = Nsubchannel-M:1:Nsubchannel
                for m = 4:1:4 + p-1
                    DL_subFrame(k,m) = 10*I1;
                end
                for m = 4+p+1:1:Nofdmasymbol-1
                    DL_subFrame(k,m) = 10*I2;
                end
            end
            for k = Nsubchannel-M:1:Nsubchannel-M+reste_I1
                DL_subFrame(k,p+4) = 10*I1;
```

```
            end
            for k = Nsubchannel-M:1:Nsubchannel-M+reste_I2
                DL_subFrame(k,Nofdmasymbol) = 10*I2;
            end
            % Burst(I3)
            Nofdmasymbol_I3 = ceil(Burst(I3)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I3 = Nofdmasymbol_I3*(Nsubchannel-M)*DataCarrier -
Burst(I3);
            reste_I3 = floor(reste_Burst_I3/DataCarrier);
            EspaceLibre(3) = (Nsubchannel-M - reste_I3);
            for k = 1:1:Nsubchannel-M
                for m = 6:1:6 + Nofdmasymbol_I3-1
                    DL_subFrame(k,m) = 10*I3;
                end
            end
            for k = 1:1:1+reste_I3
                DL_subFrame(k,6 + Nofdmasymbol_I3) = 10*I3;
            end
            % Burst(I4)
            Nofdmasymbol_I4 = ceil(Burst(I4)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I4 = Nofdmasymbol_I4*(Nsubchannel-M)*DataCarrier -
Burst(I4);
            reste_I4 = floor(reste_Burst_I4/DataCarrier);
            EspaceLibre(I4) = (Nsubchannel-M - reste_I4);
            for k = 1:1:Nsubchannel-M
                for m = 6 + Nofdmasymbol_I3+1:1:6 + Nofdmasymbol_I3+1 +
Nofdmasymbol_I4-1
                    DL_subFrame(k,m) = 10*I4;
                end
            end
            for k = 1:1:1+reste_I4
                DL_subFrame(k,6 + Nofdmasymbol_I3+1 + Nofdmasymbol_I4) =
10*I4;
            end
            % Burst(I5)
            Nofdmasymbol_I5 = ceil(Burst(I5)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I5 = Nofdmasymbol_I5*(Nsubchannel-M)*DataCarrier -
Burst(I5);
            reste_I5 = floor(reste_Burst_I5/DataCarrier);
            EspaceLibre(I5) = (Nsubchannel-M - reste_I5);
            for k = 1:1:Nsubchannel-M
                for m = 6 + Nofdmasymbol_I3+Nofdmasymbol_I4+2:1:6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5-1
                    DL_subFrame(k,m) = 10*I5;
                end
            end
            for k = 1:1:1+reste_I5
                DL_subFrame(k,6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5) = 10*I5;
            end
            % Burst(I6)
```

```
            Nofdmasymbol_I6 = ceil(Burst(I6)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I6 = Nofdmasymbol_I6*(Nsubchannel-M)*DataCarrier -
Burst(I6);
            reste_I6 = floor(reste_Burst_I6/DataCarrier);
            EspaceLibre(I6) = (Nsubchannel-M - reste_I6);
            for k = 1:1:Nsubchannel-M
                for m = 6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+2:1:6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6-1
                    DL_subFrame(k,m) = 10*I6;
                end
            end
            for k = 1:1:1+reste_I6
                DL_subFrame(k,6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6) = 10*I6;
            end
            % Burst(I7)
            Nofdmasymbol_I7 = ceil(Burst(I7)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I7 = Nofdmasymbol_I7*(Nsubchannel-M)*DataCarrier -
Burst(I7);
            reste_I7 = floor(reste_Burst_I7/DataCarrier);
            EspaceLibre(I7) = (Nsubchannel-M - reste_I7);
            for k = 1:1:Nsubchannel-M
                for m = 6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+2:1:6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7-1
                    DL_subFrame(k,m) = 10*I7;
                end
            end
            for k = 1:1:1+reste_I7
                DL_subFrame(k,6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7) = 10*I7;
            end
            % Burst(I8)
            Nofdmasymbol_I8 = ceil(Burst(I8)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I8 = Nofdmasymbol_I8*(Nsubchannel-M)*DataCarrier -
Burst(I8);
            reste_I8 = floor(reste_Burst_I8/DataCarrier);
            EspaceLibre(I8) = (Nsubchannel-M - reste_I8);
            for k = 1:1:Nsubchannel-M
                for m = 6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7+2:1:6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7+Nofdmasymbol_I8-1
                    DL_subFrame(k,m) = 10*I8;
                end
            end
            for k = 1:1:1+reste_I8
```

```
                DL_subFrame(k,6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7+Nofdmasymbol_I8) = 10*I8;
            end
        else
            sprintf('Region 2    avec ajout de subch')
            M=M+1;
            p = ceil((Nofdmasymbol-
3)*(Burst(I1)/(M*DataCarrier*(Nofdmasymbol-3))));  %  Trouver La longueur
du DL_Burst_1
            q = ceil((Nofdmasymbol-
3)*(Burst(I2)/(M*DataCarrier*(Nofdmasymbol-3))));  % p + q = 1
            % Burst(I1)  et Burst(I2)
             reste_Burst_I1 = p*M*DataCarrier - Burst(I1);
             reste_Burst_I2 = q*M*DataCarrier - Burst(I2);
             reste_I1 = floor(reste_Burst_I1/DataCarrier);
             reste_I2 = floor(reste_Burst_I2/DataCarrier);
             EspaceLibre(I1) = (M - reste_I1);
             EsapceLibre(I2) = (M - reste_I2);
             for k = Nsubchannel-M+1:1:Nsubchannel
                 for m = 4:1:4 + p-1
                     DL_subFrame(k,m) = 10*I1;
                 end
                 for m = 4+p+1:1:Nofdmasymbol-1
                     DL_subFrame(k,m) = 10*I2;
                 end
             end
             for k = Nsubchannel-M+1:1:Nsubchannel-M+reste_I1
                 DL_subFrame(k,p+4) = 10*I1;
             end
             for k = Nsubchannel-M:1:Nsubchannel-M+reste_I2
                 DL_subFrame(k,Nofdmasymbol) = 10*I2;
             end
             % Burst(I3)
             Nofdmasymbol_I3 = ceil(Burst(I3)/(DataCarrier*(Nsubchannel-
M)));
             reste_Burst_I3 = Nofdmasymbol_I3*(Nsubchannel-M)*DataCarrier -
Burst(I3);
             reste_I3 = floor(reste_Burst_I3/DataCarrier);
             EspaceLibre(3) = (Nsubchannel-M - reste_I3);
             for k = 1:1:Nsubchannel-M
                 for m = 6:1:6 + Nofdmasymbol_I3-1
                     DL_subFrame(k,m) = 10*I3;
                 end
             end
             for k = 1:1:1+reste_I3
                 DL_subFrame(k,6 + Nofdmasymbol_I3) = 10*I3;
             end
             % Burst(I4)
             Nofdmasymbol_I4 = ceil(Burst(I4)/(DataCarrier*(Nsubchannel-
M)));
             reste_Burst_I4 = Nofdmasymbol_I4*(Nsubchannel-M)*DataCarrier -
Burst(I4);
             reste_I4 = floor(reste_Burst_I4/DataCarrier);
             EspaceLibre(I4) = (Nsubchannel-M - reste_I4);
```

```
            for k = 1:1:Nsubchannel-M
                for m = 6 + Nofdmasymbol_I3+1:1:6 + Nofdmasymbol_I3+1 +
Nofdmasymbol_I4-1
                    DL_subFrame(k,m) = 10*I4;
                end
            end
            for k = 1:1:1+reste_I4
                DL_subFrame(k,6 + Nofdmasymbol_I3+1 + Nofdmasymbol_I4) =
10*I4;
            end
            % Burst(I5)
            Nofdmasymbol_I5 = ceil(Burst(I5)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I5 = Nofdmasymbol_I5*(Nsubchannel-M)*DataCarrier -
Burst(I5);
            reste_I5 = floor(reste_Burst_I5/DataCarrier);
            EspaceLibre(I5) = (Nsubchannel-M - reste_I5);
            for k = 1:1:Nsubchannel-M
                for m = 6 + Nofdmasymbol_I3+Nofdmasymbol_I4+2:1:6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5-1
                    DL_subFrame(k,m) = 10*I5;
                end
            end
            for k = 1:1:1+reste_I5
                DL_subFrame(k,6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5) = 10*I5;
            end
            % Burst(I6)
            Nofdmasymbol_I6 = ceil(Burst(I6)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I6 = Nofdmasymbol_I6*(Nsubchannel-M)*DataCarrier -
Burst(I6);
            reste_I6 = floor(reste_Burst_I6/DataCarrier);
            EspaceLibre(I6) = (Nsubchannel-M - reste_I6);
            for k = 1:1:Nsubchannel-M
                for m = 6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+2:1:6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6-1
                    DL_subFrame(k,m) = 10*I6;
                end
            end
            for k = 1:1:1+reste_I6
                DL_subFrame(k,6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6) = 10*I6;
            end
            % Burst(I7)
            Nofdmasymbol_I7 = ceil(Burst(I7)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I7 = Nofdmasymbol_I7*(Nsubchannel-M)*DataCarrier -
Burst(I7);
            reste_I7 = floor(reste_Burst_I7/DataCarrier);
            EspaceLibre(I7) = (Nsubchannel-M - reste_I7);
            for k = 1:1:Nsubchannel-M
                for m = 6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+2:1:6 +
```

```
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7-1
                    DL_subFrame(k,m) = 10*I7;
                end
            end
            for k = 1:1:1+reste_I7
                DL_subFrame(k,6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7) = 10*I7;
            end
            % Burst(I8)
            Nofdmasymbol_I8 = ceil(Burst(I8)/(DataCarrier*(Nsubchannel-
M)));
            reste_Burst_I8 = Nofdmasymbol_I8*(Nsubchannel-M)*DataCarrier -
Burst(I8);
            reste_I8 = floor(reste_Burst_I8/DataCarrier);
            EspaceLibre(I8) = (Nsubchannel-M - reste_I8);
            for k = 1:1:Nsubchannel-M
                for m = 6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7+2:1:6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7+Nofdmasymbol_I8-1
                    DL_subFrame(k,m) = 10*I8;
                end
            end
            for k = 1:1:1+reste_I8
                DL_subFrame(k,6 +
Nofdmasymbol_I3+Nofdmasymbol_I4+Nofdmasymbol_I5+Nofdmasymbol_I6+Nofdmasymb
ol_I7+Nofdmasymbol_I8) = 10*I8;
            end

        end

    end

    %##################  Fin de l'allocation des DL Burst_i  ##############

     SecondRemplissage = 0;

    while (1)
        lastPacket = lastPacket + 1;;
        Profil = BurstProfil(lastPacket);
        Taille = TaillePaquet(lastPacket);

        if Profil == 1
            if Taille*8/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
Taille*8/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
Taille*8/DataCarrier;
            else
                break;
            end
```

```
        elseif Profil == 2
            if 0.5*Taille*8*(4/3)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
0.5*Taille*8*(4/3)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
0.5*Taille*8*(4/3)/DataCarrier;
            else
               break;
            end
        elseif Profil == 3
            if 0.25*Taille*16/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
0.25*Taille*16/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
0.25*Taille*16/DataCarrier;
            else
               break;
            end
        elseif Profil == 4
            if 0.25*Taille*8*(4/3)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
0.25*Taille*8*(4/3)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
0.25*Taille*8*(4/3)/DataCarrier;
            else
               break;
            end
        elseif Profil == 5
            if 1/6*Taille*8*(3/2)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
1/6*Taille*8*(3/2)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
1/6*Taille*8*(3/2)/DataCarrier;
            else
               break;
            end
        elseif (Profil == 6)&((Nombre_profil == 6)|(Nombre_profil ==
7)|(Nombre_profil == 8))
            if 1/2*Taille*8*(3/2)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
1/2*Taille*8*(3/2)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
1/2*Taille*8*(3/2)/DataCarrier;
            else
               break;
            end
        elseif (Profil == 7)&((Nombre_profil == 7)|(Nombre_profil == 8))
            if 1/4*Taille*8*(4/3)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
1/4*Taille*8*(4/3)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
1/4*Taille*8*(4/3)/DataCarrier;
            else
               break;
            end
```

```
        elseif (Profil == 8)&(Nombre_profil == 8)
            if 1/6*Taille*8*(2)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
1/6*Taille*8*(2)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
1/6*Taille*8*(2)/DataCarrier;
            else
               break;
            end
        end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
temps(iii) = toc;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%#########################################################################
vide = 0;
for i = 1:1:Nsubchannel
    for j = 1:1:Nofdmasymbol
        if DL_subFrame(i,j) == 0
            vide = vide + 1;
        end
    end
end

TauxDeRemplissage = (Nsubchannel*(Nofdmasymbol+UL_OFDMA_Symb) - vide +
SecondRemplissage-
0.5*Nombre_profil)/(Nsubchannel*(Nofdmasymbol+UL_OFDMA_Symb))*100;
sprintf('le taux de remplissage de la trame est : %0.5g
pourcent',TauxDeRemplissage);
Taux(iii)=TauxDeRemplissage;
end
mean(Taux)
iii=1:1:500;
plot(Taux);
mean(temps)
%#########################################################################

Frame = [DL_subFrame,TTG,UL_subFrame,RTG];
```

ANNEXE III

Algorithme pour une trame de 1.25 MHz

```
%#########################################################################
%initialiser tout
clc;
clear all;
%#########################################################################
% PUSC : Nsubchannel = 3; Nused = 85; Data=24
% FUSC : Nsubchannel = 2; Nused = 107; Data=48
    mode = 3; %3=PUSC,  2=FUSC
    Nombre_profil = 8;
    Ratio_DL_UL = 2/3;
    T_frame = 0.005;

    BW = 1250000;
    Nfft = 128;
    Nused = 85;% a changer

    if T_frame == 0.005
        N_OFDMA_Symb = 48; %Trame de 5 ms
    elseif T_frame == 0.01
        N_OFDMA_Symb = 96; %Trame de 10 ms
    elseif T_frame == 0.02
        N_OFDMA_Symb = 192; %Trame de 20 ms
    end

    if mode == 2
        Nsubchannel = 2;%FUSC
        DataCarrier = 48;
    else
        Nsubchannel = 3;%PUSC
        DataCarrier = 24;
    end

    DL_OFDMA_Symb = N_OFDMA_Symb*2/3 - 2;
    UL_OFDMA_Symb = N_OFDMA_Symb - DL_OFDMA_Symb  - 4;

    if Nombre_profil == 5
        a6 = 0;a7 = 0;a8 = 0;
    elseif Nombre_profil == 6
        a6 = 1;a7 = 0;a8 = 0;
    elseif Nombre_profil == 7
        a6 = 1;a7 = 1;a8 = 0;
    elseif Nombre_profil == 8
        a6 = 1;a7 = 0;a8 = 1;
    end


%#########################################################################
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Tableau des CID
for k=1:1:1000
    CID(k)=k;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

for iii = 1:1:500
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Tableau des Burst profil
BurstProfil = randsrc(1000,1,[1 2 3 4 5 6 7 8]);
TaillePaquet = poissrnd(150,1,1000);


%#########################################################################

%Construction de la trame

    %initialisation de la trame (128)
    DL_subFrame = zeros(Nsubchannel,DL_OFDMA_Symb);
    UL_subFrame = zeros(Nsubchannel,UL_OFDMA_Symb);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DL_subframe
    %Preamble
    for k = 1:1:Nsubchannel
        DL_subFrame(k,1) = 0.1*10;
    end

    %FCH
    for k = 1:1:2
        for m = 2:1:3
            DL_subFrame(k,m) = 0.2*10;
        end
    end

    %DL_MAP
    if mode == 2 % FUSC
        for m = 4:1:5
            for k = 1:1:2
                DL_subFrame(k,m) = 0.4*10;
            end
        end
    elseif mode == 3 % PUSC
        for m = 4:1:5
            for k = 1:1:2
                DL_subFrame(k,m) = 0.4*10;
            end
        end
        DL_subFrame(3,2) = 0.4*10;
        DL_subFrame(3,3) = 0.4*10;
    end
```

```
    %UL_MAP

    if mode == 2 % FUSC
        for m = 6:1:7
            for k = 1:1:2
                DL_subFrame(k,m) = 0.6*10;
            end
        end
    elseif mode == 3 % PUSC
        for m = 6:1:7
            for k = 1:1:2
                DL_subFrame(k,m) = 0.6*10;
            end
        end
        DL_subFrame(3,4) = 0.6*10;
        DL_subFrame(3,5) = 0.6*10;
    end

    header = 7;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% UL_subframe
    Total_UL = 0;
    slot_tot = UL_OFDMA_Symb*Nsubchannel;

    % Ranging
    for m = 1:1:2
         UL_subFrame(1,m) = 5;
         Total_UL = Total_UL + 2*DataCarrier;
    end
    if mode == 2
        %Burst1
        for m = 3:1:floor(slot_tot/5)
            UL_subFrame(1,m) = 1*10;
            Total_UL = Total_UL + 2*DataCarrier;
        end

        %Burst2
        for m = floor(slot_tot/5+0.5)+1:1:floor(2*slot_tot/5)
            UL_subFrame(1,m) = 2*10;
            Total_UL = Total_UL + 2*DataCarrier;
        end

        %Burst3
        for m = floor(2*slot_tot/5+0.5)+1:1:UL_OFDMA_Symb
            UL_subFrame(1,m) = 3*10;
            Total_UL = Total_UL + 4*DataCarrier;
        end
        for m = 1:1:floor(0.5*slot_tot/5)
             UL_subFrame(2,m) = 3*10;
             Total_UL = Total_UL + 4*DataCarrier;
        end
```

```
        %Burst4
        for m =
floor(0.5*slot_tot/5+0.5)+1:1:floor(0.5*slot_tot/5+0.5)+1+floor(slot_tot/5
)
            UL_subFrame(2,m) = 4*10;
            Total_UL = Total_UL + 4*DataCarrier;
        end

        %Burst5
        for m =
floor(0.5*slot_tot/5+0.5)+1+floor(slot_tot/5)+1:1:UL_OFDMA_Symb
            UL_subFrame(2,m) = 5*10;
            Total_UL = Total_UL + 6*DataCarrier;
        end

    else
        %Burst1
        for m = 3:1:floor(slot_tot/5)
            UL_subFrame(1,m) = 1*10;
            Total_UL = Total_UL + 2*DataCarrier;
        end

        %Burst2
        for m = floor(slot_tot/5+0.5)+1:1:UL_OFDMA_Symb
            UL_subFrame(1,m) = 2*10;
            Total_UL = Total_UL + 2*DataCarrier;
        end
        for m = 1:1:floor(0.5*slot_tot/5)
            UL_subFrame(2,m) = 2*10;
            Total_UL = Total_UL + 2*DataCarrier;
        end

        %Burst3
        for m =
floor(0.5*slot_tot/5+0.5)+1:1:floor(0.5*slot_tot/5+0.5)+1+floor(slot_tot/5
)
            UL_subFrame(2,m) = 3*10;
            Total_UL = Total_UL + 3*DataCarrier;
        end

        %Burst4
        for m =
floor(0.5*slot_tot/5+0.5)+1+floor(slot_tot/5+0.5)+1:1:UL_OFDMA_Symb
            UL_subFrame(2,m) = 4*10;
            Total_UL = Total_UL + 4*DataCarrier;
        end
        for m = 1:1:floor(0.5*slot_tot/5)
            UL_subFrame(3,m) = 4*10;
            Total_UL = Total_UL + 2*DataCarrier;
        end

        %Burst5
        for m = floor(0.5*slot_tot/5+0.5)+1:1:UL_OFDMA_Symb
```

```
            UL_subFrame(3,m) = 5*10;
            Total_UL = Total_UL + 6*DataCarrier;
        end

    end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % TTG
    for k = 1:1:Nsubchannel
        for m = 1:1:mode
            TTG(k,m) = -1;
        end
    end
% RTG
    for k = 1:1:Nsubchannel
        for m = 1:1:mode
            RTG(k,m) = -2;
        end
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DL_subframe (DL-MAP & DL burst #1 à #Np)

  % Time slot disponibles et les 2 regions
    Nofdmasymbol = DL_OFDMA_Symb;

    %reception des paquets et traitement(mapping)
    Surface_Diponible = Nsubchannel*DataCarrier*Nofdmasymbol -
(header*Nsubchannel*DataCarrier + Nombre_profil*Nsubchannel/2);
    Burst_1 = 0;Burst_2 = 0;Burst_3 = 0;Burst_4 = 0;Burst_5 = 0;Burst_6 =
0;Burst_7 = 0;Burst_8 = 0;
    k=0;
    while Burst_1 + Burst_2 + Burst_3 + Burst_4 + Burst_5 + a6*Burst_6 +
a7*Burst_7 + a8*Burst_8 < Surface_Diponible
        k = k + 1;
        if BurstProfil(k)==1
            Burst_1= Burst_1 + TaillePaquet(k)*8;

        elseif BurstProfil(k)==2
            Burst_2= Burst_2 + 0.5*TaillePaquet(k)*8*4/3;

        elseif BurstProfil(k)==3
            Burst_3= Burst_3 + 0.25*TaillePaquet(k)*8*2;

         elseif BurstProfil(k)==4
            Burst_4= Burst_4 + 0.25*TaillePaquet(k)*8*3/2;

        elseif BurstProfil(k)==5
            Burst_5= Burst_5 + 1/6*TaillePaquet(k)*8*3/2;
```

```
        elseif BurstProfil(k)==6
            Burst_6= Burst_6 + a6*0.5*TaillePaquet(k)*8*3/2;

        elseif BurstProfil(k)==7
            Burst_7= Burst_7 + a7*1/4*TaillePaquet(k)*8*4/3;

        elseif BurstProfil(k)==8
            Burst_8= Burst_8 + a8*1/6*TaillePaquet(k)*8*2;
        end
    end

    if BurstProfil(k)==1
            Burst_1= Burst_1 - TaillePaquet(k)*8;

    elseif BurstProfil(k)==2
            Burst_2= Burst_2 - 0.5*TaillePaquet(k)*8*4/3;

    elseif BurstProfil(k)==3
            Burst_3= Burst_3 - 0.25*TaillePaquet(k)*8*2;

    elseif BurstProfil(k)==4
            Burst_4= Burst_4 - 0.25*TaillePaquet(k)*8*3/2;

    elseif BurstProfil(k)==5
            Burst_5= Burst_5 - 1/6*TaillePaquet(k)*8*3/2;

    elseif BurstProfil(k)==6
            Burst_6= Burst_6 - a6*0.5*TaillePaquet(k)*8*3/2;

    elseif BurstProfil(k)==7
            Burst_7= Burst_7 - a7*1/4*TaillePaquet(k)*8*4/3;

    elseif BurstProfil(k)==8
            Burst_8= Burst_8 - a8*1/6*TaillePaquet(k)*8*2;
    end

    EsapceLibre = [0 0 0 0 0 0 0 0];
    lastPacket = k;

    %###################  Début de l'allocation des DL Burst_i ########

    Burst =
[Burst_1,Burst_2,Burst_3,Burst_4,Burst_5,Burst_6,Burst_7,Burst_8];
    Burst_tmp =
[Burst_1,Burst_2,Burst_3,Burst_4,Burst_5,Burst_6,Burst_7,Burst_8];


    %%%%% l'allocation du DL Burst_1
    L1 = floor(Burst_1/(DataCarrier*Nsubchannel));
    L2 = ceil(Burst_1/(DataCarrier*Nsubchannel));
    for k = 1:1:Nsubchannel
        for m = header+1:1:header+L1
```

```
        DL_subFrame(k,m) = 10;
    end
end
used1 = header+L1;
if L1 ~= L2
   L3 = ceil(Burst_1/(DataCarrier*Nsubchannel) - L1);
   if Burst_1  ~=  0
       for k = 1:1:L3
           DL_subFrame(k,header+L1+1) = 10;
       end
       used1 = header+L1+1;
   end
end
EspaceLibre(1) = floor(L2-Burst_1/(DataCarrier*Nsubchannel));

%%%%% l'allocation du DL Burst_2
L1 = floor(Burst_2/(DataCarrier*Nsubchannel));
L2 = ceil(Burst_2/(DataCarrier*Nsubchannel));
for k = 1:1:Nsubchannel
    for m = used1+1:1:used1+L1
        DL_subFrame(k,m) = 20;
    end
end
    used2 = used1+L1;
    if L1 ~= L2
        L3 = ceil(Burst_2/(DataCarrier*Nsubchannel) - L1);
        if Burst_2  ~=  0
            for k = 1:1:L3
                DL_subFrame(k,used1+L1+1) = 20;
            end
            used2 = used1+L1+1;
        end
    end

EspaceLibre(2) = floor(L2-Burst_2/(DataCarrier*Nsubchannel));

%%%%% l'allocation du DL Burst_3
L1 = floor(Burst_3/(DataCarrier*Nsubchannel));
L2 = ceil(Burst_3/(DataCarrier*Nsubchannel));
for k = 1:1:Nsubchannel
    for m = used2+1:1:used2+L1
        DL_subFrame(k,m) = 30;
    end
end
    used3 = used2+L1;
    if L1 ~= L2
        L3 = ceil(Burst_3/(DataCarrier*Nsubchannel) - L1);
        if Burst_3  ~=  0
            for k = 1:1:L3
                DL_subFrame(k,used2+L1+1) = 30;
            end
            used3 = used2+L1+1;
        end
    end
```

```
    EspaceLibre(3) = floor(L2-Burst_3/(DataCarrier*Nsubchannel));

    %%%%% l'allocation du DL Burst_4
    L1 = floor(Burst_4/(DataCarrier*Nsubchannel));
    L2 = ceil(Burst_4/(DataCarrier*Nsubchannel));
    for k = 1:1:Nsubchannel
        for m = used3+1:1:used3+L1
            DL_subFrame(k,m) = 40;
        end
    end
        used4 = used3+L1;
        if L1 ~= L2
            L3 = ceil(Burst_4/(DataCarrier*Nsubchannel) - L1);
            if Burst_4  ~=  0
                for k = 1:1:L3
                    DL_subFrame(k,used3+L1+1) = 40;
                end
            used4 = used3+L1+1;
            end
        end

    EspaceLibre(4) = floor(L2-Burst_4/(DataCarrier*Nsubchannel));

    %%%%% l'allocation du DL Burst_5
    if (Nombre_profil == 5)|(Nombre_profil == 6)|(Nombre_profil ==
7)|(Nombre_profil == 8)
        L1 = floor(Burst_5/(DataCarrier*Nsubchannel));
        L2 = ceil(Burst_5/(DataCarrier*Nsubchannel));
        for k = 1:1:Nsubchannel
            for m = used4+1:1:used4+L1
                DL_subFrame(k,m) = 50;
            end
        end
            used5 = used4+L1;
            if L1 ~= L2
                L3 = ceil(Burst_5/(DataCarrier*Nsubchannel) - L1);
                if Burst_5  ~=  0
                    for k = 1:1:L3
                        DL_subFrame(k,used4+L1+1) = 50;
                    end
                used5 = used4+L1+1;
                end
            end
        end
        EspaceLibre(5) = floor(L2-Burst_5/(DataCarrier*Nsubchannel));


    %%%%% l'allocation du DL Burst_6 (si il existe)
    if (Nombre_profil == 6)|(Nombre_profil == 7)|(Nombre_profil == 8)
        L1 = floor(Burst_6/(DataCarrier*Nsubchannel));
        L2 = ceil(Burst_6/(DataCarrier*Nsubchannel));
        for k = 1:1:Nsubchannel
            for m = used5+1:1:used5+L1
```

```
            DL_subFrame(k,m) = 60;
        end
    end
        used6 = used5+L1;
        if L1 ~= L2
            L3 = ceil(Burst_6/(DataCarrier*Nsubchannel) - L1);
            if Burst_6  ~=  0
                for k = 1:1:L3
                    DL_subFrame(k,used5+L1+1) = 60;
                end
            used6 = used5+L1+1;
            end
        end
    end
    EspaceLibre(6) = floor(L2-Burst_6/(DataCarrier*Nsubchannel));


%%%%% l'allocation du DL Burst_7 (si il existe)
if (Nombre_profil == 7)|(Nombre_profil == 8)
    L1 = floor(Burst_7/(DataCarrier*Nsubchannel));
    L2 = ceil(Burst_7/(DataCarrier*Nsubchannel));
    for k = 1:1:Nsubchannel
        for m = used6+1:1:used6+L1
            DL_subFrame(k,m) = 70;
        end
    end
        used7 = used6+L1;
        if L1 ~= L2
            L3 = ceil(Burst_7/(DataCarrier*Nsubchannel) - L1);
            if Burst_7  ~=  0
                for k = 1:1:L3
                    DL_subFrame(k,used6+L1+1) = 70;
                end
                used7 = used6+L1+1;
            end
        end
    end
    EspaceLibre(7) = floor(L2-Burst_7/(DataCarrier*Nsubchannel));


%%%%% l'allocation du DL Burst_8 (si il existe)
if Nombre_profil == 8
    L1 = floor(Burst_8/(DataCarrier*Nsubchannel));
    L2 = ceil(Burst_8/(DataCarrier*Nsubchannel));
    for k = 1:1:Nsubchannel
        for m = used7+1:1:used7+L1
            DL_subFrame(k,m) = 80;
        end
    end
        used8 = used7+L1;
        if L1 ~= L2
            L3 = ceil(Burst_8/(DataCarrier*Nsubchannel) - L1);
            if Burst_8  ~=  0
                for k = 1:1:L3
```

```
                        DL_subFrame(k,used7+L1+1) = 80;
                    end
                    used8 = used7+L1+1;
                end
            end
        end
        EspaceLibre(8) = floor(L2-Burst_8/(DataCarrier*Nsubchannel));

    %###################  Fin de l'allocation des DL Burst_i  ########

     SecondRemplissage = 0;

    while (1)
        lastPacket = lastPacket + 1;;
        Profil = BurstProfil(lastPacket);
        Taille = TaillePaquet(lastPacket);

        if Profil == 1
            if Taille*8/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
Taille*8/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
Taille*8/DataCarrier;
            else
                break;
            end
         elseif Profil == 2
             if 0.5*Taille*8*(4/3)/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
0.5*Taille*8*(4/3)/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
0.5*Taille*8*(4/3)/DataCarrier;
             else
                break;
            end
         elseif Profil == 3
             if 0.25*Taille*16/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
0.25*Taille*16/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
0.25*Taille*16/DataCarrier;
             else
                break;
             end
         elseif Profil == 4
             if 0.25*Taille*8*(4/3)/DataCarrier < EspaceLibre(Profil)
                SecondRemplissage = SecondRemplissage +
0.25*Taille*8*(4/3)/DataCarrier;
                EspaceLibre(Profil) = EspaceLibre(Profil) -
0.25*Taille*8*(4/3)/DataCarrier;
             else
                break;
            end
         elseif Profil == 5
```

```
            if 1/6*Taille*8*(3/2)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
1/6*Taille*8*(3/2)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
1/6*Taille*8*(3/2)/DataCarrier;
            else
               break;
            end
         elseif (Profil == 6)&((Nombre_profil == 6)|(Nombre_profil ==
7)|(Nombre_profil == 8))
            if 1/2*Taille*8*(3/2)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
1/2*Taille*8*(3/2)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
1/2*Taille*8*(3/2)/DataCarrier;
            else
               break;
           end
         elseif (Profil == 7)&((Nombre_profil == 7)|(Nombre_profil == 8))
            if 1/4*Taille*8*(4/3)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
1/4*Taille*8*(4/3)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
1/4*Taille*8*(4/3)/DataCarrier;
            else
               break;
            end
         elseif (Profil == 8)&(Nombre_profil == 8)
            if 1/6*Taille*8*(2)/DataCarrier < EspaceLibre(Profil)
               SecondRemplissage = SecondRemplissage +
1/6*Taille*8*(2)/DataCarrier;
               EspaceLibre(Profil) = EspaceLibre(Profil) -
1/6*Taille*8*(2)/DataCarrier;
            else
               break;
            end

         end
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
temps(iii) = toc;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%#########################################################################
vide = 0;
for i = 1:1:Nsubchannel
    for j = 1:1:Nofdmasymbol
        if DL_subFrame(i,j) == 0
            vide = vide + 1;
        end
    end
end
```

```
TauxDeRemplissage = (Nsubchannel*(Nofdmasymbol + 12) - vide +
SecondRemplissage)/(Nsubchannel*(Nofdmasymbol + 12))*100;

Taux(iii)=TauxDeRemplissage;
end
sprintf('le taux de remplissage de la trame est : %0.5g
pourcent',mean(Taux))
mean(Taux)
iii=1:1:500;
plot(Taux);
mean(temps)
%##########################################################################
```

BIBLIOGRAPHIE

[1] ''air interface for fixed Broadband Wireless Access Systems'', IEEE STD 802.16 – 2004, Octobre, 2004.

[2] (2006). IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1. IEEE Std 802.16e-2005 and IEEE Std 802.16-2004/Cor 1-2005 (Amendment and Corrigendum to IEEE Std 802.16-2004): 0_1-822.

[3] ''Mobile WiMAX – Part I: A Technical Overview and Performance Evaluation'', WiMAX FORUM, Mars, 2006.

[4] ''Mobile WiMAX – Part II: A comparative Analysis'', WiMAX FORUM, Mars, 2006.

[5] ''Can WiMAX Address Your Applications? '', WiMAX FORUM, Octobre, 2005.

[6] L.J. Cimini, "Analysis and Simulation of a Digital Mobile Channel Using Orthogonal Frequency Division Multiplexing," IEEE Trans. Comm., vol. COM-33, no. 7, pp 665-675, Juin 1985.

[7] Richard Van Nee and Ramjee Prasad, "OFDM for Wireless Multimedia Communications," Artech House, 2000.

[8] Hassan Yagoobi, "Scalable OFDMA Physical Layer in IEEE 802.16 WirelessMAN", Intel Technology Journal, Vol 08, Août 2004.

[9] G. Nair, J. Chou, T. Madejski, K. Perycz, P. Putzolu and J. Sydir "IEEE 802.16 Medium Access Control and Service Provisioning", Intel Technology Journal, vol 08, Août 2004.

[10] F. Wang, A. Ghosh, R. Love, K. Stewart et.al., "IEEE 802.16e System Performance-Analysis and Simulation Results", Proc. of PIMRC, Berlin, Germany, Sept. 2005.

[11] "OFDMA PHY SAP Interface Specification for 802.16 Broadband Wireless Access Base Stations", Intel Technology Journal, Mai 2006.

[12] Kuo-Hui Li, "IEEE 802.16e-2005 Air Interface Overview", WiMAX Solutions Division, Intel Mobility Group, June 05, 2006

[13] David CARSENAT, "CONTRIBUTION A L'ÉTUDE DE RÉSEAUX DE COMMUNICATION SANS FIL. APPLICATION AU LMDS.", UNIVERSITÉ DE LIMOGES, ÉCOLE DOCTORALE, Science – Technologie – Santé, FACULTÉ DES SCIENCES ET TECHNIQUES, 15 Octobre 2003

[14] Kwon, T., H. Lee, et al. (2005). "Design and implementation of a simulator based on a cross-layer protocol between MAC and PHY layers in a WiBro Compatible.IEEE 802.16e OFDMA system." Communications Magazine, IEEE 43(12): 136-146.

[15] Everitt, D. E. (1994). "Traffic engineering of the radio interface for cellular mobile networks." Proceedings of the IEEE 82(9): 1371-1382.

[16] Fang, Y. (2002). "General modeling and performance analysis for location management in wireless mobile networks." Computers, IEEE Transactions on 51(10): 1169-1181.