VNIVERSIDAD NACIONAL AVFNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

PROGRAMA DE MAESTRIA Y DOCTORADO EN INGENIERIA
FACULTAD DE INGENIERÍA

IMPLEMENTACIÓN Y ANÁLISIS DE UN ALGORITMO DISTRIBUIDO DE CALENDARIZACIÓN DE RECURSOS BASADOS EN EL PROTOCOLO WiMAX-MESH

TE S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERIA

INGENIERÍA ELÉCTRICA – TELECOMUNICACIONES

P R E S E N T A :

ING. LISANDRA MAGNOLIA JARQUIN RAMOS

TUTOR:

DR. VICTOR RANGEL LICEA

AÑO

2012

JURADO ASIGNADO:

Presidente:	DR. GUTIÉRREZ CASTREJÓN RAMÓN
Secretario:	DR. BENITEZ PEREZ HECTOR
Vocal:	DR. RANGEL LICEA VICTOR
1er. Suplente:	DR. MARTÍNEZ LÓPEZ JOSÉ ISMAEL
2do. Suplente:	DR. GARCÍA GARDUÑO VÍCTOR

Lugar donde se realizó la tesis: FACULTAD DE INGENIERÍA.

TUTOR DE TESIS:
DR. RANGEL LICEA VICTOR

FIRMA

Agradecimientos

A Dios
Por guiar siempre mi camino y haber hecho realidad este hermoso sueño.

A mi mamá, Sra. Teresa Ramos Martínez (†)
Por haberme dado la vida, porque siempre estuviste a mi lado incondicionalmente, y por haberme enseñado el valor de la responsabilidad. Siempre estás en mis pensamientos y en mi corazón, este trabajo está dedicado a ti, eres mi inspiración.

A mi hermano, Cristian Ivan Jarquin Ramos
Por estar a mi lado, apoyándome en todo momento. Este triunfo también es tuyo.

A mis amigos Victoria Vega, Verónica Bonilla, Paul Domínguez, José Antonio Ramírez, Manuel Ortíz y Juan Carlos Beltrán
Por brindarme su amistad.

"Haz todo lo necesario para
lograr tu más ardiente deseo y
acabarás lográndolo "

Beethoven

Agradecimientos

A la Universidad Nacional Autónoma de México y a la Facultad de Ingeniería
Eres mi Alma Mater, te agradezco haberme dado la oportunidad de realizar mis estudios en la Facultad de Ingeniería primero en la licenciatura como Ingeniera y ahora como Maestra en Ingeniería, es un orgullo formar parte de esta máxima casa de estudios.

A mi director de tesis, Dr. Victor Rangel Licea
Le agradezco muy sinceramente por el apoyo y la paciencia que me brindó, ya que su contribución fue primordial para la realización de este trabajo. Fue un privilegio haber formado parte de sus tesistas.

A M.I Yasmin Macedo Reza
Por todo su apoyo y asesoría para el desarrollo de este trabajo.

A la CEP-UNAM
Por la beca que me otorgaron para realizar mis estudios de maestría.

A la DGAPA-UNAM
Por el apoyo recibido por medio del proyecto PAPIIT No. IN108910, Diseño de Algoritmos de reservación de capa cruzada en redes móviles y mesh de banda ancha.

Al CONACYT
Por el apoyo que recibí con el proyecto CONACYT 105279, Diseño de técnicas de reservación de capacidad de redes BWA móviles.

Lisandra Magnolia Jarquin Ramos

Resumen

En este trabajo de investigación se presenta un modelo que realiza la reservación de recursos, basándose en una red WiMAX (Worldwide Interoperability for Microwave Access) [1] con topología *mesh*, que trabaja en modo distribuido coordinado.

En las redes *mesh*, cada estación suscriptora (SS) es la responsable de enviar sus solicitudes de ancho de banda, para poder transmitir sus paquetes de datos, así como asignarlo cuando se lo soliciten. La solicitud de recursos es mediante el proceso de tres vías, es decir, la SS enviará una solicitud *(REQUEST)* a la estación con la que se quiere comunicar; ésta recibe la solicitud y le asignará en qué momento puede trasmitirle la información, y se lo dará a conocer mediante un mensaje de concesión *(GRANT)*. Finalmente la estación transmisora le confirmará al receptor con un mensaje de recibido *(ASK),* que ya conoce los recursos que le fueron asignados, para posteriormente enviar la información.

Existen varios algoritmos para asignar recursos, este modelo se basa mediante la técnica de calendarización FIFO (First In, First Out), es decir que se asignan recursos a las solicitudes de los nodos como van llegando (primero que llega, primero que sale).

Este algoritmo de calendarización de recursos se desarrolló en el software llamado OPNET Modeler v.8. Se implementó en una red *mesh* que consta de seis SS´s y una estación base (BS).

Los resultados obtenidos con el modelo que se implementó en OPNET se compararon con el modelo teórico que se desarrolló, verificando que su funcionamiento fuera el correcto. También se demuestra que si es posible obtener una alta tasa de transmisión de datos, sin embargo, a medida que el nodo destino se encuentre a más de un salto de distancia del nodo transmisor, esta tasa de transmisión disminuirá considerablemente.

1. Todas las abreviaciones se encuentran definidas en la lista de acrónimos.

Índice General

Agradecimientos 3

Resumen 5

Capítulo 1. 12
Introducción 12

1.1 Definición del problema 12

1.2 Antecedentes 13

1.3 Objetivos y contribuciones 14

1.4 Estructura de la tesis 15

Capítulo 2. 16
Redes de banda ancha 16

2.1 Introducción 16

2.2 Sistemas de banda ancha 16

2.2.1 Aplicaciones de BWA 17

2.3 Desarrollo de las redes inalámbricas de banda ancha 18

2.3.1. Distribución de Video: LMDS, MMDS y DVB 18

2.4 Sistema WiMAX 19

2.4.1 Sistema Pre-Wimax 19

2.4.2 Sistema Wimax 19

2.5 Estándar IEEE 802.16 20

2.6 Estado del Arte 21

Capítulo 3 24
Descripción del protocolo IEEE 802.16 24

3.1 Introducción 24

3.2 Modos de operación para redes *mesh* 24

3.4 Características del *frame* para el modo de operación *mesh* 26

3.4.1 *Subframe* de Control de la Red 27

3.4.2 *Subframe* de Control para la calendarización 28

3.5 Mecanismos para envío de mensajes 29

3.5.1 Calendarización distribuida 29

3.5.2 Mensajes MSH-DSCH 30

3.5.2.1 MSH-DSCH Scheduling IE 32

3.5.2.2 *MSH-DSCH Request IE* 33

3.5.2.3 MSH-DSCH Availabilities IE 35

3.5.2.4 *MSH-DSCH Grants IE* 36

Capítulo 4. 38
Algoritmo para la reservación de recursos 38

4.1 Introducción 38

4.2 Diseño del algoritmo de reservación de recursos 39

4.3 Descripción de las variables utilizadas en el algoritmo 45

4.4 Descripción de los parámetros a utilizar 47

4.5 Implementación del algoritmo 48

4.6 Características del canal 49

4.7 Análisis del algoritmo 52

4.8 Resultados 54

Capítulo 5. 55
Comportamiento dinámico de las redes WiMAX en modo *mesh* 55

5.1 Introducción 55

5.2 Análisis del algoritmo implementado 55

Capítulo 6. 75
Conclusiones 75

Bibliografía 77

Apéndice 79

Apéndice A. Lista de acrónimos 79

Apéndice B. Diagrama de envío de paquetes 80

Apéndice C. Programa para encontrar la Oportunidad de Transmisión de cada nodo... 85

Apéndice D . Programa para determinar el tiempo de transmisión de los paquetes de datos de cada nodo..... 92

Figura 1.1: Red inalámbrica tipo *mesh.*..... 13

Figura 2.1: Aplicación del Acceso Inalámbrico de Banda Ancha (BWA) con un acceso fijo. 18

Figura 3.1: Características del *frame* para el modo de operación *mesh.* 26

Figura 3.2: *Subframe* de Control del *frame mesh.* 28

Figura 3.3: Las tres partes del *Subframe* de Control para la calendarización en el *subframe mesh.* 29

Figura 4.1: Modelo de Red y Modelo de Nodo. 39

Figura 4.2: Modelos de Procesos para la transmisión de tráfico, implementado en OPNET Modeler v.8. 40

Figura 4.3: Modelo de Procesos implementado en OPNET Modeler para la capa MAC (*mesh_mac_ss)*. 40

Figura 4.4: Encapsulamiento del mensaje a transmitir. 42

Figura 4.5: División del *subframe* de control y del *subframe* de datos en símbolos. 51

Figura 4.6: Número de *minislots* que corresponden al *subframe* de datos. 51

Figura 4.7: Oportunidades de transmisión para un Xmt Holdoff exponent de 0 y mx de 1. 53

Figura 4.8: Resultados obtenidos mediante OPNET Modeler v.8. 54

Figura 5.1: Modelo de Red con 6 estaciones suscriptoras. 56

Figura 5.2: Un usuario transmitiendo datos a un salto 57

Figura 5.3: Utilización del sistema con un usuario transmitiendo a un salto. 58

Figura 5.4: Retardo del sistema con un usuario transmitiendo a un salto. 59

Figura 5.5: *Throughput* del sistema con un usuario transmitiendo a un salto 60

Figura 5.6: Proceso de calendarización para una carga ofrecida de 19Mbps. 62

Figura 5.7: Un usuario transmitiendo datos a dos saltos. 64

Figura 5.8: Utilización del sistema con un usuario transmitiendo a dos saltos. 65

Figura 5.9: Retardo del sistema con un usuario transmitiendo a dos saltos 66

Figura 5.10: *Throughput* del sistema con un usuario transmitiendo a dos saltos. 67

Figura 5.11: Un usuario transmitiendo datos a tres saltos 68

Figura 5.12: Utilización del sistema con un usuario transmitiendo a tres saltos. 69

Figura 5.13: Retardo del sistema con un usuario transmitiendo a tres saltos. 69

Figura 5.14: *Throughput* del sistema con un usuario transmitiendo a tres saltos. 70

Figura 5.15: Los seis usuarios transmitiendo tráfico al mismo tiempo. 71

Figura 5.16: Utilización del sistema con todos los usuarios transmitiendo. 72

Figura 5.17: Retardo del sistema con todos los usuarios transmitiendo. 73

Figura 5.18: *Throughput* del sistema con todos los usuarios transmitiendo. 74

Tabla 3.1: Formato de mensajes MSH-DSCH. 31

Tabla 3.2: MSH-DSCH Scheduling IE. 32

Tabla 3.3: MSH-DSCH Request IE. 34

Tabla 3.4: *MSH-DSCH Availabilities IE.* 35

Tabla 3.5: *MSH-DSCH Grants IE.* 36

Tabla 4.1: Máxima capacidad del canal 52

Tabla 4.2: Parametros Xmt Holdoff exponent y mx. 52

Tabla: 4.3 Tiempo correspondiente a la oportunidad de transmisión. 53

Tabla 5.1: Parámetros de las estaciones suscriptoras 56

Tabla 5.2: Direccionamiento de tráfico. 72

Capítulo 1.

Introducción

1.1 Definición del problema

En la actualidad, las redes de comunicaciones son indispensables para la vida diaria de las personas, sin embargo hay lugares donde no es posible utilizar una red cableada, es aquí donde se emplean las redes inalámbricas, ya que con ellas también podemos enviar archivos, mandar correos y visualizar páginas *web*.

Pero a medida que se utiliza este tipo de comunicación, se demandan mejores servicios, por tal motivo, es necesario contar con la tecnología que pueda brindarnos lo que necesitamos. Un sistema de comunicación inalámbrica que puede conectarse a redes de datos de alta velocidad como el Internet es la tecnología WiMAX. Este tipo de red se utiliza debido a su bajo costo y alta confiabilidad. Se puede implementar en edificios, a través de un campus o en el área metropolitana.

El sistema WiMAX puede operar como PMP (Point to Multipoint) o en modo *mesh* [1], que es el modo de operación en el que nos enfocamos en este trabajo.

En una red *mesh*, cada nodo es un AP (Access Point) y también un router, ya que ahí se crean múltiples rutas para la señal inalámbrica, tal como se muestra en la figura 1.1. Eso significa que cada SS puede comunicarse con múltiples SSs, formando un arreglo tipo malla, lo cual les permite tener múltiples rutas para que la información llegue al destino, sin la necesidad de estar comunicados directamente con la estación base.

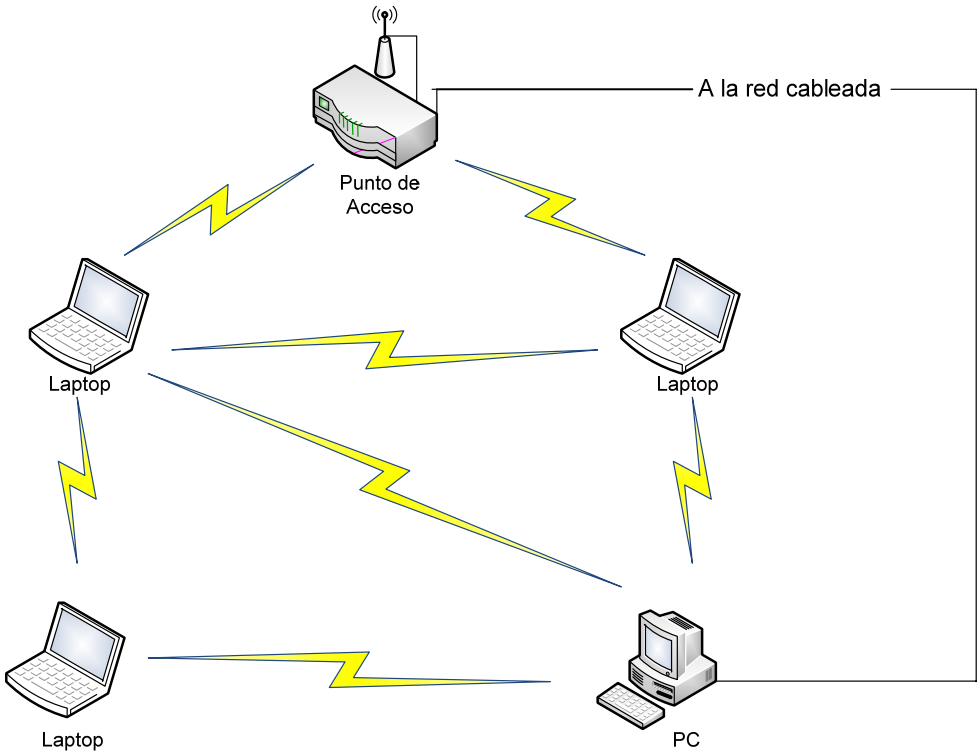


Figura 1.1: Red inalámbrica tipo *mesh.*

La forma de operación del modo *mesh* puede ser centralizada o distribuida, la diferencia entre la forma distribuida con respecto a la centralizada, es que ésta no requiere de un control central por medio de la BS [2]. La ventaja de utilizar el modo *mesh* en forma distribuida, es permitir que las tasas de transmisión de los datos sean altas, ya que permite ampliar el rango de cobertura, sin embargo, al aumentar el rango también aumenta el número de usuarios que desean acceder a la red, por tal motivo se empieza a tener problemas con el rendimiento, por lo tanto es necesario contar con un algoritmo, que permita calendarizar los recursos de la red de manera justa.

1.2 Antecedentes

La evolución de las redes de datos ha marcado la pauta para que las redes inalámbricas tomen un papel importante en las comunicaciones.

A finales del siglo IX, Maxwell mostró por medio de sus ecuaciones, que la transmisión de datos podía realizarse sin necesidad de cables. Años más tarde Marconi demostró que la transmisión inalámbrica era posible para distancias muy largas.

Desde entonces se han creado tecnologías que han sido utilizadas, tal como GSM (Global System for Mobile Communications), que es conocida como la segunda generación del sistema celular. Posteriormente vienen las tecnologías de tercera generación 3G, como UMTS (Universal Mobile Telecommunication System. Esta evolución ha provocado que las personas que utilizan este tipo de tecnología requieran tasas de transmisión más altas, áreas de cobertura más amplias, dando pauta para que las redes inalámbricas de banda ancha tomen importancia en la actualidad.

1.3 Objetivos y contribuciones

El objetivo de este trabajo, es desarrollar un mecanismo de reservación de recursos en forma distribuida para redes WiMAX *mesh*, basándose en el sistema coordinado y en la versión del estándar IEEE 802.16-2004 [3], posteriormente se implementará en el software OPNET Modeler v.8, ya que éste no cuenta con ningún algoritmo de calendarización para este tipo de redes.

Para lograr el objetivo, primero se analizará el proceso de tres vías, ya que las redes WiMAX *mesh* utilizan este procedimiento para solicitar y conceder recursos. Se diseñará e implementará este proceso tomando en cuenta el comportamiento real de una red, donde los usuarios solicitarán ancho de banda en cualquier momento, y para cualquier destino. Finalmente se diseñará un algoritmo para asignar los recursos, considerando que las solicitudes de las diferentes fuentes se almacenarán en una cola del tipo FIFO, es decir, que se le otorgarán los recursos a las solicitudes conforme vayan llegando.

Este trabajo de tesis contribuye con un algoritmo de calendarización de recursos, que toma en cuenta tanto el proceso de tres vías, como la asignación de recursos, con el fin de modelar de una manera más real el comportamiento de estas redes. Para comprobar el funcionamiento del diseño, éste se implementó en una red de seis SS´s y una BS. La ventaja de este modelo es que se puede extender a una red con mayor número de nodos. Además de que sirve como base para proponer nuevas técnicas para mejorar el rendimiento de las redes *mesh* tanto para el envío de datos como para el proceso de tres vías.

Así mismo, se llevó a cabo un análisis del comportamiento dinámico de dicho mecanismo, obteniendo la máxima capacidad que tiene un sistema de 25 Mhz, y de acuerdo a los resultados se sabe que las aplicaciones que se pueden utilizar son las que no son sensibles al retardo, como el envío del correo electrónico, la trasferencia de archivos y acceso a la *web*.

1.4 Estructura de la tesis

A continuación se presenta la estructura que sigue esta tesis:

Capítulo 2. Redes de banda ancha: Este capítulo se dedica para dar a conocer las tecnologías de Banda Ancha, así como de su principal aplicación. Se describe el sistema WiMAX y el estándar 802.16. Finalmente se realiza el estado del arte donde se mencionan las investigaciones más recientes para la calendarización de recursos en redes *mesh*.

Capítulo 3. Descripción del protocolo IEEE 802.16: En este capítulo se aborda el estándar 802.16, en base a la topología *mesh,* se describen sus principales características, la forma de funcionamiento, los modos de operación para las redes *mesh*, así como la descripción del *frame* para este tipo de redes. Por último se revisa el mecanismo de envío de mensajes para el proceso de tres vías.

Capítulo 4. Algoritmo para la reservación de recursos: Se muestra el modelo de reservación de recursos que se diseñó, así como la explicación de las variables y parámetros utilizados.

Capítulo 5. Comportamiento dinámico de las redes WiMAX en modo *mesh*: Se realiza un análisis de los resultados obtenidos con el algoritmo implementado.

Capítulo 6. Conclusiones: Se resaltan los aspectos más importantes de este trabajo, haciendo énfasis en las contribuciones que se lograron y en los trabajos futuros que se pueden realizar en base a estas contribuciones.

Capítulo 2.

Redes de banda ancha

2.1 Introducción

La finalidad de este capítulo es mostrar la importancia que tienen las redes inalámbricas de banda ancha en la actualidad, debido a las altas tasas de transmisión que pueden lograr sobre una amplia área de cobertura.

Se empieza describiendo las tecnologías y sus aplicaciones que forman parte de este tipo de redes. Se hace énfasis en WiMAX que es la tecnología en la que se enfoca este trabajo y finalmente se desarrolla el estado del arte, donde se muestran las investigaciones relacionadas con la calendarización de recursos para redes WiMAX *mesh*.

2.2 Sistemas de banda ancha

Los sistemas inalámbricos han ido evolucionando a través del tiempo, ya que se han creado diferentes tecnologías que van cubriendo las necesidades del ser humano.

Una tecnología es GSM (Global System for Mobile Communications), ya que principalmente es utilizada para la transmisión de voz, que también cuenta con una transmisión de datos a baja velocidad tal como SMS (Short Message Service) [4].

Por otro lado las tecnologías UMTS (Universal Mobile Telecommunication System) y CDMA 2000 (Code Division Multiple Access), son consideradas como sistemas de tercera Generación 3G [5].

Existen grupos de trabajo, que se encargan de estudiar las características y el funcionamiento de los sistemas de banda ancha, a continuación se presentan algunos, dejando pendiente a IEEE 802.16, ya que se describirá más adelante.

- IEEE 802.20, MBWA (Mobile Broadband Wireless Access). El objetivo de este grupo es definir una tecnología para una interfaz aérea basada en servicios sobre IP (Internet Protocol) [6].

- IEEE 802.21, MIH (Media Independent Handover). Es un nuevo estándar y su objetivo es que se pueda realizar la transferencia de datos entre dos tecnologías inalámbricas diferentes [7].

2.2.1 Aplicaciones de BWA

Las redes de telefonía celular de tercera generación 3G y 2.5 G, las cuales tienen como servicio la entrega de paquetes de datos a mediana velocidad, han sido la competencia directa de los servicios inalámbricos de banda ancha, ya que la telefonía celular y BWA comparten un enlace aéreo y en algunos casos tecnología básica.

La primera aplicación de BWA es fijar posiciones de acceso de alta velocidad de datos. Este acceso puede ser utilizado para Internet, TV y otras aplicaciones de alta velocidad. El principal objetivo de BWA es ser un DSL inalámbrico (Digital Suscriber Line).

El término de banda ancha inalámbrica generalmente se refiere a la transmisión de datos a alta velocidad que se realiza dentro de una interface aérea y en una infraestructura fija, por medio de las SS´s y la BS. Este es distinto de la transmisión de datos móviles donde los suscriptores pueden acceder a la red mientras están en movimiento y solo la BS permanece fija.

Otra posible aplicación de acceso de alta velocidad con BWA se muestra en la figura 2.1, donde el Internet también llamado backbone está enlazado a una BS, y esta a su vez puede estar en línea de vista con otra BS. La segunda BS tiene comunicación con las SS´s sin tener línea de vista.

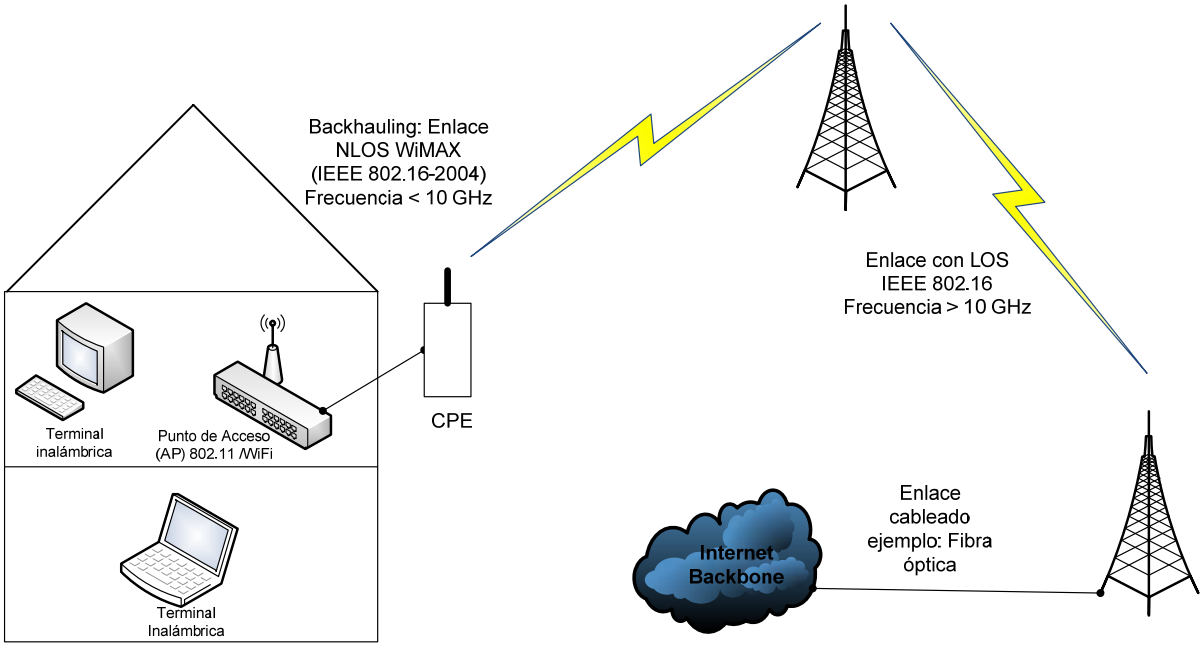


Figura 2.1: Aplicación del Acceso Inalámbrico de Banda Ancha (BWA) con un acceso fijo.

La suscriptora que se muestra en la figura 2.1 es un CPE (Customer Premises Equipment). El CPE es un equipo que detecta la relación que hay entre la BS y el equipo terminal del usuario.

Después del CPE, el usuario puede instalar una terminal como una computadora personal (PC) o una TV y también puede conectar un punto de Acceso WiFi para instalar una WLAN.

Entonces las dos principales aplicaciones de BWA fijo son la red inalámbrica de última milla para tasas de transmisión altas y el WiFi. [4]

2.3 Desarrollo de las redes inalámbricas de banda ancha

2.3.1. Distribución de Video: LMDS, MMDS y DVB

El LMDS (Local Multipoint Distribution Service) es un sistema de acceso inalámbrico fijo especificado por Davic (Digital Audio Video Council), un consorcio de proveedores de equipo de video, operadores de redes y otras industrias de telecomunicaciones. Esta tecnología de comunicaciones de banda ancha punto-multipunto opera a los 28 GHz.

MMDS (Multichannel Multipoint Distribution Service) es una tecnología BWA, que principalmente se usa como un método alternativo de la televisión por cable. El MMDS opera en frecuencias menores a LMDS en 2.5 GHz y 2.7 GHz con velocidades de datos más bajas [4].

DVB (Digital Video Broadcasting). Se encuentra normalizada por ETSI (European Telecommunications Standards Institute). Las redes que están normalizadas son: por cable DVB-C, por satélite DBV y las de televisión terrestre DVB-T [8].

2.4 Sistema WiMAX

2.4.1 Sistema Pre-Wimax

La primera versión del estándar IEEE 802.16 apareció en el 2001 y la versión completa fue publicada hasta 2004. Pero antes de esta fecha ya existían compañías que fabricaban con su propia tecnología productos de banda ancha.

Con la llegada del estándar 802.16 muchos de estos productos aseguraban basarse en el estándar, sin embargo esto no sirvió como prueba para verificar la interoperabilidad entre ellos. Esos productos se conocían entonces como productos pre-WiMAX.

Una lista de fabricantes de equipos pre-WiMAX son: Airspan, Alvarion, Aperto, Motorola, Navini, NexNet, Proxim, Redline y SR Telecom. Intel y Sequans son proveedores de componentes [4].

2.4.2 Sistema Wimax

WiMAX es un sistema de comunicación inalámbrica que permite conectarse a redes de datos de alta velocidad como el Internet, utilizando ondas de radio como medio de transmisión a velocidades que pueden exceder los 120 Mbps para cada canal de radio y está definido en IEEE 802.16.

WiMAX es un sistema que es utilizado en redes inalámbricas de área metropolitana WMAN. Éstas son utilizadas en todo el mundo y sus aplicaciones incluyen servicios de Internet de banda ancha inalámbrico al consumidor, líneas interconectadas y el servicio de

televisión digital. WiMAX puede competir con DSL, cable modem y conexiones ópticas de banda ancha.

También existe el Foro WiMAX que es una industria o corporación no lucrativa, formada para promover y certificar la compatibilidad e interoperabilidad de los productos inalámbricos de banda ancha 802.16. Los primeros productos que utilizaron esta tecnología están basados en la versión del estándar 2004 llamada 802.16-2004.

2.5 Estándar IEEE 802.16

El comité de estándar IEEE 802 LAN/MAN desarrolló los estándares para redes cableadas y para redes inalámbricas. En este trabajo nos enfocamos en el estándar IEEE 802.16 que es un estándar para redes inalámbricas de área metropolitana (WMAN) y que es mejor conocido como WiMAX.

La primera versión de esta tecnología fue desarrollada para el acceso de banda ancha inalámbrica fija, que trabaja en las bandas de 10-66 GHz con línea de vista, esto fue en el año 2001. La siguiente generación se llamó IEEE 802.16a, la cual ya se comunicaba sin línea de vista en las bandas de 2-11 GHz. Las investigaciones más recientes se encuentran en el estándar 802.16e, y están basadas en WiMAX Móvil.

El estándar IEEE especifica la capa física (PHY) y la capa de acceso al medio, MAC (Media Access Control). El estándar 802.16 define cuatro capas físicas diferentes, de las cuales sólo dos están certificadas por el Foro WiMAX y son:

- OFDM (Orthogonal Frecuency Division Multiplexing): Basado en un FFT (Fast Furier Transform) de tamaño de 256 y dirigido a redes fijas.

- OFDM (escalable): Basado en un FFT desde 128 hasta 2048, para redes móviles basadas en 802.16e.

También la capa física puede trabajar con las siguientes características:

- Operación TDD (Time Division Duplex), FDD (Frecuency Division Duplex) y FDD half duplex (H-FDD).
- Acceso TDM (Time Division Multilexing) con el tamaño del *frame* variable (2-20 ms).

- Un gran rango de ancho de banda soportado (1.25-28 MHz).
- Modulación múltiple y esquemas de codificación: QPSK (Quadrature Phase Shift Keying), 16QAM (Quadrature Amplitud Modulation), y 64QAM.
- Sistema de antena adaptiva (AAS) y MIMO (Multiple Input Multiple Output).

2.6 Estado del Arte

La calendarización de recursos en la redes WiMAX *mesh,* es un tema fundamental, ya que afecta de manera significativa el rendimiento de este tipo de redes. Debido a su complejidad, las investigaciones que se realizan, se enfocan en un tema específico, como es la asignación de recursos para transmitir datos, por medio de la calendarización centralizada, o por medio de la calendarización distribuida, ya sea de manera coordinada o no coordinada para ésta última. Otro factor de análisis es el proceso de tres vías que se utiliza para transmitir los mensajes de solicitud, concesión y confirmación de recursos.

En [9] proponen un algoritmo centralizado para asignar recursos de la red, en base a la demanda del tráfico, dándole preferencia a las estaciones suscriptoras que tengan baja demanda; por otro lado en [10] sugieren un algoritmo llamado HRF (*Highest Response First),* donde no solo toman en cuenta el nivel de demanda del tráfico, sino que incluyen un factor de prioridad, que depende del tiempo de espera que los nodos con alta demanda de tráfico sufren, debido a la transmisión de datos que los nodos con baja demanda realizan. Este factor entre mayor sea hará que los nodos con mayor demanda puedan transmitir antes de otros nodos con baja demanda, y así equilibrar las transmisiones de todos los nodos.

Otro tema de análisis es la influencia que tiene el parámetro *Xmt Holdoff Exponent* sobre la eficiencia de la red. Este parámetro es utilizado en el proceso de tres vías, para calcular las oportunidades de transmisión que los nodos emplean, para enviar sus mensajes de control (solicitud, concesión, confirmación). En [11] plantean un mecanismo dinámico que permite utilizar el menor valor de *Xmt Holdoff Exponent* en los nodos que tienen mayor prioridad, con el objetivo de aumentar el *throughput* y disminuir el retardo promedio.

El funcionamiento de la calendarización distribuida permite que cada estación suscriptora sea su propio administrador para la asignación de recursos, por tal motivo el diseño e implementación de este tipo de calendarización es más compleja que la calendarización centralizada. Debido a esto no existen muchos trabajos que se enfoquen en este tema, sin embargo se muestran a continuación los más relevantes.

En [12] realizan un algoritmo de asignación de *minislots* con el fin de asignar más números de *minislots* al nodo transmisor, haciendo que los segmentos de *minislots* disponibles sean grandes, y así satisfacer el nivel de demanda que cada solicitud requiera. Sin embargo los resultados mostrados indican que la utilización por *frame* es baja, ya que si el tamaño del segmento no cubre todo el nivel de demanda solicitado, este no sé utiliza, haciendo que esa parte del ancho de banda se desperdicie.

Por otra parte [13] plantean otros tres esquemas diferentes para mejorar la eficiencia de la calendarización de datos. El primer esquema que llamaron MG *(Multi-Grant)* pretende conceder todo el nivel de demanda que le fue solicitado por el nodo transmisor, con los diferentes segmentos de *minislots* libres que se encuentran en el frame, aumentando el número de *minislots* utilizados por *frame*. El segundo esquema es MR *(Multi-Request),* donde soporta múltiples solicitudes por parte del nodo fuente y el tercero es la combinación de los dos anteriores MRMG (*Multi Request- Multi-Grant*). Es importante señalar que solo hubo transmisiones de datos a un salto y que los nodos que solicitaron recursos no concedían ancho de banda.

Un escrito donde se lleva a cabo la transmisión de datos a más de un salto es en [14], aquí se le asigna un mayor peso al flujo de tráfico retransmitido, además de que utilizan un índice de equidad para evaluar la calidad de servicio del tráfico. Con ayuda del algoritmo FEBA (Fair End to end Bandwidth Allocation) y el algoritmo DDR (Deficit Round Robin) hacen la negociación del ancho de banda, ellos modificaron estos algoritmos para utilizar un *Xmt Holdoff Exponent* igual a cero. En los resultados presentados muestran que a medida que el número saltos aumenta el retardo también aumenta, además de que no cuenta con control de admisión.

En [15] proponen un algoritmo de calendarización distribuida coordinada, que consta en un administrador de ancho de banda basado en DDR y un calendarizador de paquetes con SCFQ (Self-clocked Fair Quering), este esquema permite asignar ancho de banda a cada flujo basándose en su prioridad.

Por último en [16] proponen un modelo de reservación de recursos, basándose en las conexiones activas que se encuentran dentro de la vecindad extendida de un nodo.

El tema de interés para este trabajo de investigación es la calendarización distribuida coordinada, sin embargo es importante conocer todos los trabajos relacionados que cubren el comportamiento de una red *mesh.*

En esta tesis se propone un algoritmo de calendarización de recursos, tomando en cuenta la sincronización del algoritmo de elección con el proceso de tres vías. El esquema para asignar los *minislots* será del tipo FIFO (First in – First out). Además que manejaremos solicitudes múltiples para reducir el número de veces que se invocaría el proceso de tres vías. También usaremos múltiples concesiones; este término es diferente con respecto a [13], ya que en este trabajo se refiere a que en un solo mensaje de concesión, el nodo receptor responderá a todos los nodos que le requirieron ancho de banda y no solamente a un nodo.

Para elaborar el algoritmo de calendarización propuesto, primero se realizará un modelo teórico, para calcular el número de *minislots* que le corresponden al *subframe* de datos, tomando en cuenta que para el *subframe* de control se le asignarán 16 oportunidades de transmisión, y con esto encontrar la máxima capacidad que un canal de 25 Mhz nos puede ofrecer.

Finalmente implementaremos el algoritmo desarrollado en el software OPNET Modeler v.8, ya que éste no cuenta con ningún tipo de algoritmo de reservación de recursos para redes *mesh*.

Para concluir, se comparará la máxima capacidad del canal que se obtuvo en el modelo teórico y los resultados de la simulación, para verificar que el algoritmo desarrollado funcione adecuadamente, además de simular otros escenarios para determinar el comportamiento de la red con respecto a la carga ofrecida.

Capítulo 3.

Descripción del protocolo IEEE 802.16

3.1 Introducción

El objetivo de este capítulo es explicar de manera clara el protocolo IEEE 802.16, enfocándonos en el modo *mesh*, ya que este protocolo puede trabajar tanto en modo *mesh* como en modo PMP. El modo *mesh* necesita del proceso de tres vías para poder solicitar y asignar ancho de banda, sin embargo proceso para asignar el ancho de banda es un tema abierto, ya que en el estándar no está definido.

Primero explicaremos los modos de operación para las redes *mesh,* ya que estas pueden trabajar de manera centralizada o de manera distribuida, ésta última es en la que nos enfocaremos para desarrollar nuestro algoritmo de calendarización. Definiremos como se utiliza el *frame* para la transmisión de paquetes de datos, ya que éste se divide en dos partes, la primera se utiliza para enviar los mensajes de solicitud y concesión de ancho de banda y la otra parte se utiliza para transmitir los paquetes de datos. Finalmente explicaremos detalladamente cada mensaje que se utiliza para el proceso de tres vías.

3.2 Modos de operación para redes *mesh*

Una red que utiliza un medio compartido debe tener un mecanismo de calendarización eficiente. En WiMAX se puede trabajar con la topología *mesh* o con PMP. La topología PMP soporta el modo TDD y FDD, mientras que la topología *mesh* sólo soporta TDD para la transmisión. Contrario al modo PMP, en el modo *mesh* no hay una separación en el *frame* para el enlace ascendente y para el descendente.

Las estaciones con la que un nodo está directamente conectado son llamados vecinos. Los vecinos de un nodo formarán una vecindad. Los vecinos de un nodo se consideran que están a un salto de distancia desde el nodo. Una vecindad extendida contiene, adicionalmente todos los vecinos de la vecindad.

Dependiendo del algoritmo de transmisión utilizado, este puede realizarse en base de la igualdad utilizando una calendarización distribuida, o en base a la superioridad de la estación base, la cual resulta en la calendarización centralizada o puede ser una combinación de ambas.

En la calendarización distribuida, todos los nodos coordinarán sus mensajes de transmisión en su vecindad de dos saltos y difundirán sus calendarios (recursos disponibles, solicitudes y concesiones) a todos sus vecinos. Los nodos asegurarán que las transmisiones resultantes no causen colisiones con el tráfico de datos y control programados para cualquier otro nodo en la vecindad de 2 saltos y competirán por el acceso al canal utilizando un algoritmo de elección.

Todas las comunicaciones están en el contexto de un enlace, el cual es establecido entre dos nodos. Un enlace se utilizará para todas las transmisiones de datos entre los dos nodos.

La clasificación de tráfico y la regulación del flujo son realizadas en el ingreso del nodo a la red, por el protocolo de clasificación/regulación de la capa superior. Los parámetros de servicios asociados con cada mensaje serán comunicados junto con el mensaje a través de la capa MAC.

Cuando lo autorice la red, el nodo candidato recibirá un identificador de 16 bits (Node ID) previa solicitud a la estación base. El ID del nodo es la base para identificar los nodos durante la operación normal. El ID del nodo se coloca en el subencabezado *mesh*, que sigue al encabezado MAC genérico, en mensajes unicast y broadcast.

Para direccionar los nodos en la vecindad local, se utiliza un identificador de enlace de 8 bits. A cada nodo se le asigna un ID por cada enlace y que está establecido hacia sus vecinos. Los ID de los enlaces son comunicados durante el proceso de establecimiento de enlace. El ID del enlace es transmitido como parte del CID en el encabezado genérico MAC en mensajes unicast. El ID del enlace se utiliza en la calendarización distribuida para identificar solicitudes y concesiones de recursos. Ya que estos mensajes son broadcast, los nodos receptores pueden determinar la programación utilizando el ID del nodo de los transmisores en el subencabezado *mesh*, y el ID del enlace en la carga del mensaje MSH-DSCH (Mesh Mode Schedule with Distributed Scedulling).

3.4 Características del *frame* para el modo de operación *mesh*

Un *frame* para el modo de operación *mesh* está dividido en un *subframe* de control y en un *subframe* de datos como se muestra en la figura 3.1.

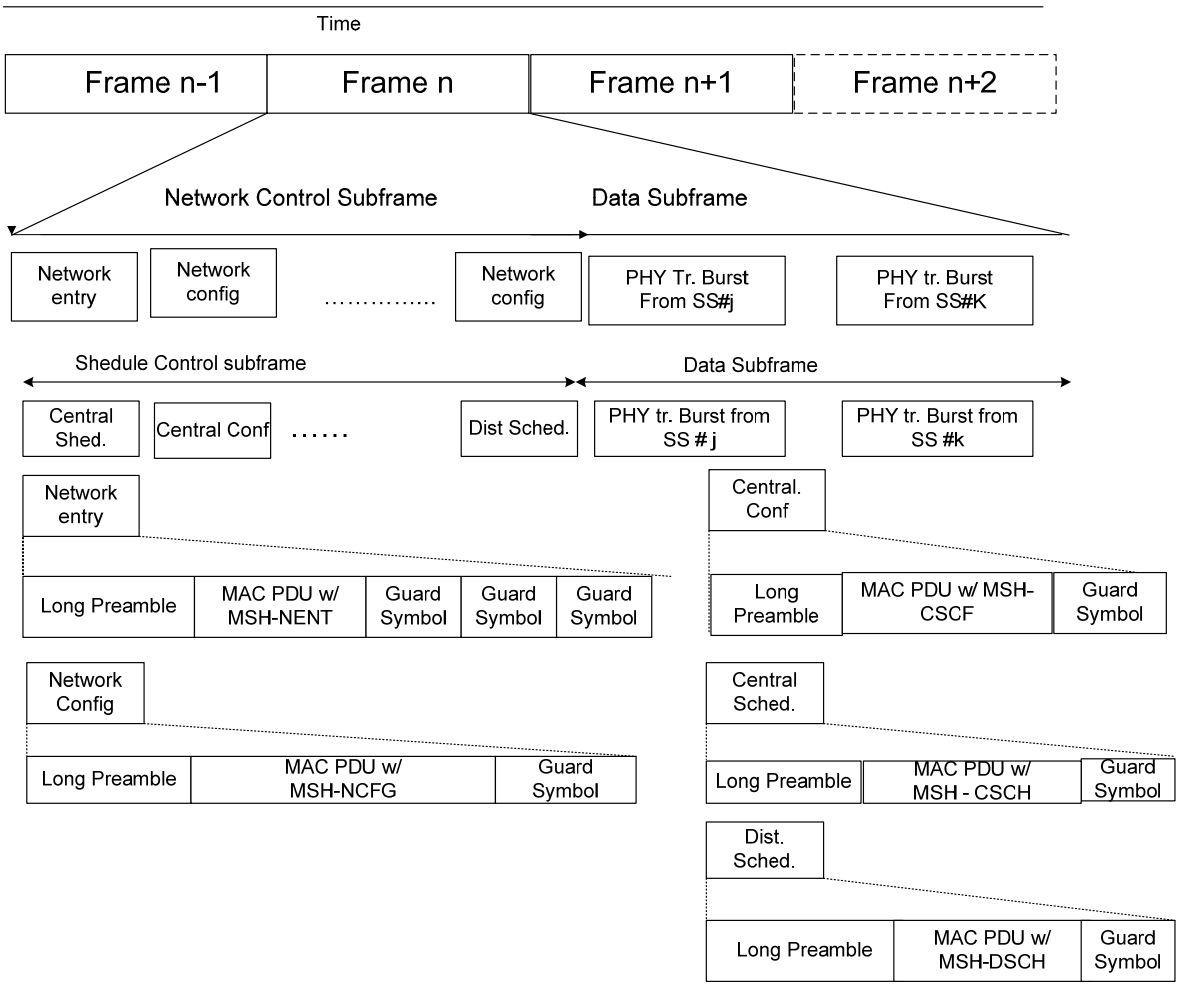


Figura 3.1: Características del *frame* para el modo de operación *mesh.*

El *subframe* de control tiene dos funciones básicas. Una es la creación y mantenimiento de cohesión entre los diferentes sistemas, llamado "Control de la Red" (Network Control). La otra función es la calendarización coordinada de la transferencia de datos entre sistemas, llamada "Control de Planeación" (Schedule control).

La longitud del *subframe* de control es MSH-CTRL-LEN X 7 símbolos OFDM, el MSH-CTRL-LEN está indicado en el mensaje *Network Descriptor*. Durante un *subframe* de control, el mensaje *Network Descriptor* indica cuantos mensajes de calendarización Distribuida (MSH-DSCH-NUM) pueden ocurrir en el *subframe* de control. El primer (MSH-CTRL-LEN – MSH-DSCH-NUM) X 7 símbolos OFDM está asignado para transmitir ráfagas que contengan mensajes de calendarización centralizada (MSH-CSCH) y MSH-CSCF (Mesh Centralized Scheduling Configuration), mientras que el resto es asignado para la transmisión de ráfagas que contengan mensajes para la calendarización distribuida (MSH-DSCH).

Los mensajes de calendarización distribuida además pueden ocurrir en el *subframe* de datos, si no entran en conflicto con la distribución realizada en el *subframe* de control. Todas las transmisiones en el *subframe* de control son enviadas utilizando QPSK-1/2 (Quadrature Phase Shift Keying).

Como indica el mensaje MSH-NCFG *Network Descriptor* hay un número de mensajes (MSH-DSCH-NUM) para la calendarización distribuida (MSH-DSCH). Esto implica que los primeros (MSH-CTRL-LEN) – (MSH-DSCH-NUM) * 7 símbolos OFDM son reservados para transmitir mensajes de calendarización centralizada (MSH-CSCH) y configuración centralizada *mesh* (MSH-CSF).

3.4.1 *Subframe* de Control de la Red

El *subframe* de control se compone de dos partes como se muestra en la figura 3.2. Los PDUs (Protocol Data Unit) de estas dos partes, para la entrada a la red y la configuración de la red, contienen dos mensajes: MSH-NENT y MSH-NCFG.

- MSH-NENT(Mesh Network Entry): Es un mensaje de administración que proporciona los medios para que un nuevo nodo obtenga la sincronización y la entrada inicial a la red en una red *mesh*.
- MSH-NCFG (Mesh Network Configuration): Es un mensaje de administración que proporciona un nivel básico de comunicación entre nodos de diferentes redes; este contiene parámetros del canal como la modulación, esquemas de codificación, etc.

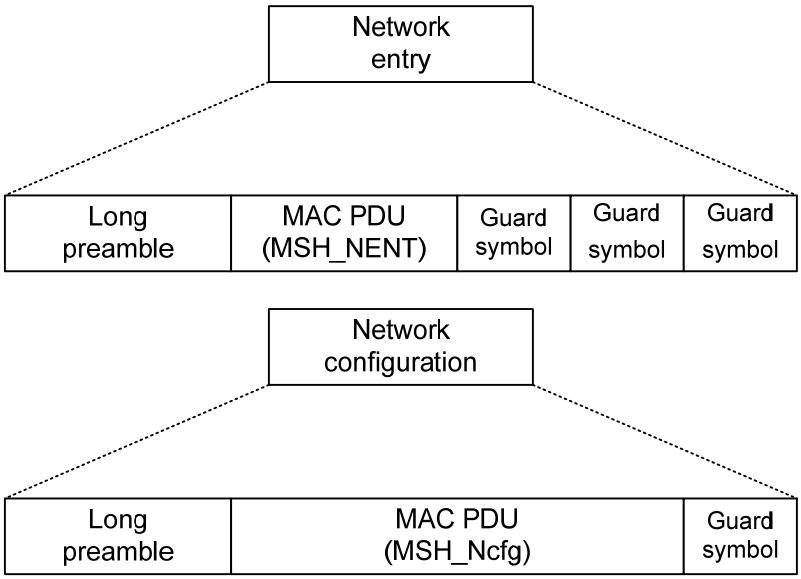


Figura 3.2: *Subframe* de Control del *frame mesh.*

3.4.2 *Subframe* de Control para la calendarización

El *subframe* de control que se utiliza para la calendarización se compone de tres partes.

El MAC PDU de estas tres partes son la configuración centralizada, la calendarización centralizada y la calendarización distribuida que contienen tres mensajes: MSH-CSCF, MSH-CSCH y MSH-DSCH, como se muestra en la figura 3.3.

- MSH-CSCF (Mesh Centralised Shedule Configuration) y MSH-CSCH (Mesh Centralised Schedule) son mensajes de administración y se utilizan en la calendarización centralizada.

- MSH-DSCH (Mesh Distributed Schedule): Es un mensaje de administración que se transmite cuando se utiliza la calendarización distribuida. En la calendarización distribuida coordinada, todos los nodos transmiten un mensaje MSH-DSCH a intervalos regulares para informar a todos sus vecinos acerca de los recursos libres, para que los vecinos pueden solicitar recursos.[3]

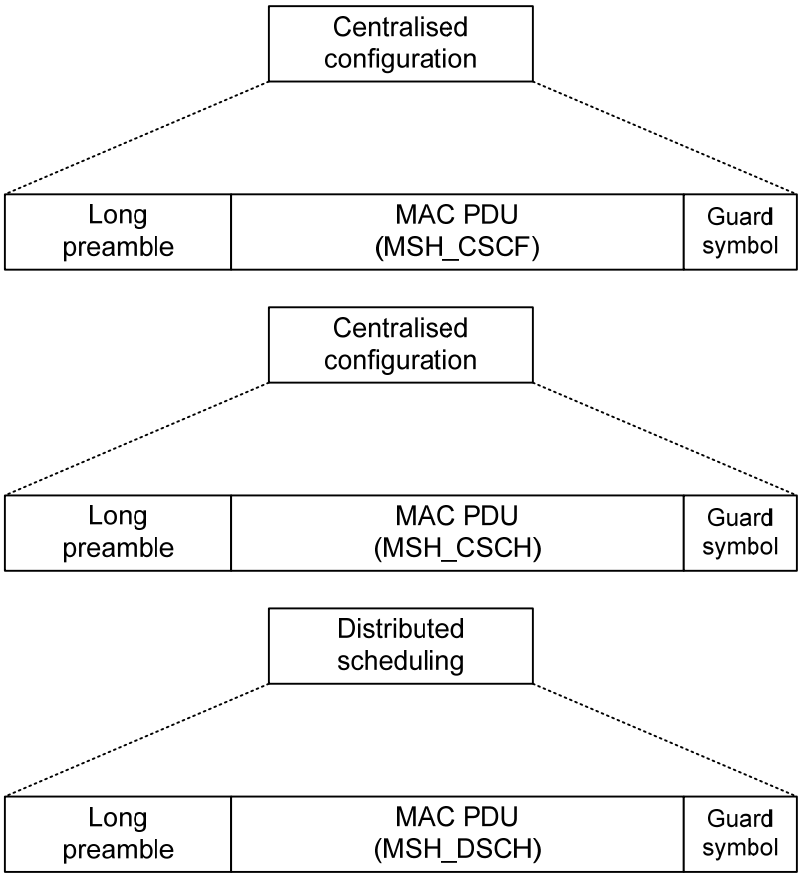


Figura 3.3: Las tres partes del *Subframe* de Control para la calendarización en el *subframe mesh.*

3.5 Mecanismos para envío de mensajes

3.5.1 Calendarización distribuida

El modo de calendarización distribuida coordinada utiliza alguna o una porción entera del *subframe de* control para enviar mensajes de su propia calendarización. Dentro de un canal dado, todas las estaciones vecinas reciben los mismos mensajes de calendarización. Todas las estaciones de la red utilizan el mismo canal para transmitir información en un formato específico para solicitud y concesión de recursos.

La calendarización distribuida coordinada asegura que las transmisiones están calendarizadas de manera que no dependan de la estación base, en este tipo de calendarización se enfocará este documento.

La calendarización distribuida no coordinada puede ser utilizada para una inicialización rápida, como por ejemplo ad-hoc que está basada en los enlaces. La calendarización distribuida no coordinada está establecida directamente para solicitudes y reservaciones entre dos nodos, y se programarán para asegurar que las transmisiones de datos resultantes (y los paquetes de solicitud y reservación en sí mismos) no causen colisiones con los datos ni con el tráfico de control para los métodos de calendarización centralizada y distribuida coordinada.

Tanto la calendarización distribuida coordinada y la no coordinada utilizan una negociación de 3 vías.

- MSH-DSCH: *Request*: Son los mensajes de solicitud de recursos.

- MSH-DSCH: *Grant*: Son los mensajes donde se conceden los recursos, y se envían en respuesta para indicar un subconjunto de las disponibilidades sugeridas que se ajustan a la solicitud.

- MSH-DSCH:Grant La reservación es enviada por el solicitante original y contiene una copia de la reservación de la otra parte, para confirmar.

La diferencia entre una calendarización distribuida coordinada y una no coordinada es que en el caso coordinado, los mensajes MSH-DSCH se han programado en el *subframe* de control de una manera libre de colisiones; mientras que en el caso no coordinado, los mensajes MSH-DSCH pueden colisionar [3].

3.5.2 Mensajes MSH-DSCH

En la calendarización distribuida coordinada, todos los nodos transmiten un mensaje MSH-DSCH en un intervalo regular para informar a todos sus vecinos sobre sus recursos disponibles. También estos mensajes son utilizados para transmitir solicitudes de recursos y concesiones hacia sus vecinos. A continuación se presentan los formatos de los mensajes de acuerdo al estándar [3].

El formato del mensaje MSH-DSCH se muestra en la tabla 3.1.

Coordination Flag

En este campo de la estructura del mensaje se establece el tipo de calendarización distribuida que se va a utilizar, ya sea coordinado y no coordinado.

0 = Coordinado
1 = No coordinado

Grant/Request Flag

En este campo se establece si es un mensaje para solicitar recursos, para concederlos o para confirmar.

0 = Mensaje de solicitud
1 = Mensaje de concesión, también es utilizado para la confirmación.

Sintaxis	Tamaño	Notas
MSH-DSCH_Message_Format() {		
Management Message Type =41	8 bits	
Coordination Flag	1 bit	
Grant/Request Flag	1 bit	
Sequence counter	6 bits	
No. Requests	4 bits	
No. Availabilities	4 bits	
No. Grants	6 bits	
Reserved	2 bits	Deberá ser 0
if (Coordination Flag == 0)		
MSH-DSCH_Scheduling_IE()	Variable	
for (i=0; i< No_Requests; ++i)		
MSH-DSCH_Request_IE()	16 bits	
for (i=0; i< No_Availabilities; ++i)		
MSH-DSCH_Availability_IE()	32 bits	
for (i=0; i< No_Grants; ++i)		
MSH-DSCH_Grant_IE()	40 bits	
}		

Tabla 3.1: Formato de mensajes MSH-DSCH.

Sequence Counter
Contador que se incrementa secuencialmente al solicitar mensajes en la calendarización distribuida no coordinada. Para la calendarización coordinada, este contador permite que los nodos detecten mensajes de calendarización perdidos.

No. Requests
Número de solicitudes en el mensaje.

No. Availabilities
Número de disponibilidades en el mensaje. Las disponibilidades son utilizadas para indicar rangos de *minislots* libres que los vecinos podrían utilizar para la reservación de recursos.

No. Grants
Número de concesiones en el mensaje.

3.5.2.1 MSH-DSCH Scheduling IE

La información para la calendarización distribuida coordinada se lleva a cabo en el mensaje MSH-DSCH y se utiliza para distribuir la información para determinar el tiempo de transmisión de los mensajes, los parámetros que utiliza este mensaje se observan en la tabla 3.2.

Sintáxis	Tamaño	Notas
MSH-DSCH_Scheduling_IE() {		
Next Xmt Mx	5 bits	
Xmt holdoff exponent	3 bits	
No. SchedEntries	8 bits	
for (i=0; i< No_SchedEntries; ++i) {		
Neighbor Node ID	16 bits	
Neighbor Next Xmt Mx	5 bits	
Neighbor Xmt holdoff exponent	3 bits	
}		
}		

Tabla 3.2: MSH-DSCH Scheduling IE.

Next Xmt Mx

Es el siguiente intervalo en donde el nodo puede transmitir y se calcula de la siguiente manera:

$$2^{Xmt\ Holdoff\ Exponent} * Next\ Xmt\ Mx < NextXmtTime \leq 2^{XmtHoldoffExponent} * (NextXmtMx + 1)$$

Neighbor Next Xmt Mx

Es el intervalo donde el nodo vecino puede transmitir.

Xmt Holdoff Exponent

El *Xmt Holdoff Time* es el número de oportunidades de transmisión después del Next Xmt Time que este nodo no es elegible para transmitir paquetes MSH-DSCH, es decir es el tiempo de espera para después poder transmitir un mensaje.

$$Xmt\ HoldOff\ Time = 2^{XmtHoldOffExponent+4}$$

Neighbor Xmt Holdoff Exponent

Comunica el *Xmt Holdoff Exponent* reportado por este vecino.

No. SchedEntries

Número de vecinos en las entradas para la calendarización del mensaje MSH-DSCH.

Neighbor Node ID

El ID del nodo del vecino al que se reportará.

3.5.2.2 *MSH-DSCH Request IE*

Las solicitudes que se envían en el mensaje MSH-DSCH deben transmitir las solicitudes de recursos en base al enlace. Las solicitudes deben incluir todos los parámetros listados en la tabla 3.3.

Sintáxis	Tamaño	Notas
MSH-DSCH_Request_IE() {		
Link ID	8 bits	
Demand Level	8 bits	
Demand Persistence	3 bits	
reserved	1 bit	Debe ser 0
}		

Tabla 3.3: MSH-DSCH Request IE.

Link ID

Es el ID asignado por el nodo transmisor hacia el enlace del vecino que involucra esta solicitud.

Demand Level

Demanda que se requiere para transmitir los datos y se solicita en *minislots*.

Demand Persistence

Campo persistente para la demanda. Es el número de *frames* en donde la demanda existe.

0 = reservación cancelada
1 = un frame
2 = 2 frames
3 = 4 frames
4 = 8 frames
5 = 32 frames
6 = 128 frames
7 = Bueno hasta cancelado o reducido

3.5.2.3 MSH-DSCH Availabilities IE

Las disponibilidades transmitidas en el mensaje MSH-DSCH deben ser utilizadas para indicar los rangos de *minislots* libres que los vecinos puedan utilizar para sus reservaciones. Las disponibilidades deben incluir todos los parámetros que se muestran en la tabla 3.4.

Sintáxis	Tamaño	Notas
MSH-DSCH_Availability_IE() {		
Start Frame number	8 bits	8 LSB del número de frame
Minislot start	8 bits	
Minislot range	7 bits	
Direction	2 bits	
Persistence	3 bits	
Channel	4 bits	
}		

Tabla 3.4: *MSH-DSCH Availabilities IE.*

Start Frame number
Es el *frame* donde empieza la disponibilidad de recursos. Se indica con los 8 bits menos significativos del número de *frame* en el cual comienza la disponibilidad.

Minislot start
La posición inicial de la disponibilidad dentro del *frame*.

Minislot range
El número de *minislots* libres para concederse.

Direction
0 = Rango de *minislost* no disponibles.
1 = Disponibilidad para transmisión en este rango de *minislot*.
2 = Disponibilidad para recepción en este rango de *minislot*.
2 = Disponibilidad para transmisión o recepción.

Persistence

Es el número de *frames* sobre el cual la disponibilidad es válida.

0 = Disponibilidad cancelada
1 = Un frame
2 = 2 frames
3 = 4 frames
4 = 8 frames
5 = 32 frames
6 = 128 frames
7 = Bueno hasta cancelada o reducida

Channel

Número de canal lógico. Un subconjunto de posibles números de canales físicos es mapeado hacia canales lógicos en el *Network Descriptor*.

3.5.2.4 *MSH-DSCH Grants IE*

Las concesiones que se encuentran en el mensaje MSH-DSCH deben transmitir información acerca del rango de *minislots* concedidos, seleccionados desde el rango reportado como disponibles. Estos mensajes se utilizarán como concesión y como confirmación. Los mensajes para las concesiones tienen los siguientes parámetros que se muestran en la tabla 3.5.

Sintaxis	Tamaño	Notas
MSH-DSCH_Grants_IE() {		
Link ID	8 bits	
Start Frame number	8 bits	8 bits menos significativos del número de frame inicial.
Minislot start	8 bits	
Minislot range	8 bits	
Direction	1 bit	
Persistence	3 bits	
Channel	4 bits	
}		

Tabla 3.5: *MSH-DSCH Grants IE.*

Link ID

El ID asignado al enlace que va del nodo transmisor hacia el vecino que involucra esta concesión.

Start Frame number

Es el *frame* donde empieza la disponibilidad de recursos. Se indica con los 8 bits menos significativos del número de *frame* en el cual comienza la disponibilidad.

Minislot start

La posición inicial de la reservación dentro de un *frame*.

Minislot range

El número de *minislots* reservados.

Direction

0= Desde el solicitante
1= Al solicitante

Persistence

Campo de persistencia para las concesiones. Número de *frames* sobre el cual la concesión está asignada.

0 = reservación cancelada
1 = un *frame*
2 = 2 *frames*
3 = 4 *frames*
4 = 8 *frames*
5 = 32 *frames*
6 = 128 *frames*
7 = Bueno hasta cancelar o reducir

Channel

Número de canal lógico, el cuál es el número lógico del canal físico. Un subconjunto de posibles números de canales físicos es mapeado hacia canales lógicos en el *Network Descriptor*.

Capítulo 4.

Algoritmo para la reservación de recursos

4.1 Introducción

Para que las redes *mesh* puedan solicitar ancho de banda y transmitir sus paquetes de datos, es necesario realizar el proceso de tres vías, los mensajes de este proceso se envían en las oportunidades de transmisión que cada nodo tiene, y se calculan de acuerdo a su Xmt HoldOff Exponent y el Xmt Mx de cada nodo. Por otra parte los algoritmos que se utilizan para asignar los recursos son muy importantes debido a que con ellos la eficiencia de una red puede mejorar.

Este capítulo describe como se diseñó el algoritmo distribuido coordinado para la calendarización de recursos en una red tipo *mesh*. El modelo desarrollado se implementó en el software llamado OPNET Modeler v.8 y consta de 3 fases. La primera fase fue crear el proceso para simular el tráfico que se desea transmitir con paquetes de tamaño constante. En la segunda fase se realizó el proceso de tres vías para que las estaciones suscriptoras soliciten recursos y puedan transmitir sus paquetes de datos, de acuerdo al estándar IEEE 802.16-2004, en la versión *mesh*. Finalmente se implementó la calendarización de recursos de acuerdo a los requerimientos solicitados.

4.2 Diseño del algoritmo de reservación de recursos

Modelo de la Red

Se construyó un modelo de red que consta de una estación base y seis estaciones suscriptoras tal y como se muestra en la figura 4.1.

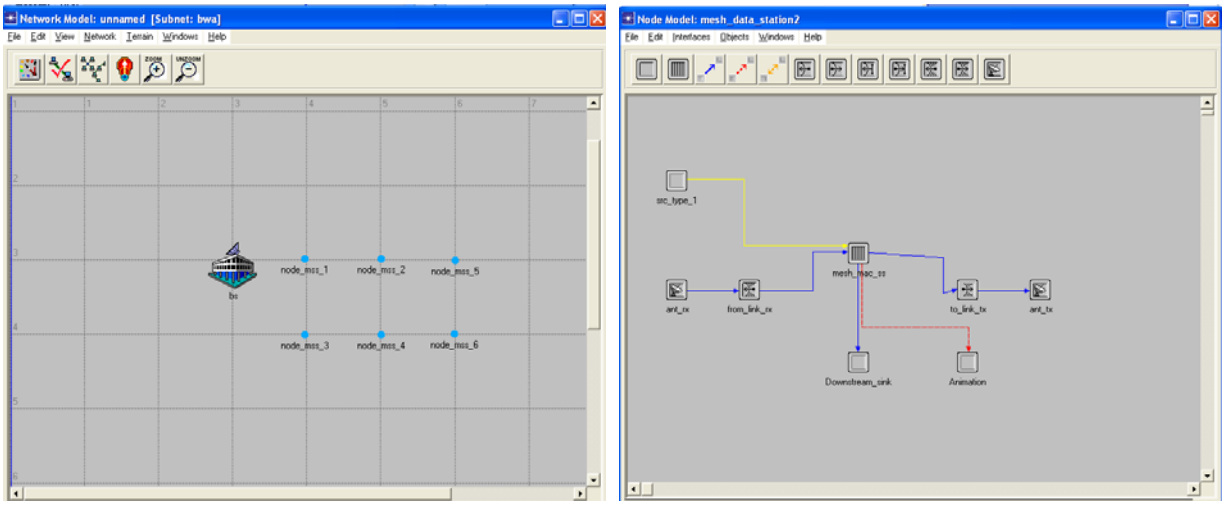


Figura 4.1: Modelo de Red y Modelo de Nodo.

Modelo de Nodos

En este modelo se representa la capa MAC y la capa de aplicación, las cuales se utilizaron para realizar el proceso de calendarización de recursos en las estaciones suscriptoras.

La capa de aplicación está representada por el nodo src_type_1, aquí se programó el tráfico que se transmitirá.

En la capa MAC está implementada en *mesh_mac_ss*, aquí se programó el proceso de tres vías y la asignación de recursos para los nodos que quieren transmitir. El modelo de nodos utilizado se muestra en la figura 4.1.

Modelo de Procesos

Para este trabajo existen dos modelos de procesos, uno para el tráfico y el segundo para la capa MAC, tal y como se muestran en las figuras 4.2 y 4.3 respectivamente.

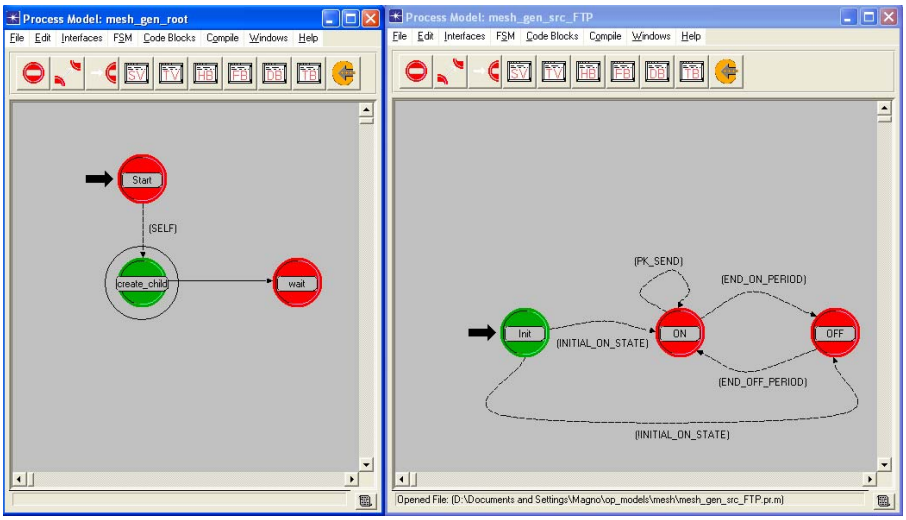


Figura 4.2: Modelos de Procesos para la transmisión de tráfico, implementado en OPNET Modeler v.8.

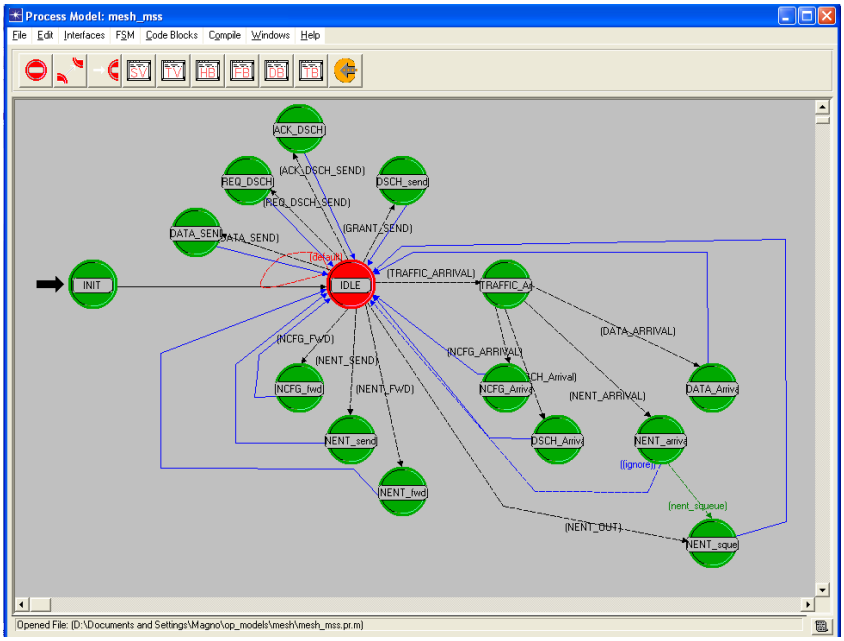


Figura 4.3: Modelo de Procesos implementado en OPNET Modeler para la capa MAC (*mesh_mac_ss)*.

A continuación se explica cada uno de los estados que se realizaron para solicitar y asignar recursos que se encuentran en la capa MAC (*mesh_mac_ss)*.

INIT

Aquí se inicializan las variables, se crean las listas y estructuras donde se almacena la información que se utiliza en la programación para solicitar y conceder recursos. De acuerdo a la dirección MAC de la estación suscriptora se genera la lista de sus vecinos que se encuentran a uno y a dos saltos, que son con los que competirá por las oportunidades de transmisión.

IDLE

Cuando llega un flujo de información a este estado, lo clasifica de acuerdo al tipo de formato que tenga el flujo, éstos pueden ser los mensajes utilizados para solicitar y conceder recursos *(REQUEST, GRANT y AC*K), o un paquete de datos que vino desde la capa de aplicación o desde otra estación suscriptora.

TRAFFIC_Arrived

En este estado llegan los diferentes flujos de información, ya sea los paquetes de datos o los paquetes de solicitud, concesión o confirmación. Aquí serán procesados para verificar el tipo de paquete que se trata y así enviarlo al siguiente estado que le corresponda.

DATA_Arrival

Aquí llegan los paquetes de datos que se crearon por la capa de aplicación y los que son enviados por otra estaciones suscriptoras para su retransmisión.

Cuando el paquete viene desde la capa de aplicación se obtiene su longitud en *bytes*, la dirección destino y la dirección donde se generó el paquete. Posteriormente se encapsula el paquete sumándole los encabezados de las capas TCP, IP, Ethernet y WiMAX como se muestra en la figura 4.4.

Ya encapsulado el paquete se almacena en una cola, la capacidad de ésta puede ser infinita o sólo puede soporta hasta 100 paquetes. Cuando un paquete llega y la cola ya está a su máxima capacidad en el caso de que la longitud de la cola sea finita, se destruye el paquete.

Cuando el paquete es el primero en la cola o si no hay proceso de tres vías en ejecución, se obtiene el tiempo de la oportunidad de transmisión que le corresponde para transmitir su solicitud de recursos, de acuerdo a sus parámetros *Xmt Holdoff Exponent, Next Xmt Mx* y el tiempo en que se recibió el paquete.

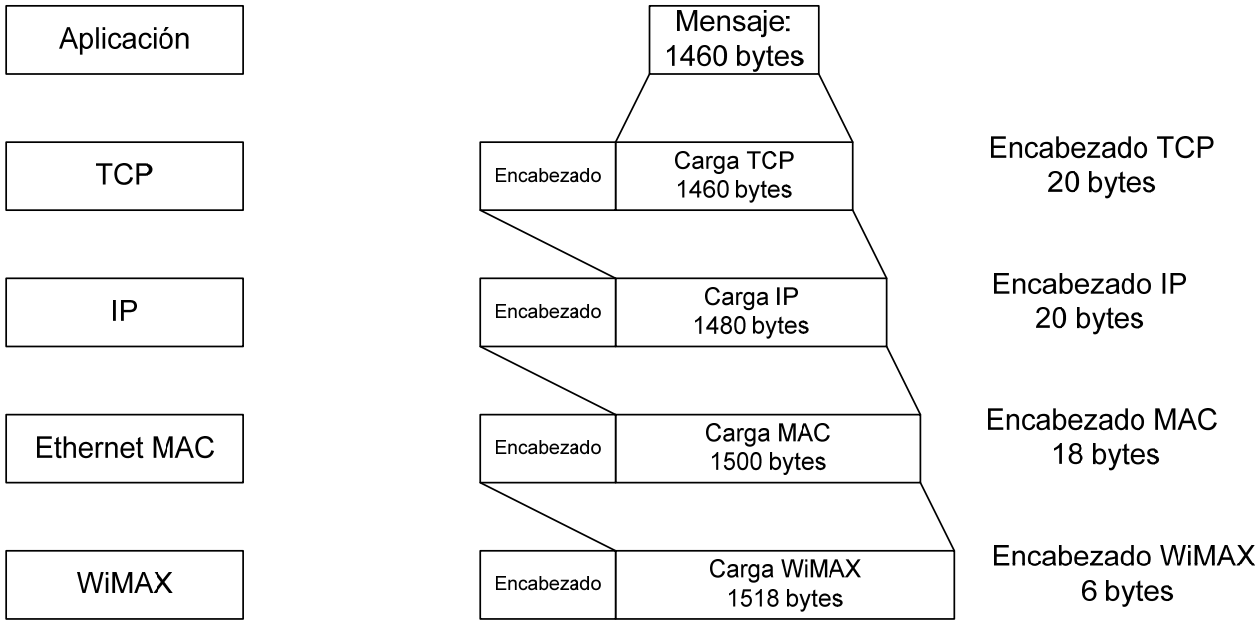


Figura 4.4: Encapsulamiento del mensaje a transmitir.

En caso de que el paquete venga desde otra estación suscriptora, se verifica si ya llegó a su destino final o si se requiere retransmitirse; en caso de que el paquete sea para él lo recibe y termina la transmisión, en caso contrario se re direcciona el paquete con la dirección del siguiente salto. El proceso para solicitar los recursos para transmitir el paquete es el mismo que se describió anteriormente.

DSCH_Arrival

En este estado llegan los paquetes de solicitud, concesión y confirmación que se utilizan en el proceso de tres vías. Estos paquetes son de tipo *broadcast*. En una solicitud o una concesión de recursos pueden ir dirigida a varias estaciones suscriptoras, por lo tanto se verificará el enlace para confirmar si el paquete es para ese nodo en especial.

Si el paquete es para solicitar recursos, se verifica el número de paquetes que se desean transmitir, también es necesario guardar en un arreglo la dirección MAC de donde vino la solicitud, ya que a esta misma estación suscriptora le pueden llegar más solicitudes de otras estaciones suscriptoras.

Se verificarán los parámetros que vienen en el paquete para poder asignarle recursos, tales como el nivel de demanda y la persistencia. Una vez que se hayan asignado los recursos, se programa la interrupción para generar el paquete de asignación de recursos *(GRANT)* y se envía.

En caso de que llegue un paquete *GRANT* a un nodo, también se verifica el enlace, en caso de que sea para él el paquete se realiza la interrupción para enviar la confirmación y también para preparar el envío de datos de información de acuerdo a los recursos dados.
En caso de que sea una confirmación, se termina el proceso de tres vías.

REQ_DSCH_SEND

Este estado es para configurar el paquete de solicitud de recursos, y se envía en el tiempo de la oportunidad de transmisión que le fue asignado, si es que gana, ya que es necesario competir con los nodos vecinos para evitar colisiones.

En caso de que gane, obtenemos el número de *slots* que se requieren para transmitir el paquete y la posición en la que se encuentra en la cola. Este proceso se realiza para todos los paquetes de datos que se encuentren en ese momento en la cola. Con la información recopilada se encapsula el paquete con el formato *REQUEST*, tal y como lo dice en el estándar. En caso de que no gane la oportunidad de transmitir se calcula la siguiente oportunidad de transmisión.

DSCH_send

Aquí se lleva a cabo el encapsulamiento del paquete de concesión de recursos, antes de crear el paquete y configurarlo con la información, es necesario que compita con sus nodos vecinos. En caso de que gane se forma el paquete *GRANT* donde se van a asignar los recursos.

Para asignar los recursos se realiza lo siguiente:

Se obtiene el número de solicitudes que le llegaron a la estación suscriptora, en caso de que se haya recibido más de una solicitud, se le asignarán los recursos a la primera que llegó, después a la segunda y así sucesivamente.

Del paquete de solicitud, obtenemos el número de paquetes de datos que se desean transmitir y la dirección MAC de donde vino el paquete.

Se calcula la oportunidad de transmisión de la estación suscriptora que transmitió la solicitud con el tiempo actual y sus parámetros *Xmt Holdoff Exponent y Next Xmt Mx*. Con la oportunidad para transmitir que obtuvo compite en el algoritmo de selección con sus vecinos, esto con el fin de asegurar en que tiempo exactamente se enviará el mensaje de confirmación.

Con el tiempo en que se transmitirá el mensaje de confirmación, se calcula el número de *frame* donde empezará a asignarle los recursos que necesita.

El número de *frame* y el número de *slot* que se les asignará para transmitir se comparan con el *frame* y el *slot* disponibles de las estaciones suscriptoras vecinas con el fin de evitar colisiones.

Ya que estén asignados los *slots*, estos valores se meten dentro del mensaje *GRANT*, para ser enviado.

ACK_DSCH

Este estado se utiliza para formar el mensaje de confirmación y para programar las interrupciones de los paquetes de datos para ser enviados en el tiempo especificado.

Primero obtenemos el número de mensajes *GRANT* que recibió la estación suscriptora. Para cada mensaje *GRANT*, se obtiene el número de paquetes de datos que recibieron recursos para transmitirse.

Para cada paquete de dato obtenemos su número de *frame* y *slot* que se les asigno para calcular el tiempo que le corresponde para transmitir.

Con la información que vino de los mensajes *GRANTS*, se forma el paquete *ACK* y se envía.

Finalmente se programan las interrupciones de los tiempos para transmitir todos los paquetes de datos.

DATA_SEND

Aquí se realiza el envío de paquetes de datos, primero verificamos que el tiempo en el que entró a este estado, sea el correcto y que se encuentre dentro de la lista de envío de paquetes, ya que hayamos encontrado el tiempo, obtenemos el índice en el que se encuentra en la cola.

Con el índice, obtenemos el paquete correcto y lo enviamos a su destino; y así sucesivamente se irán enviando los paquetes a los cuales se les programó su tiempo de transmisión.

En caso de que todavía haya paquetes en la lista de solicitudes que no hayan sido enviados, la variable *handshake_lock* permanece en 1, esto significa que todavía no han llegado todos los mensajes *GRANTS*.

Hasta que todos los paquetes que solicitaron recursos ya hayan sido transmitidos, las variables serán reiniciadas; en caso de que hayan llegado más paquetes a la cola y estos no hayan solicitado recursos, se programara una nueva interrupción para solicitar recursos.

4.3 Descripción de las variables utilizadas en el algoritmo

Las variables más importantes que son utilizadas para programar la solicitud de recursos son:

handshake_lock

El objetivo de esta variable es que en el momento que se van a solicitar recursos, el nodo quede bloqueado con el fin de que no solicite nuevamente recursos para los mismos paquetes.

Cuando tiene el valor de “0” significa que estamos solicitando recursos.

Cuando tiene el valor de “1” significa que ya se tiene el tiempo de transmisión del mensaje de *REQUEST*.

Si tiene el valor de “2” significa que estamos en el proceso de concesión de ancho de banda.

packet_data_queue

Esta es una lista en donde se van a guardar las solicitudes de recursos que se estén procesando en ese momento.

packet_send_queue

En esta lista se guardan los tiempos de transmisión de los paquetes de datos y el índice de donde se encuentra el dato en la cola.

tiempos_calendarizacion(*time_pk, Xmt Holdoff Exponent , Next Xmt Mx*)

Esta función realiza el cálculo de las oportunidades de transmisión de acuerdo al tiempo de llegada del paquete, su *xmt holdoff exponent* y su *next xmt mx*. El tiempo que arroje esta función, se utiliza para programar la interrupción en la que se enviarán los paquetes para el proceso de tres vías.

frame_request_DSCH

Es el número de *frame* en el que se encuentra la oportunidad de transmisión que le corresponde al tiempo en que llega un paquete de datos, este valor lo regresa la función tiempos_calendarizacion.

slot_request_DSCH

Es el número de *slots* en el que se encuentra la oportunidad de transmisión.

time_request_DSCH

Es el tiempo que le corresponde a la oportunidad de transmisión, este valor es el que se utiliza para programar la interrupción para el envío de mensajes de tres vías.

frame_ack

Este valor se utiliza para calcular el *frame* en el que van a empezar a transmitirse los datos de información.

start_Frame_number

Es el número de *frame* que le corresponde al dato de información que espera ser transmitido.

minislot_start_c

Número de *slot* en donde empezará a transmitirse el dato.

4.4 Descripción de los parámetros a utilizar

Para implementar el modelo es necesario entender los parámetros de la red que se van a utilizar:

Next Xmt Mx

Es el siguiente intervalo en donde el nodo puede transmitir y se calcula de la siguiente manera:

$$2^{Xmt\ Holdoff\ Exponent} * Next\ Xmt\ Mx < NextXmtTime \leq 2^{XmtHoldoffExponent} * (NextXmtMx + 1)$$

Xmt Holdoff Exponent

El *Xmt Holdoff Time* es el número de oportunidades de transmisión después del Next Xmt Time que este nodo no es elegible para transmitir paquetes MSH-DSCH, es decir es el tiempo de espera para después poder transmitir un mensaje.

$$Xmt\ Holdoff\ Time = 2^{XmtHoldoffExponent+4}$$

4.5 Implementación del algoritmo

Para este trabajo se utiliza un sistema que consta de un *frame* de señalización y 9 *frames* para calendarización, cada *frame* dura 10 ms.

Se implementó el algoritmo para utilizar 16 oportunidades de transmisión por *subframe* de control.

Las oportunidades de transmisión son los tiempos en donde el nodo puede transmitir sus paquetes de solicitud de recursos *(REQUEST),* concesión de recursos *(GRANT)* y confirmación (*ACK),* estos son los mensajes necesarios para poder llevar a cabo el proceso de tres vías y así solicitar los recursos que deseen, para obtener estas oportunidades se realizó un programa en c y se implementó en el software OPNET Modeler V.8, que se muestra en el apéndice C. Esta función está dividida en tres partes, la primera parte nos calcula el número de *slot* y el número de *frame* en el que se encuentra el paquete de datos que se desea transmitir.

Con esta información y los parámetros del nodo que se mencionaron en la sección anterior, podemos saber cuál es la oportunidad de transmisión que le corresponde para enviar sus mensajes de solicitud y concesión de recursos. Este cálculo se realiza en la segunda y tercera parte del programa, ya que debemos calcular el número de *frame* y *slot* en el que se encuentra la oportunidad y finalmente hacer la conversión al tiempo que le corresponde.

Para asignar los recursos fue necesario utilizar la función de calendarización junto con el algoritmo de selección, esto para encontrar el tiempo en que el nodo transmitirá su mensaje *ACK*, esto es debido a que después de enviar su confirmación debe empezar a enviar los datos. Por lo tanto el nodo enviará sus datos en el mismo *frame* que envió su confirmación, pero en la parte de *subframe* de datos.

Con el valor del *minislot star* y *star frame* que se le asignó al paquete de datos se realiza una comparación con los valores de sus vecinos para saber si algún otro nodo va a transmitir en ese tiempo, en caso de que otro nodo ya tenga asignando ese tiempo, se suma la cantidad de slots que ya están ocupados y así evitar que exista una colisión.

4.6 Características del canal

El ancho de banda del canal que ocupa este trabajo de investigación es de 25 MHz. Es necesario calcular el número de símbolos que tiene el *frame,* para asignar el número de *slots* que se ocuparán para el *subframe* de control y los símbolos que se utilizarán para el *subframe* de datos. Sabiendo esto, obtendremos el *Data Rate* máximo que se puede lograr con el sistema para la transferencia de datos.

La tabla 4.X indica el significado de cada término utilizado en el cálculo del máximo *Data Rate* que el sistema puede lograr para la transmisión de datos.

Variable	Definición
BW	Ancho de banda nominal del canal.
N	Factor de muestreo.
NFFT	Número de portadoras.
Fd	Tiempo que dura el *frame*.
Fs	Frecuencia de muestreo.
Df	Espacio entre subportadoras.
Tb	Tiempo útil del símbolo.
Ts	Tiempo del símbolo OFDM.
G	Tiempo de guarda.
Op. Tx	Oportunidad de transmisión que se utiliza para enviar los mensajes de solicitud, concesión y confirmación para la reservación de recursos.
Data Rate	Capacidad que tiene el sistema para la transmitir datos.
M	Número de bits por símbolo.
Cc	Tasa de codificación

Tabla 4.x: Definición de términos para obtener el máximo *Data Rate* del sistema.

$BW = 25\ [MHz]$
$n = \frac{8}{7}$
$NFFT = 256$
$Fd = 10\ [ms]$
$Fs = \ n * BW$
$Fs = \frac{8}{7} * 25\ MHz = 28.571429\ [MHz]$
$df = \frac{Fs}{NFFT}$
$df = \frac{28.571429\ [MHz]}{256} = 111607.143\ [Hz]$

$Tb = \frac{1}{df}$
$Tb = \frac{1}{111607.143\ [Hz]} = 8.96\ [\mu s]$
$Ts_{-1/8} = \ Tb + \left(\frac{1}{8} * Tb\right)$
$Ts_{-1/8} = 8.96\ [\mu s] + \left(\frac{1}{8} * 8.96\ [\mu s]\right) = 10.08\ [\mu s]$
Con $G = \frac{1}{8}$
No. de Símbolos OFDM $= \frac{Fd}{Ts\_1/8}$

No. de Símbolos OFDM $= \frac{10\ [ms]}{10.08\ [\mu s]} = 992.0635\ [s\acute{i}mbolos]$

Cálculo de los slots para el *subframe* de control:

El *frame* tiene 992 símbolos. Para este trabajo voy a considerar 16 oportunidades de transmisión para el *subframe* de control, cada oportunidad de transmisión consta de 7 símbolos OFDM, por lo tanto tenemos que:

$Op.Tx = 7\ s\acute{i}mbolos$
$16\ Op.de\ Tx. = 16 * 7 = 112\ s\acute{i}mbolos$

Por lo tanto estamos ocupando 112 símbolos para el *subframe* de control y nos quedan 880 símbolos para el *subframe* de datos, tal y como se muestra en la figura 4.5.

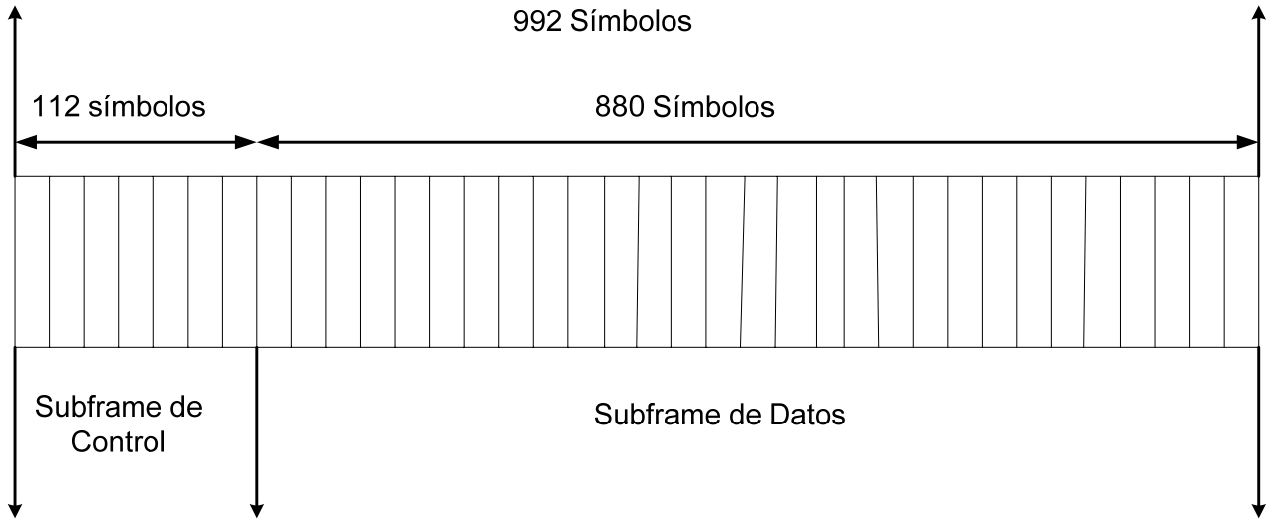


Figura 4.5: División del *subframe* de control y del *subframe* de datos en símbolos.

Cada *minislot* para transmitir datos es de 4 símbolos, por lo tanto contaremos con 220 *minislot* en el *subframe* de datos para transmitir información como se muestra en la figura 4.6.

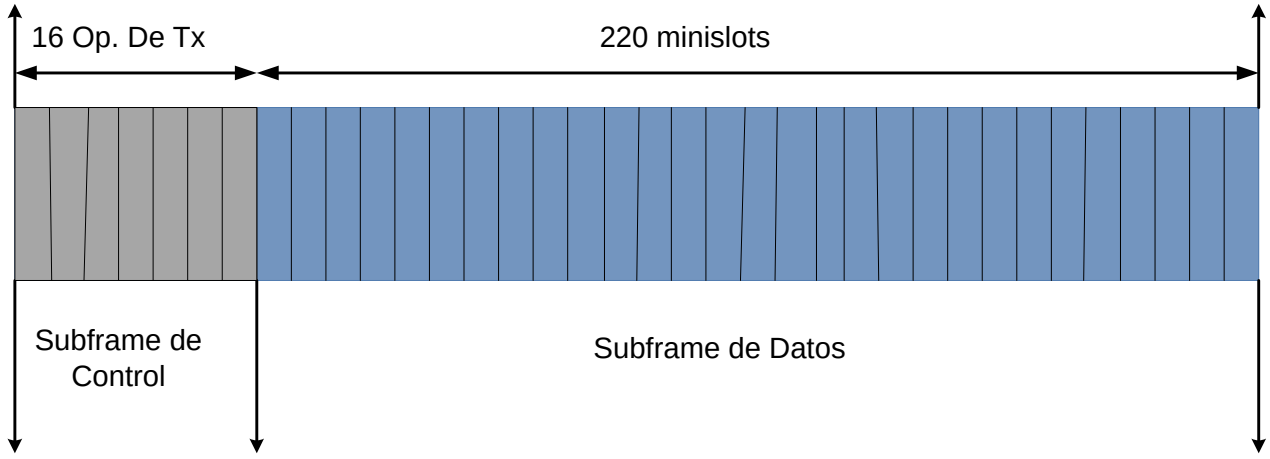


Figura 4.6: Número de *minislots* que corresponden al *subframe* de datos.

Ahora vamos a calcular *Data Rate* del Sistema, para saber cuál es la capacidad máxima que este sistema nos va a brindar.

$$Data\ Rate = \frac{\text{número de bits decodificados por símbolo OFDM}}{\text{duración del símbolo OFDM}}$$

$$Data\ Rate = \frac{\text{no. de portadoras para tx} * m * cc}{Ts_{-1/8}}$$

$$Data\ Rate = \frac{192 * 2 * 1/2}{10.08\ [\mu s]} = 19.047619\ [Mbps]$$

Como se mencionó anteriormente para este trabajo tomamos 112 símbolos para el *subframe* de control y 880 símbolos para el *subframe* de datos, por lo tanto el máximo *data rate* para transmitir los paquetes de datos y para la transmisión de paquetes de control se muestra en la tabla 4.1.

Descripción	Data Rate	Porcentaje del canal
Sistema	19.04761 [Mbps]	100.00 %
Subframe de datos	16.89714 [Mbps]	88.71 %
Subframe de control	2.15047 [Mbps]	11.29 %

Tabla 4.1: Máxima capacidad del canal

4.7 Análisis del algoritmo

Es necesario verificar que el tiempo de las oportunidades de transmisión sea el correcto, en la tabla 4.2 se muestran los parámetros utilizados para el cálculo.

Parámetro	Valor
Holdoff exponent	0
mx	1

Tabla 4.2: Parametros Xmt Holdoff exponent y mx.

Con los parámetros anteriores y utilizando 16 *minislots* para el *subframe* de control se muestra en la figura 4.7 las oportunidades de transmisión correspondientes que son los *minislots* que están de color lila.

El *minislot* que está de color rojo representa un paquete de datos que llegó en ese tiempo, se observa que su oportunidad para solicitar recursos se encuentra después del *frame* que se utiliza para señalización, en este caso está representado con el color morado.

Cada *minislot* tiene una duración de 68µs y cada frame es de 10 ms, en el algoritmo propuesto no se toma en cuenta el *frame* de señalización, con esto tenemos que los valores tanto del tiempo del paquete de datos y de su oportunidad de transmisión son los mostrados en la tabla 4.3.

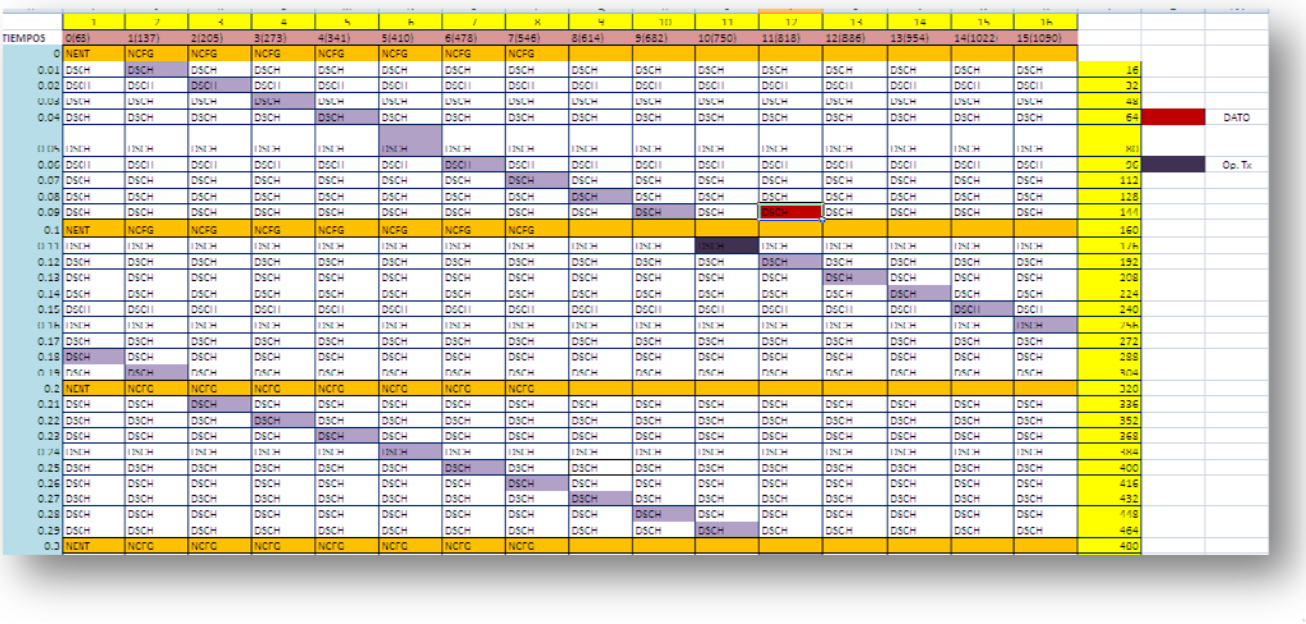


Figura 4.7: Oportunidades de transmisión para un Xmt Holdoff exponent de 0 y mx de 1.

Paquete de datos	Slot 12, Frame 9	Tiempo: 0.090816 [seg]
Oportunidad de transmisión	Slot 11, Frame 10	Tiempo: 0.110748 [seg]

Tabla: 4.3 Tiempo correspondiente a la oportunidad de transmisión.

4.8 Resultados

Para verificar que la función que calcula las oportunidades de transmisión que se implemento en OPNET Modeler v.8 esté funcionando correctamente, se utilizarán los parámetros del paquete de datos anteriormente mencionado.

En la figura 4.8 se muestra la pantalla de OPNET Modeler v.8, aquí se observa que la oportunidad de transmisión que nos da el algoritmo es de 0.110748 y comparándola con la que se obtuvo teóricamente se verifica que es la misma.

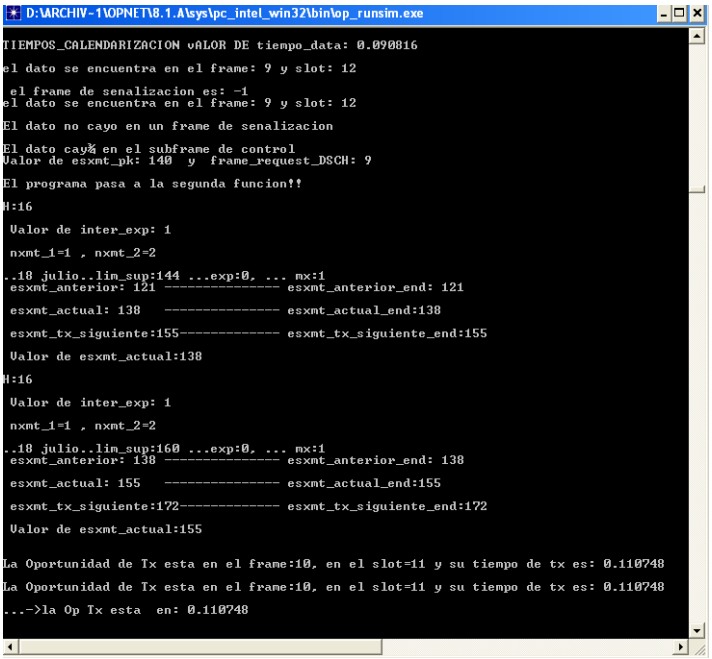


Figura 4.8: Resultados obtenidos mediante OPNET Modeler v.8.

Capítulo 5.

Comportamiento dinámico de las redes WiMAX en modo *mesh*

5.1 Introducción

El estudio del comportamiento dinámico de las redes es muy importante, ya que sirve para caracterizar el comportamiento de estos sistemas, y así poder determinar hasta donde funcionar correctamente.

En este tema de tesis se analizará el comportamiento dinámico de la redes *mesh* con el algoritmo de reservación de recursos propuesto. Examinaremos cómo se comporta el modelo desarrollado, variando la carga ofrecida en los seis usuarios que transmiten datos con el fin de visualizar el comportamiento que presentan en la utilización del sistema, el retardo que se genera al enviar los paquetes de datos y el *throughput* que se alcanza para la transmisión de información.

Con los resultados mostrados en este documento, verificaremos la capacidad máxima que tiene un canal de 25 Mhz, y como el *througput* máximo disminuye al ir aumentando el número de saltos en que se encuentra el nodo destino de la estación transmisora.

5.2 Análisis del algoritmo implementado

El algoritmo de calendarización que se desarrolló está implementado en una red con seis estaciones suscriptoras y una estación base, como se muestra en la figura 5.1. Cada SS tiene sus propios parámetros que se utilizarán para el cálculo de sus oportunidades de transmisión, para poder enviar sus mensajes de solicitud, concesión y confirmación y reservar recursos. En la tabla 5.1 se observan estos parámetros para cada SS.

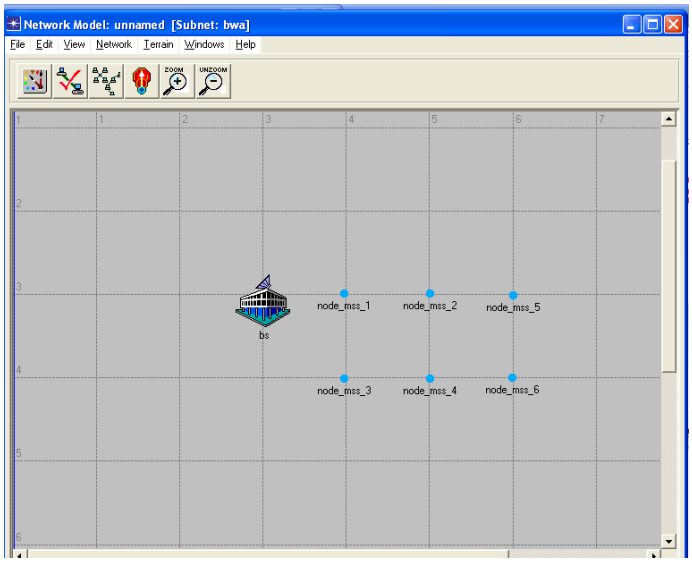


Figura 5.1: Modelo de Red con 6 estaciones suscriptoras.

Nombre del nodo	ID del nodo	Xmt HoldOff Exponent exp	Xmt mx	Xmt HoldOff Time H
Node_mms_1	1001	0	2	16
Node_mms_2	1002	0	3	16
Node_mms_3	1003	0	4	16
Node_mms_4	1004	0	5	16
Node_mms_5	1005	0	6	16
Node_mms_6	1006	0	7	16

Tabla 5.1: Parámetros de las estaciones suscriptoras.

A continuación se presentan los escenarios que se simularon en OPNET, con el propósito de conocer la máxima capacidad del sistema que puede ofrecer, las aplicaciones que se pueden utilizar de acuerdo al retardo que se obtuvo y el *throughput* ideal de acuerdo a la carga ofrecida.

Escenario 1:

En este escenario un usuario transmitirá paquetes de datos de tamaño de 1460 bytes a otro usuario que se encuentra a un salto de distancia de él, como se muestra en la figura 5.2.

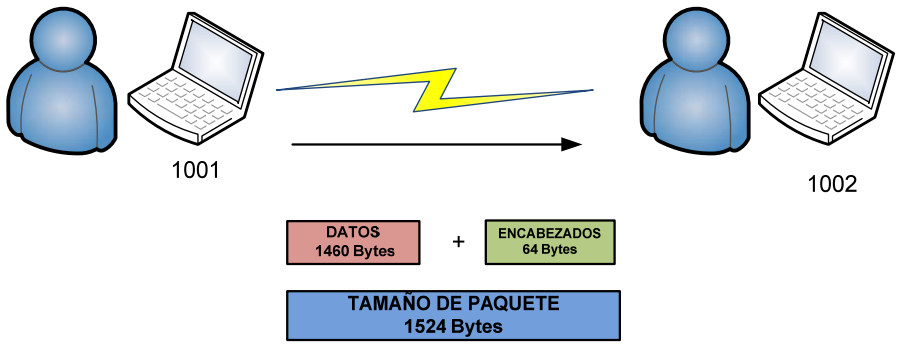


Figura 5.2: Un usuario transmitiendo datos a un salto

El objetivo de este escenario es mostrar cual es la máxima capacidad que tiene el canal y para lograrlo se va a ir incrementando la carga ofrecida hasta saturar el canal.

Las gráficas que se presentarán a continuación exponen la utilización del canal, el retardo de los paquetes que se enviaron y el *throughput* que se logró.

- **Utilización**

En la figura 5.3 se presenta la utilización que se obtuvo al transmitir paquetes de datos que viajan a través de un salto. La carga ofrecida con la cual se empezó la simulacion fue de 7 Mbps, con esta carga la utilización que se alcanzó fue del 40 % aproximadamente, esto se debe que a pesar de que la carga ofrecida es relativamente alta, sólo un usuario está transmitiendo datos y por lo tanto el resto del canal se está desperdiciando. Tambien se observa que a partir de que se suminstra una carga ofrecida de 16 Mbps la utilización ya no crece de manera gradual, es decir que ya se empezó a saturar el canal. Finalmente se obtuvo una utilizacion del 85 % con una carga ofrecida de 20 Mbps. Estos valores son satisfactorios, debido a que si se pudo saturar el canal, ya que sólo se puede utilizar el 88% del canal para la transmisión de datos, de acuerdo al modelo teórico desarrollado en el capítulo 4.

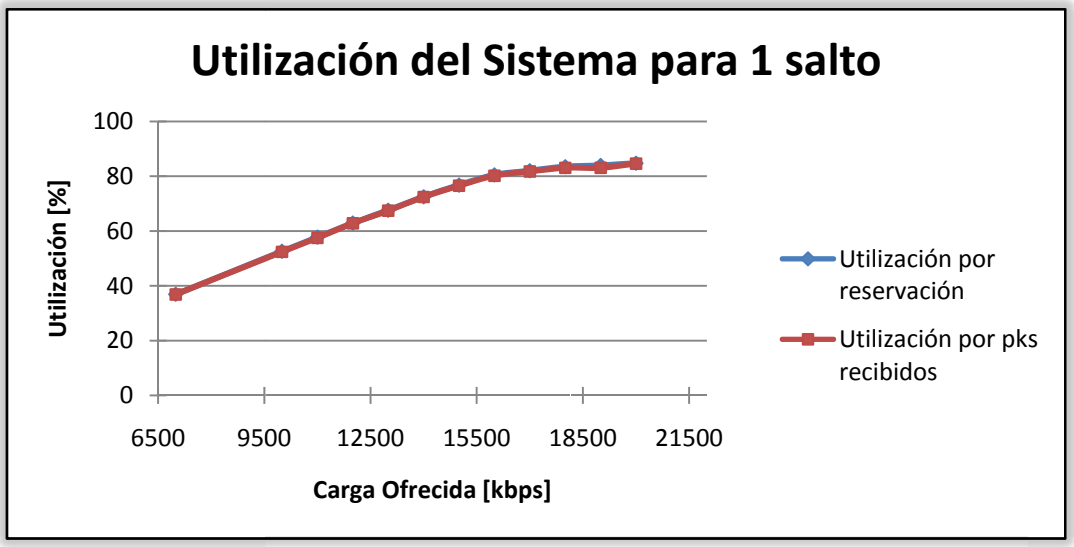


Figura 5.3: Utilización del sistema con un usuario transmitiendo a un salto.

- **Retardo**

En la figura 5.4 se exibe el comportamiento que el retardo presenta al ir aumentando la carga ofrecida, cuando los paquetes de datos viajan a un salto de distancia. Para una carga ofrecida de 7 Mbps, se obtiene un retardo de 35 ms, sin embargo, al ir aumentando la carga, el retardo tambien va creciendo pero de manera continua, hasta llegar a los 135 ms con una carga ofrecida de 15 Mbps. A partir de este momento el retardo crecerá más rápido debido a q la capacidad que el sistema tiene para asignar slots de transmisión a los paquetes a llegado a su límite, y por lo tanto los paquetes permanecerán más tiempo encolados hasta que puedan ser transmitidos, obteniendo un retardo máximo de 600 ms para una carga ofrecida de 20 Mbps.

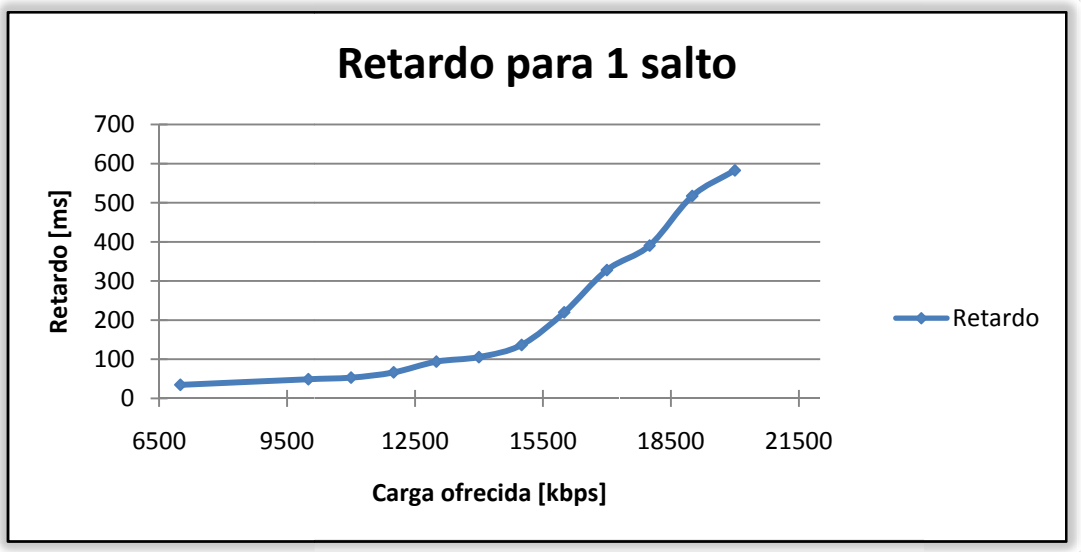


Figura 5.4: Retardo del sistema con un usuario transmitiendo a un salto.

- *Throughput*

En la figura 5.5 se presenta el *throughput* que puede alcanzar el sistema a diferente carga ofrecida, para un usuario. El objetivo de conocer el *throughput* es para saber cual es la máxima capacidad que el sistema tiene para entregar los paquetes de datos a su destino. Analizando la gráfica se observa que para una carga ofrecida que va desde los 7 Mbps hasta los 16 Mbps el *throughput* se comporta de manera satisfactoria ya que los paquetes son entregados a sus destino de manera eficiente. Cuando la carga ofrecida es mayor a los 16 Mbps, el *throughput* que ofrece el sitema ya no rebasa los 16 Mbps, por lo tanto el máximo *throughput* que puede ofrecer este modelo es de 16 Mbps.

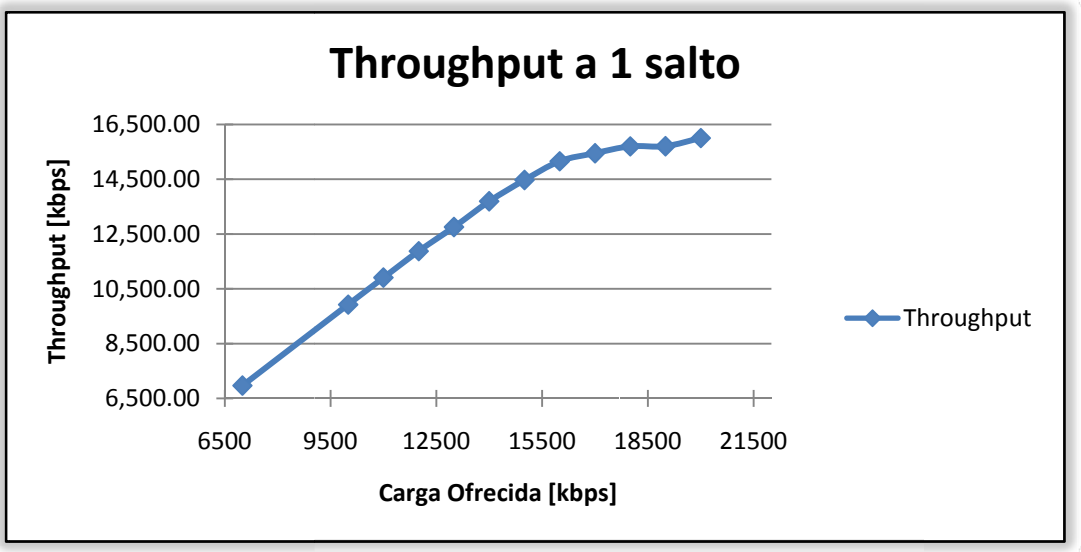
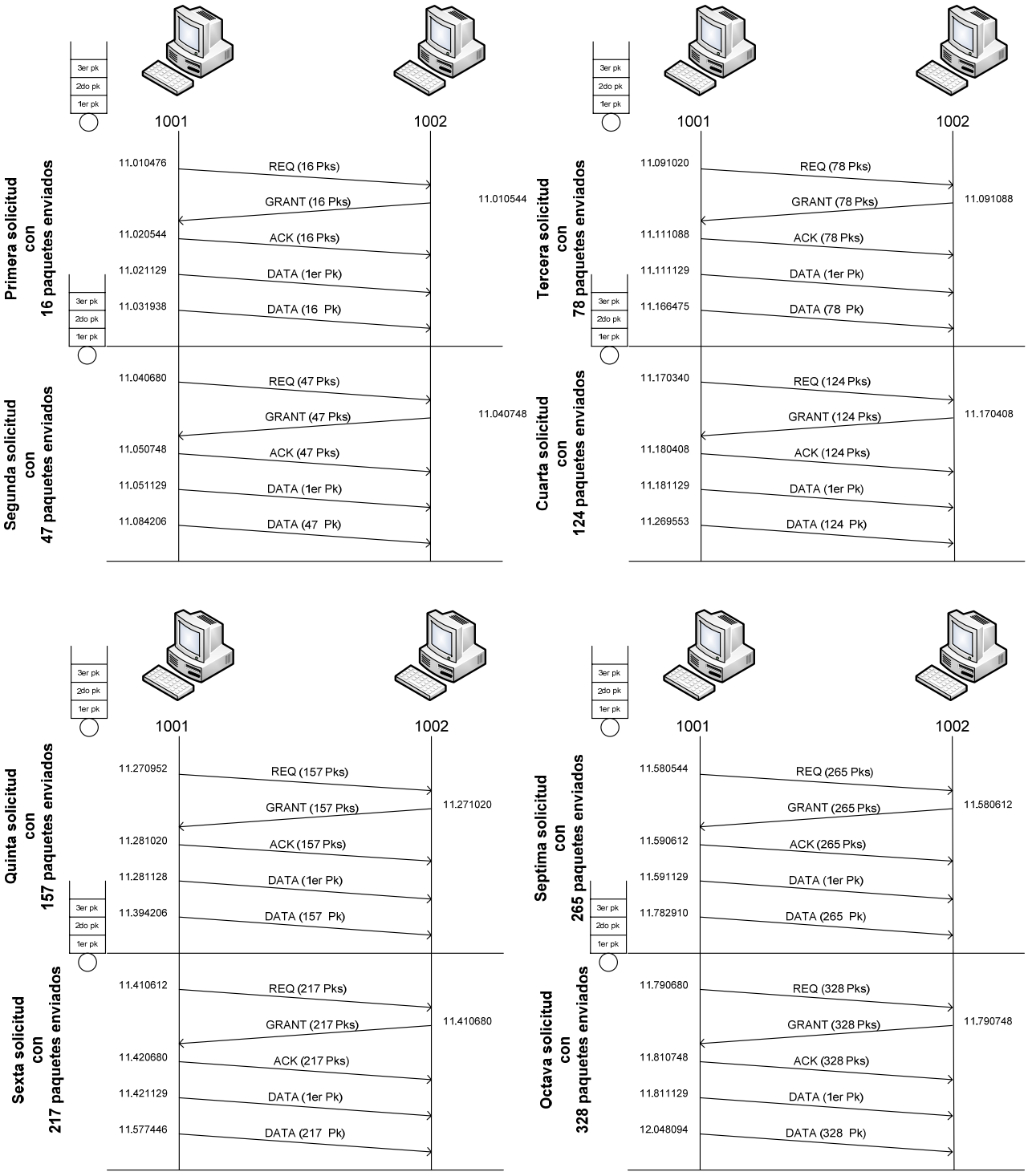


Figura 5.5: ***Throughput*** **del sistema con un usuario transmitiendo a un salto.**

Para verificar que el modelo propuesto esté funcionando correctamente, se realizó el anális más detallado del proceso que se lleva a cabo para transmitir los paquetes de datos para la carga ofrecida de 19 Mbps. Los resultados obtenidos en la simulación, tanto del *throughput* como la utilización serán comparados con el modelo teórico para conocer la exactitud que presentó nuestro modelo implementado.

En la figura 5.6 se presentan los tiempos en los que se realiza el proceso de tres vías para solicitar recursos y transmitir los paquetes de datos. Primero se envía un mensaje de solicitud, indicando el número de paquetes y el núemro de slots que requiere por paquete para transmitirlos. Posteriormente que el nodo receptor reciba la solicitud, este enviará su mensaje de conseción donde le indicara al nodo transmisor el número de *frame* y el número de *slot* que le corresponde a cada paquete para su transmisión. Finalmente cuando llegue el mensaje de conseción al transmisior el enviará su mensaje de confirmación, y esperará a que llegue el tiempo que se le asignó para enviar sus datos.

Como observamos en la figura 5.6, el número de paquetes que solicitán recursos va aumentando, esto debe a que a medida que se realiza el proceso de tres vías y el envío de los paquetes de datos, la cola que está en la capa MAC sigue almacenando los paquetes que viene de la capa de aplicación y que desean solitar recursos.


3er pk
2do pk
1er pk
1001
1002
Primera solicitud con 16 paquetes enviados
11.010476
REQ (16 Pks)
GRANT (16 Pks)
11.010544
11.020544
ACK (16 Pks)
11.021129
DATA (1er Pk)
11.031938
DATA (16 Pk)
Segunda solicitud con 47 paquetes enviados
11.040680
REQ (47 Pks)
GRANT (47 Pks)
11.040748
11.050748
ACK (47 Pks)
11.051129
DATA (1er Pk)
11.084206
DATA (47 Pk)
Tercera solicitud con 78 paquetes enviados
11.091020
REQ (78 Pks)
GRANT (78 Pks)
11.091088
11.111088
ACK (78 Pks)
11.111129
DATA (1er Pk)
11.166475
DATA (78 Pk)
Cuarta solicitud con 124 paquetes enviados
11.170340
REQ (124 Pks)
GRANT (124 Pks)
11.170408
11.180408
ACK (124 Pks)
11.181129
DATA (1er Pk)
11.269553
DATA (124 Pk)
Quinta solicitud con 157 paquetes enviados
11.270952
REQ (157 Pks)
GRANT (157 Pks)
11.271020
11.281020
ACK (157 Pks)
11.281128
DATA (1er Pk)
11.394206
DATA (157 Pk)
Sexta solicitud con 217 paquetes enviados
11.410612
REQ (217 Pks)
GRANT (217 Pks)
11.410680
11.420680
ACK (217 Pks)
11.421129
DATA (1er Pk)
11.577446
DATA (217 Pk)
Septima solicitud con 265 paquetes enviados
11.580544
REQ (265 Pks)
GRANT (265 Pks)
11.580612
11.590612
ACK (265 Pks)
11.591129
DATA (1er Pk)
11.782910
DATA (265 Pk)
Octava solicitud con 328 paquetes enviados
11.790680
REQ (328 Pks)
GRANT (328 Pks)
11.790748
11.810748
ACK (328 Pks)
11.811129
DATA (1er Pk)
12.048094
DATA (328 Pk)

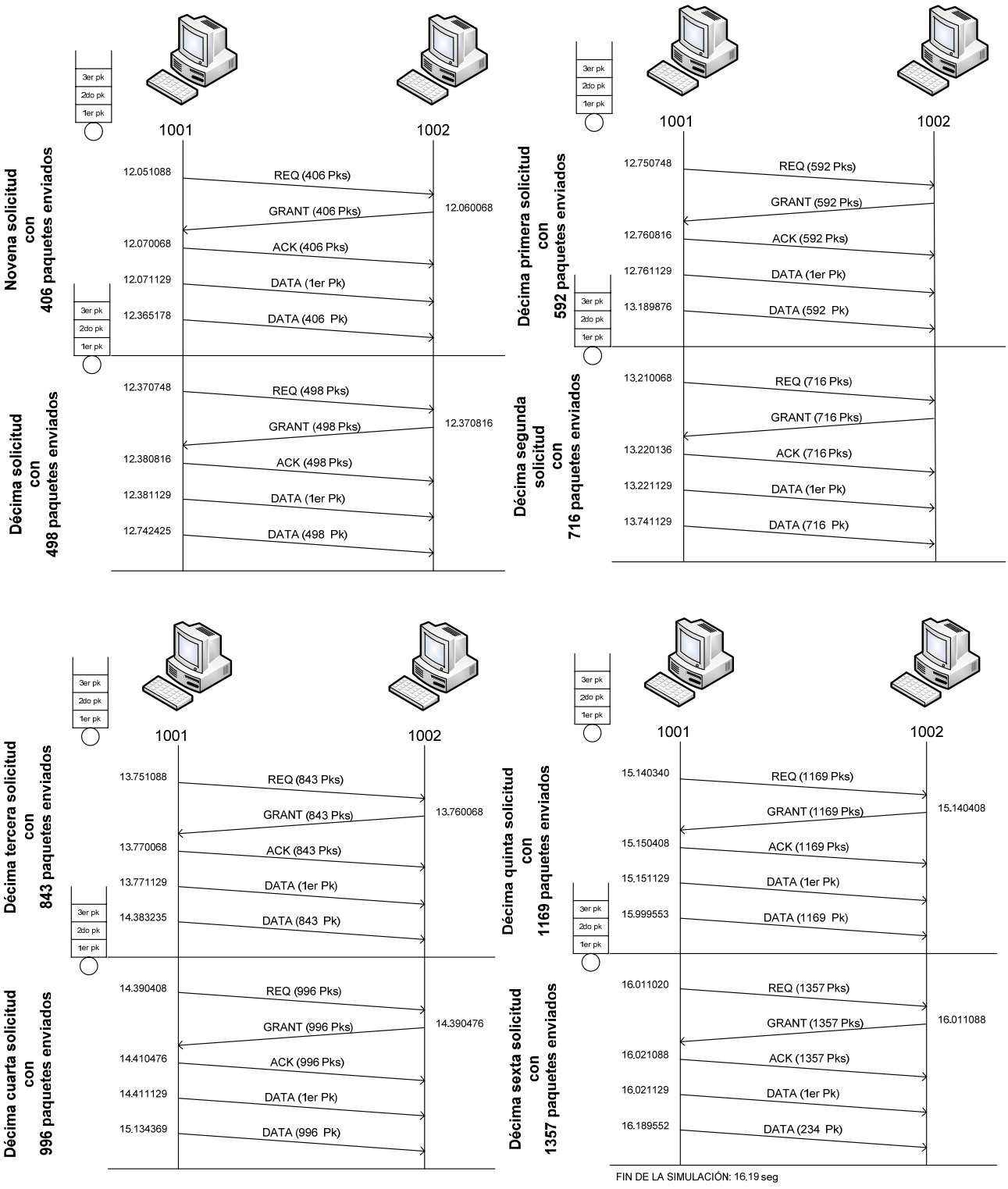


Figura 5.6: Proceso de calendarización para una carga ofrecida de 19Mbps.

Con los tiempos mostrados en la figura 5.6 y con los datos que se utilizaron para configurar el modelo de calendarización tenemos:

Tiempo de inicio de envío de tráfico: 11 [seg]
Tiempo final de simulación: 16.19 [seg]
Tiempo de evío de tráfico: 5.19 [seg]
El *througput* que obtuvimos por medio del modelo implementado para una carga ofrecida de 19 [Mbps] fue de: 15.707636096 [Mbps]

Tiempo utilizado para transmitir sólo paquetes de datos fue de: 4.842082 [seg]
Tiempo utilizado por el proceso de 3 vías fue de: 0.34747 [seg]

Si para el tiempo de 4.842082 [seg] tenemos que el *throughput* es de 15.707636096 [Mbps] y para 0.34747 [seg] que es el tiempo que no se transmitieron datos, es de 1.127188 [Mbps], tenemos un *throughput* total de 16.8348 Mbps utilizado para una carga ofrecida de 19 Mbps en un canal de 25 MHz.

Si sabemos por el modelo teórico que se desarrolló que el *throughput* máximo para este canal es de 16.89714 Mbps y por medio del modelo implementado tenemos un *throughput* de 16.8348 Mbps, tenemos que el porcentaje de error es del 0.3689 %, eso significa que nuestro modelo de calendarización tiene una exactitud del 99.631 %. Con esto demostramos que nuestro modelo fue implementado correctamente y que simulaciones más complejas se pueden llevar a cabo.

En el Apéndice B se muestra un diagrama de los *minislots* utilizados para la transmisión de paquetes, y los *slots* utilizados por el proceso de tres vías para la carga ofrecida de 19 Mbps.

Escenario 2:

El siguiente escenario muestra el comportamiento que tiene la red cuando un usuario desea transmitir información a un destino, donde los paquetes tienen que viajar a través de dos saltos para llegar a él. El tamaño de los paquetes sigue siendo de 1460 bytes, como se muestra en la figura 5.7.

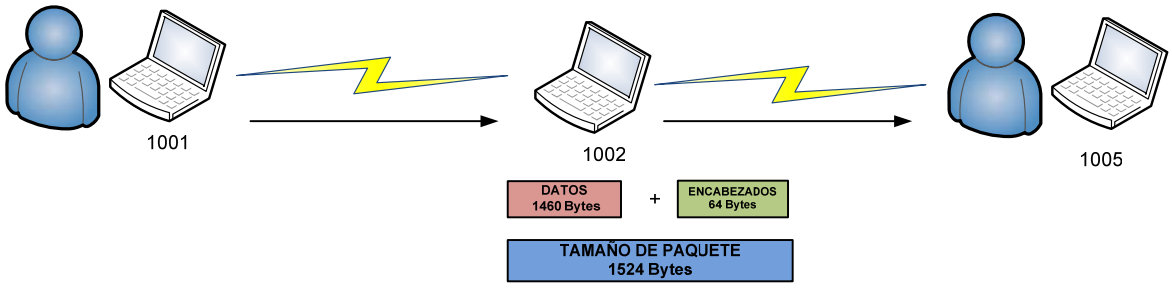


Figura 5.7: Un usuario transmitiendo datos a dos saltos.

Analizaremos los cambios que ocurren con respecto a la utilización, el retardo y el *throughput.*

- **Utilización**

En la figura 5.8 se muestra la utilización del canal en dos gráficas, una muestra la utilización con respecto a los *slots* que ya fueron reservados para transmitir, y la otra la utilización con respecto a los *slots* que ya fueron utilizados para transmitir.

Ahora empezamos a simular desde una carga ofrecida de 3 Mbps, obteniendo una utilización del canal de un 30 % aproximadamente. Si analizamos para una carga ofrecida de 7 Mbps obtenemos una utilización del 80 % aproximadamente, si este valor lo comparamos con la utilización obtenida cuando el usario transmitía paquetes que viajaba a un salto, como se mostró en la figura 5.3, vemos que aumentó al doble la utilización del canal para este caso, y es debido a que se tiene que retransmitir la misma cantidad de paquetes para que lleguen a su destino. La utilizacón del canal se logró hasta un 88 %, a partir de la carga ofrecida de 8 Mbps hasta 12 Mbps.

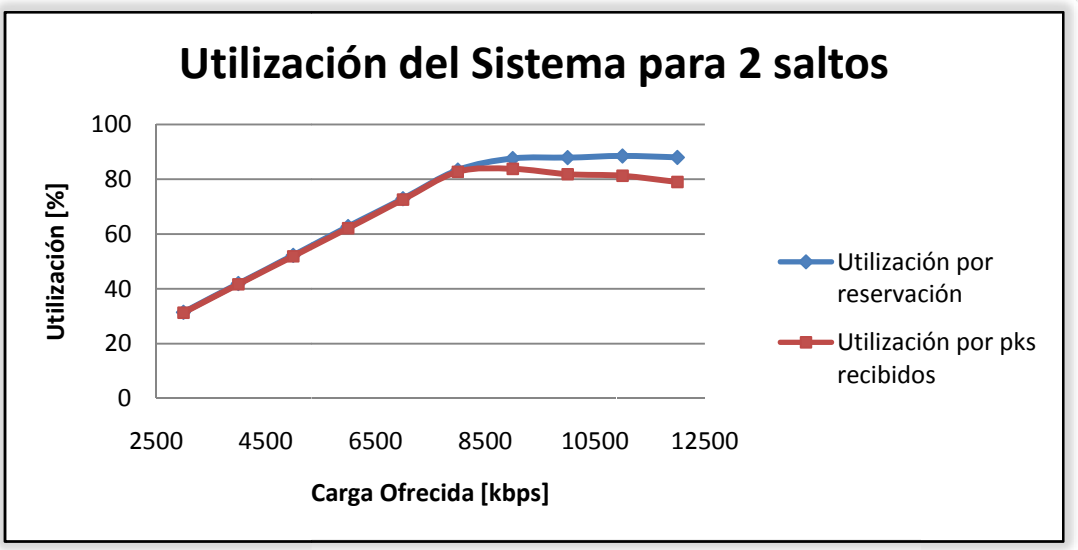


Figura 5.8: Utilización del sistema con un usuario transmitiendo a dos saltos.

- **Retardo**

En la gráfica 5.9 podemos observar la forma en que el retardo se comporta cuando se aumenta la carga ofrecida, cuando se tienen que retransmitir los paquetes de datos. Para una carga ofrecida de 3 Mbps se tiene un retardo de 83 ms e irá aumentando a medida que la carga ofrecida sea mayor hasta lleagar a los 130 ms para una carga ofrecida de 8 Mbps. A partir de este punto el retardo aumentará de manera exponencial hasta llegar a un 1.5 segundos con una carga ofrecida de 12 Mbps. Uno de los factores que hace que los retardos sean altos es debido a que se tienen que retransmitir los paquetes. Con los resultados obtenidos podemos concluir que solo las aplicaciones que no son sensibles al retardo pueden utilizarse en este modelo.

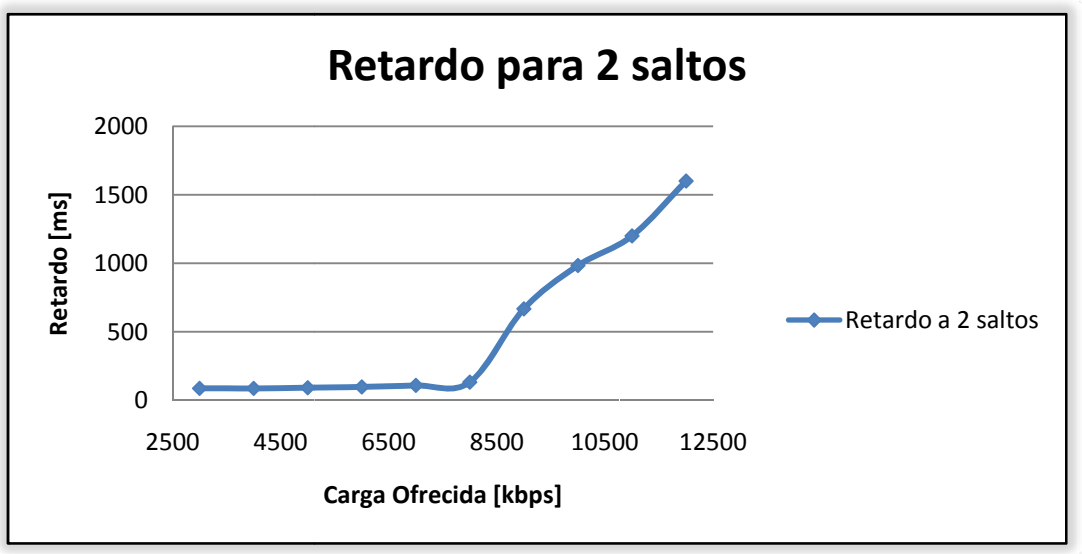


Figura 5.9: Retardo del sistema con un usuario transmitiendo a dos saltos.

- *Throughput*

En la figura 5.10 se muestra el *throughput* para diferentes cargas ofrecidas con un usuario transmitiendo datos que viajan a su destino en dos saltos, se observa que el *throughput* crece linealmente hasta una carga ofrecida de 8 Mbps aproximadamente, esto significa que el tiempo que tarda en recibirse el paquete es óptima, despues permanece constante hasta los 12 Mbps, esto significa que el máximo *throughput* que podemos obtener es de 8 Mbps, comparado con el *throughput* máximo que se alcanzó en el primer escenario que fue de 16 Mbps, vemos que se redujo a la mitad, esto se debe a que el destino ahora se encuentra a dos saltos del nodo transmisor y por lo tanto los paquetes se tienen que retransmitir para que lleguen a su destino final.

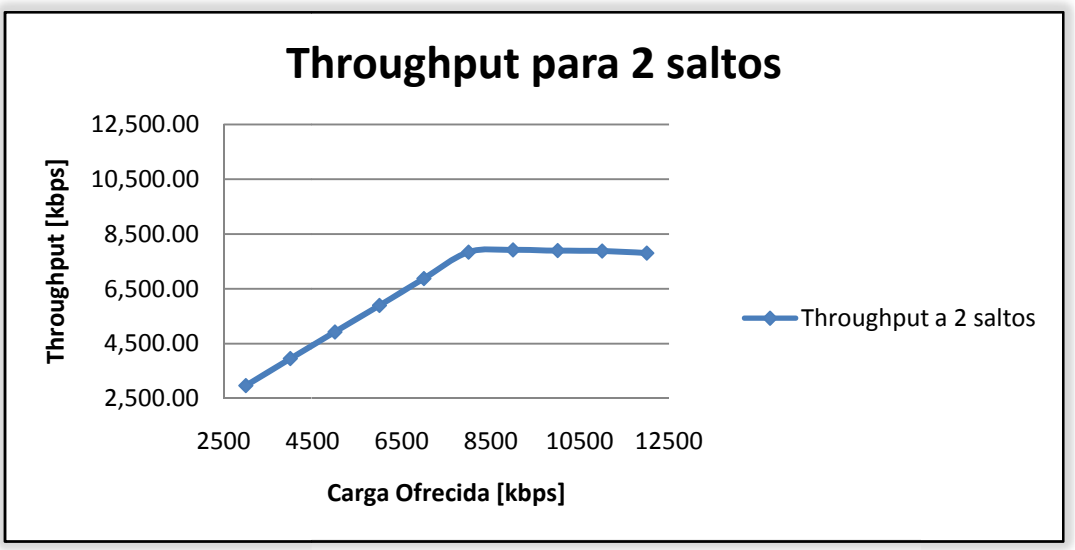


Figura 5.10: *Throughput* del sistema con un usuario transmitiendo a dos saltos.

Escenario 3:

En este escenario se visualizará el comportamiento que tiene la red cuando un usuario quiere transmitirle datos a otro que se encuentra a tres saltos, con la finalidad de observar los cambios que presentarán tanto en el *throughput* como en el retardo. Los paquetes también serán de 1460 bytes, como se observa en la figura 5.11.

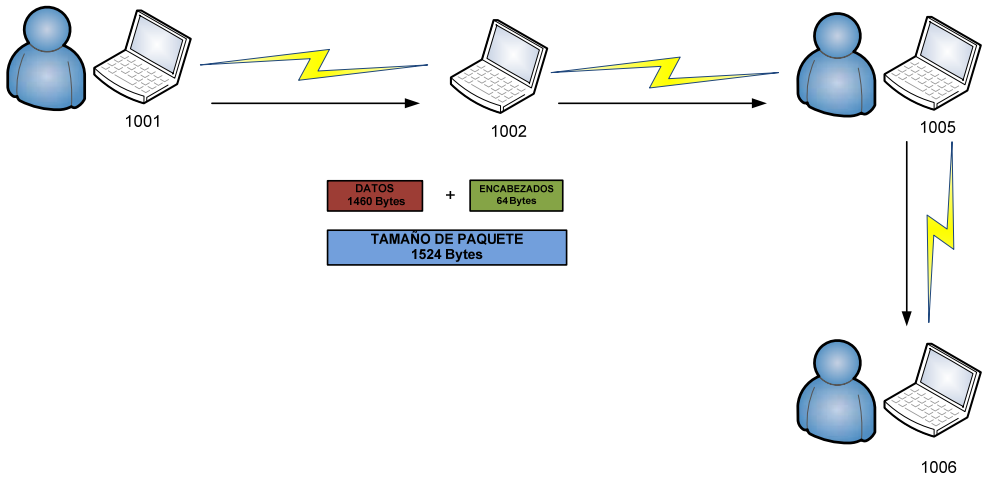


Figura 5.11: Un usuario transmitiendo datos a tres saltos.

- **Utilización**

La máxima utilización del canal, cuando los paquetes se tienen que retransmitir dos veces es cuando la carga ofrecida es de 6 Mbps, alcanzando el 88% aproximadamente, como se aprecia en la figura 5.12.

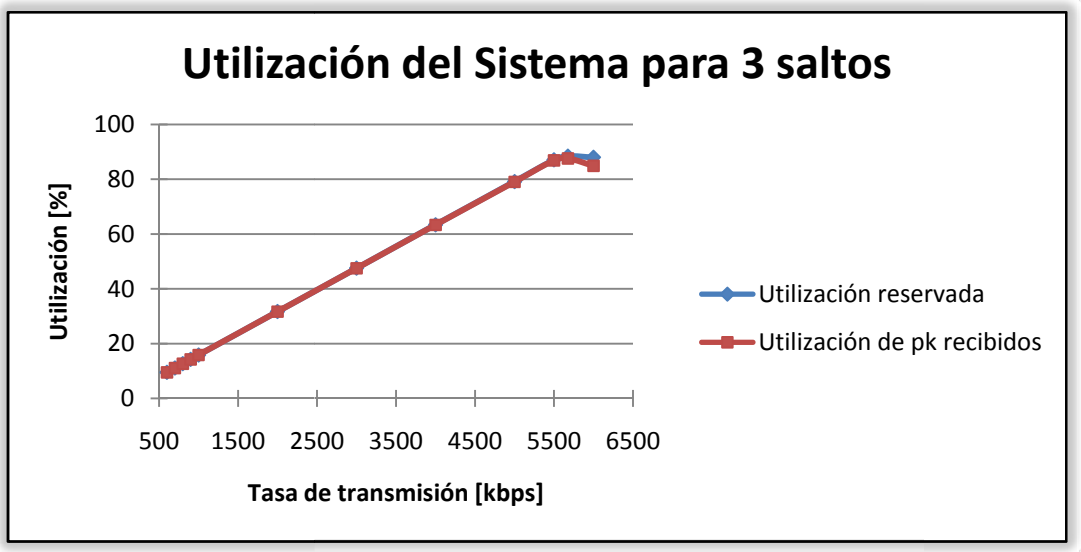


Figura 5.12: Utilización del sistema con un usuario transmitiendo a tres saltos.

- **Retardo**

Para este escenario se observa que el retardo crece paulatinamente, empezando con un valor de 109 ms para una carga ofrecida de 600 kbps hasta los 172 ms para una carga ofrecida de 5 Mbps, apartir de aquí el retardo crece demasiado llegando a un segundo de retardo como se aprecia en la figura 5.13.

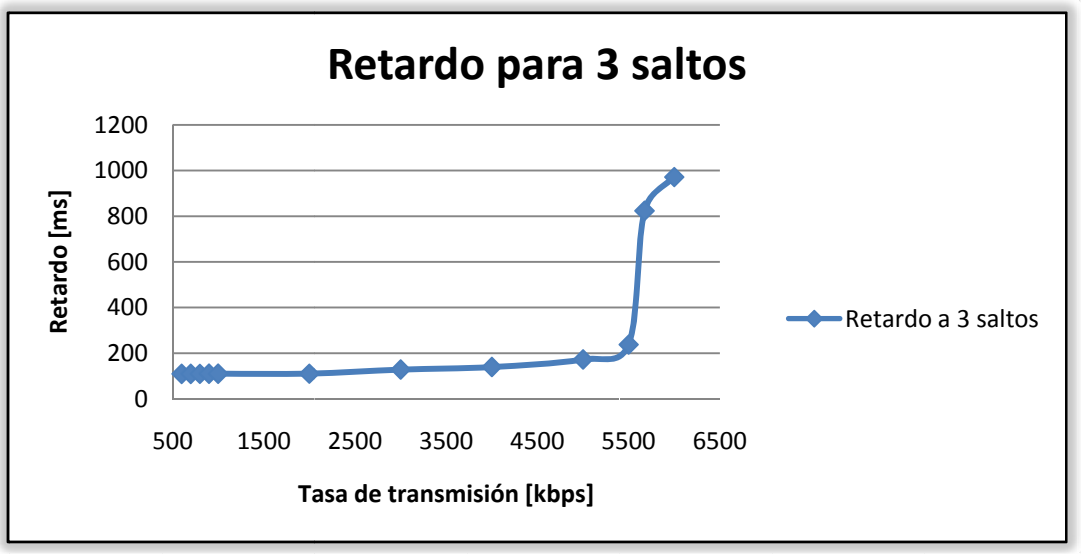


Figura 5.13: Retardo del sistema con un usuario transmitiendo a tres saltos.

- *Throughput*

Este escenario nos muestra que el *throughput* máximo es de 5.3 Mbps aproximadamente, para una carga ofrecida de 6 Mbps, si comparamos este valor con el obtenido cuando los paquetes viajan a través de 2 saltos, vemos como disminuyó considerablemente. Por lo tanto podemos concluir que entre mayor sea el número de saltos que el paquete tenga que atravesar para llegar a su destino, el *throughput* disminuya significativamente, como se puede aprecia en la figura 5.14.

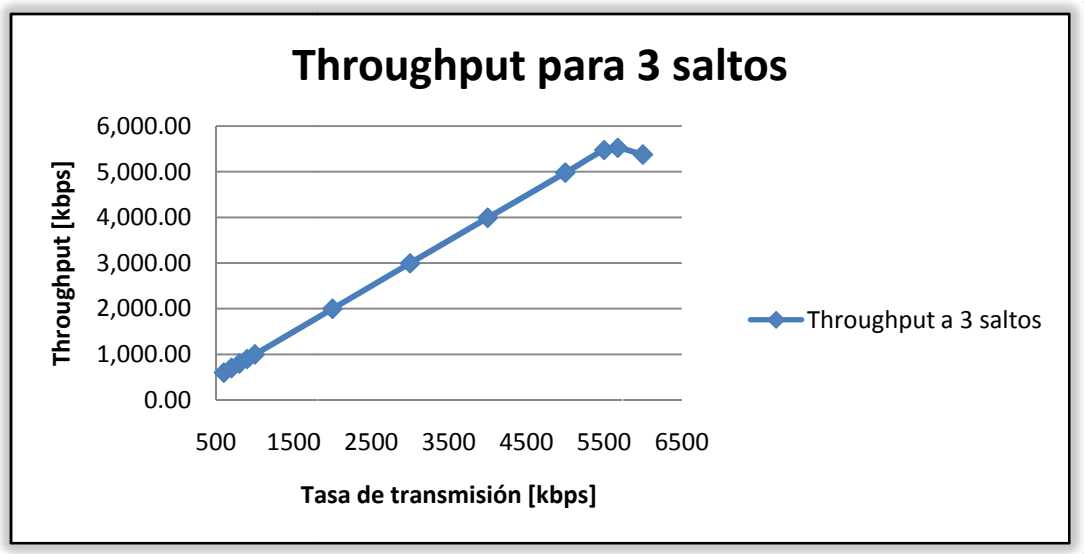


Figura 5.14: *Throughput* del sistema con un usuario transmitiendo a tres saltos.

Escenario 4:

Este último escenario pretende mostrar el comportamiento en condiciones más reales de una red tipo *mesh.* Donde los seis usuarios van a transmitir tráfico, cada uno a diferente destino haciendo que los flujos viajen a uno, dos o a tres saltos, dependiendo del nodo con el que desean comunicarse, como se muestra en la figura 5.15.

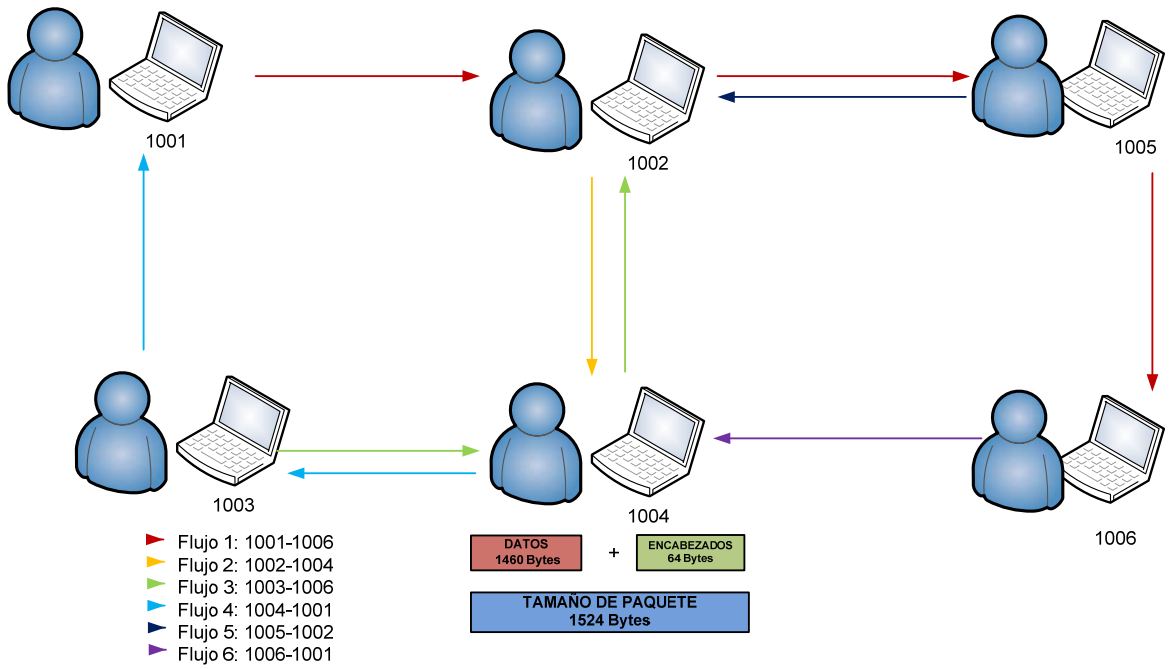


Figura 5.15: Los seis usuarios transmitiendo tráfico al mismo tiempo.

El tamaño de los paquetes es el mismo para todos los usuarios y es de 1460 bytes. Otra consideración que se hizo para este escenario es el tamaño de la cola que almacena los paquetes que vienen desde la capa de aplicación a la capa MAC, y la cola que se utiliza para almacenar los paquetes que van a ser retransmitidos. En los escenarios anteriores se manejo que las colas fueran infinitas, ya que se mostraba el comportamiento ideal de la red, sin embargo en la realidad el tamaño de las colas no es infinito, por lo tanto para este caso se almacenarán un máximo de 100 paquetes en cada cola.

En la tabla 5.2 se muestran las trayectorias que los paquetes realizarán, para llegar a su destino final.

No. De Saltos	Nodo Transmisor	Nodo Re-transmisor	Nodo Re-transmisor	Nodo Receptor
3	1001	1002	1005	1006
1	1002	-	-	1004
2	1003	1004	-	1002
2	1004	1003	-	1001
1	1005	-	-	1002
1	1006	-	-	1004

Tabla 5.2: Direccionamiento de tráfico.

- **Utilización**

El primer parámetro que vamos a analizar es la utilización del canal para este escenario. En la figura 5.16 se muestra la utilización que se obtuvo para los tres usuarios que transmitieron paquetes a su destino y que se encontraban a un salto de distancia, la máxima utilización fue del 25.73 % con una carga ofrecida de 2.2 Mbps. También se aprecia que la máxima utilización para los dos usuarios que transmitieron paquetes de datos a un destino y que se encontraban a dos saltos de distancia de ellos fue del 32.8 % Por otro lado para el único usuario que transmitía paquetes que viajaban a través de tres saltos para llegar a su destino, su utilización fue del 25.3 % como máximo. Finalmente la utilización general, tomando en cuenta a todos los usuarios fue del 88 %, logrando que se utilizara al máximo la parte del canal que estaba designada para la transmisión de datos.

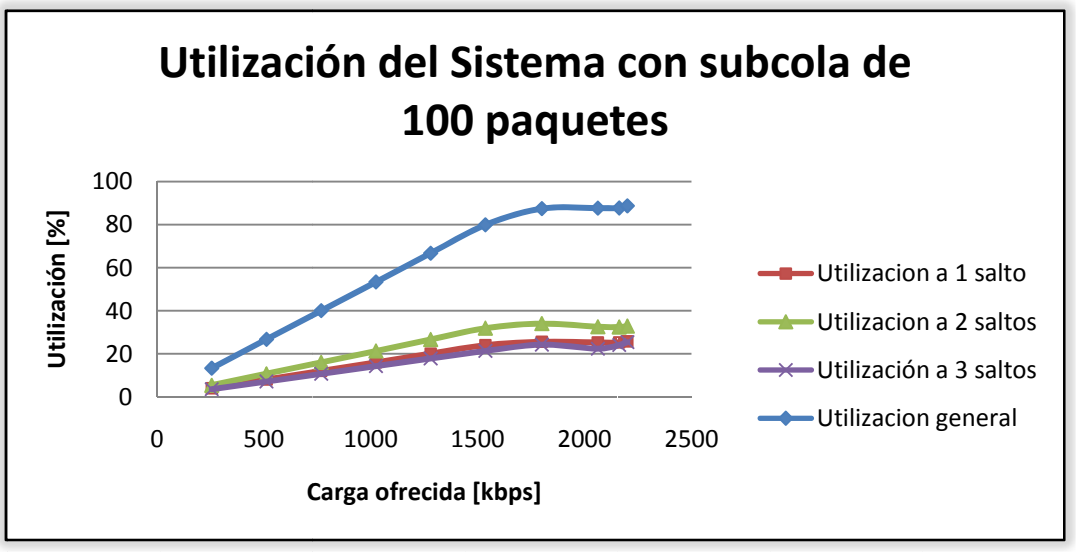


Figura 5.16: Utilización del sistema con todos los usuarios transmitiendo.

- **Retardo**

El análisis del comportamiento del retardo para una red es muy importante, ya que con éste podemos garantizar que tipo de aplicaciones se pueden correr en el modelo propuesto. En la figura 5.17 se expone cómo se comporta el retardo a diferentes cargas ofrecidas de acuerdo a número de saltos que tienen que pasar los paquetes para llegar a su destino.

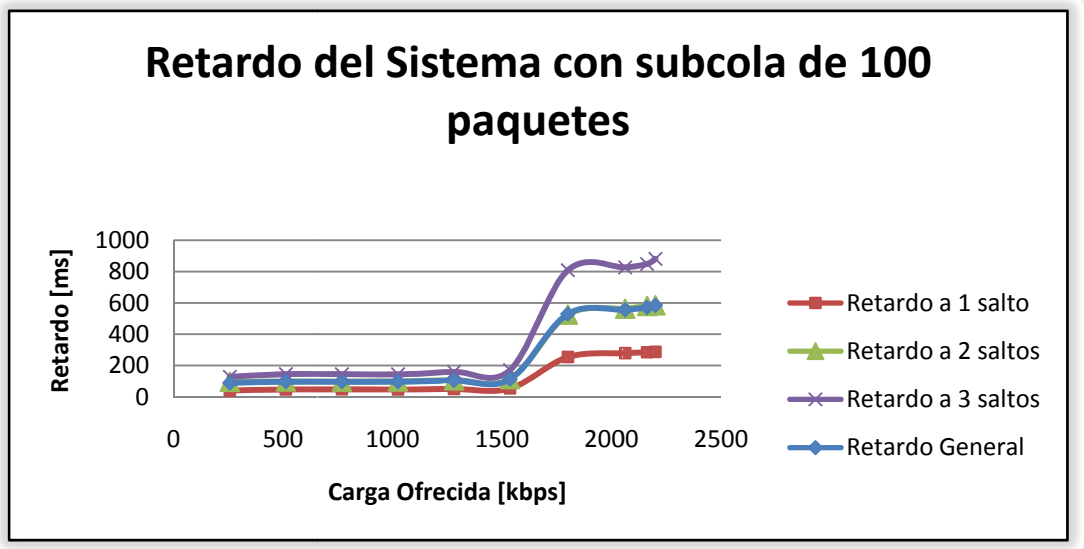


Figura 5.17: Retardo del sistema con todos los usuarios transmitiendo.

Para los usuarios que transmitieron a un salto de distancia, el retardo mínimo fue de 39.8 ms a una carga ofrecida de 256 kbps, este retardo aumentó hasta los 53.7 ms con una carga de 1.5 Mbps. A partir de aquí al aumentar la carga, el retardo creció exponencialmente hasta los 287 ms con una carga de 2.2 Mbps.

Para los usuarios que transmitían a dos saltos de distancia, el retardo mínimo fue de 96.2 ms para una carga ofrecida de 256 kbps, va creciendo gradualmente hasta 114 ms con una carga de 1.5 Mbps; alcanzando un retardo máximo de 585 ms con 2.2 Mbps de carga ofrecida.

El usuario que transmitió a tres saltos el retardo mínimo fue de 128.3 ms, teniendo 171 ms para la carga 1.5 Mbs y llegando a 880 ms para los 2.2 Mbps.

Por lo tanto, de manera general se concluye que el modelo propuesto alcanza el menor retardo promedio con la carga ofrecida de 256 kbps siendo éste de 88 ms, si se aumenta la carga ofrecida el retardo promedio crecerá, más sin embargo todavía puede ser aceptable con la carga de 1.5 Mbps, teniendo el valor de 113 ms, de aquí en adelante crecerá de manera exponencial hasta llegar a los 584 ms para 2.2 Mbps.

Con los resultados obtenidos en este escenario, ya podemos deducir que por el momento solo aplicaciones que nos son susceptibles al retardo pueden utilizarse, tales como el envío de correo electrónico, la transferencia de archivos y el acceso a la *web.*

Otro factor importante que es necesario conocer es el *throughput* máximo que puede ofrecer este sistema y a partir de que carga ofrecida se obtiene, en la figura 5.18 se muestra como fue aumentando el *throughput* hasta llegar al máximo que se puede ofrecer que es de 9.5 Mbps cuando los seis usuarios transmiten paquetes a una carga ofrecida de 2 Mbps cada uno, sin embargo los retardos que presenta con esta carga ofrecida es alrededor de los 500 ms. Sin embargo, si queremos que el retardo sea menor, pero el *throughput* siga siendo alto, los usuarios pueden utilizar una carga ofrecida de 1.5 Mbps cada y alcanzar un *throughput* de 9 Mbps. Con un retardo promedio de 113 ms.

- *Throughput*

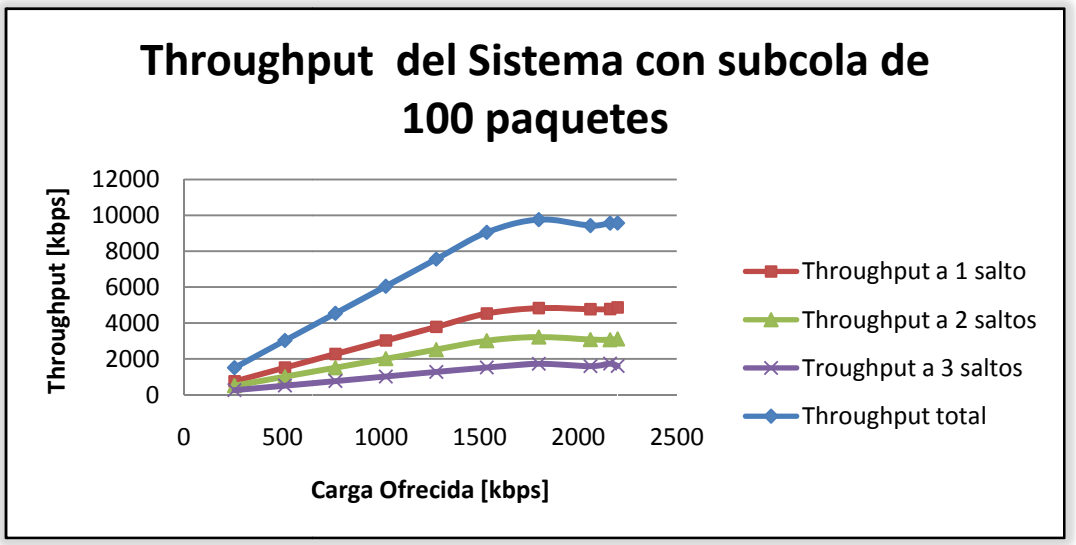


Figura 5.18: *Throughput* del sistema con todos los usuarios transmitiendo.

Capítulo 6.

Conclusiones

En este trabajo de tesis se muestra el diseño e implementación de un algoritmo de calendarización de recursos, para redes WiMAX *mesh,* que trabajan en forma distribuida coordinada. Este modelo se implementó en el software OPNET Modeler v.8, y demostró que funciona correctamente, ya que al compararlo con el modelo teórico que se realizó, tuvo un 99.63 % de exactitud.

Para el diseño propuesto, se consideró el proceso de tres vías que llevan a cabo, cada una de las estaciones suscriptoras para solicitar recursos, así mismo se utilizó un esquema FIFO para la asignación de recursos. Este proceso también se sincronizó con el algoritmo de elección para que los nodos compitieran por el canal y así lograr un comportamiento semejante al de una red real. Otro factor que se tomó en cuenta fue el enrutamiento, este se llevó a cabo por medio del algoritmo de Dijkstra, para obtener las rutas que los paquetes tienen que seguir, para llegar a su destino final.

Finalmente se realizó el análisis del comportamiento dinámico de las redes *mesh,* y se pudo conocer que la máxima capacidad de un canal de 25 Mhz es de 16.8348 Mbps para la transferencia de datos. Por otro lado también apreciamos que entre mayor sea el número de saltos que los paquetes tienen que realizar para llegar a su destino, disminuye de manera significativa el *throughput.*

Con el último escenario que se simuló, se quiere mostrar el comportamiento que tiene la red, cuando todos los usuarios están transmitiendo datos al mismo tiempo, y así analizar que ocurre con el *throughput* y con el retardo principalmente. Cuando cada uno de los usuarios generaron datos a una carga ofrecida de 256 kbps, el retardo promedio fue de 88ms con un *throughput* total de 1.5 Mbps. Al ir aumentando la carga ofrecida hasta 1.5 Mbps en la generación de paquetes, el valor del retardo fue de 113 ms con un *throughput* total de 9Mbps. Sin embargo para cargas ofrecidas mayores a 1.5 Mbp el retardo ya es demasiado grande, alcanzando los 584 ms para una carga de 2.2 Mbps y obteniendo un

thorughput total de 9.5 Mbps. Con estos resultados podemos concluir que sólo aplicaciones que no son sensibles al retardo tales como la transferencia de archivos, el envío de correo electrónico o el acceso a la web pueden ser utilizadas.

Este trabajo de investigación contribuye con um modelo de simulación para redes WiMAX *mesh,* donde es posible analizar el comportamiento dinámico de este tipo de redes y asi mejorar su rendimiento. Con este modelo se puede visualizar como se afecta el retardo y el *throughput* al aumentar la carga ofrecida para diferentes escenarios. Este estudio permitirá de manera más sencilla encontrar la mejor forma de optimizar este tipo de redes, ya que el funcionamiento es complejo.

El modelo propuesto se puede utilizar para trabajos futuros. Es importante mencionar que se va a tomar como base para un trabajo de doctorado, donde la red a analizar será de 50 nodos. Además de que se puede complementar para que soporte calidad de servicio.

.

Bibliografía

1. Ohrtman, Frank. **"WiMAX HANDBOOK"**. McGraw-Hill.2005.

2. Tang Yee Seok, Muller Peter, Sharif Hamid R.**"WiMAX SECURITY AND QUALITY OF SERVICE An End-to-end Persective"**. Wiley.2011.

3. IEEE 802.16-2004, "**IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems**". Octubre 2004.

4. Loutfi Nuaymi. "**WiMAX TECHNOLOGY FOR BROADBAND WIRELESS ACCESS".** John Wiley & Sons Ltd,. England.2007.

5. Sauter Martin. **"Comunication Systems for the Mobile Information Society**". John Wiley & Sons Ltd. England. 2006.

6. Oz Effy. **" MANAGEMENT INFORMATION SYSTEMAS"**. THOMSON. United States. 2009.

7. Pentikoussis Kostas, Blume Oliver, Calvo Aguero Ramón, Papavassilious Symeon. **"Mobile Networks and Management"**.First International Conference, MONAMI 2009.

8. España Boquera María del Carmen.**" SERVICIOS AVANZADOS DE TELECOMUNICACIÓN"**. Diaz de Santos. España. 2003.

9. Ahson Syed, Ilyas Mohammad. "**WiMAX Standards and Security**". CRC Press. 2008.

10. Wang Bin, Jin Zhigang. **"A Fair Centralized Scheduling Algorithm Based on Traffic Demand for IEEE 802.16 Mesh Networks".** IEEE. 2010.

11. Zhang Ming, Wang Suoping, He Tao. **"Study on Coordinated Distributed Scheduling in WiMAX Mesh Network".** IEEE. 2009.

12. Peng Limin, Sun Suyun. **"Coordinated Distributed Data Scheduling Scheme in IEEE 802.16 Mesh Networks"**. International Conference on Internet Computing and Information Services. IEEE. 2011.

13. Wang Shie-Yuan, Lin Chih-Che, Fang Ku-Han. **"Improving the Data Scheduling Efficiency of the IEEE 802.16(d) Mesh Network".** IEEE 2008.

14. Zhengbing Zhang, Liang Yu.**"Improved fair scheduling mechanism in distributed WiMAX Mesh Networks"**. IEEE. 2010.

15. Awadh Al Shukaili1, Naveen Chilamkurti1 and Sherali Zeadally . **"Enabling "Quality of Service" in IEEE802.16 Networks for Distributed Mesh Topologies"**. IEEE.2010.

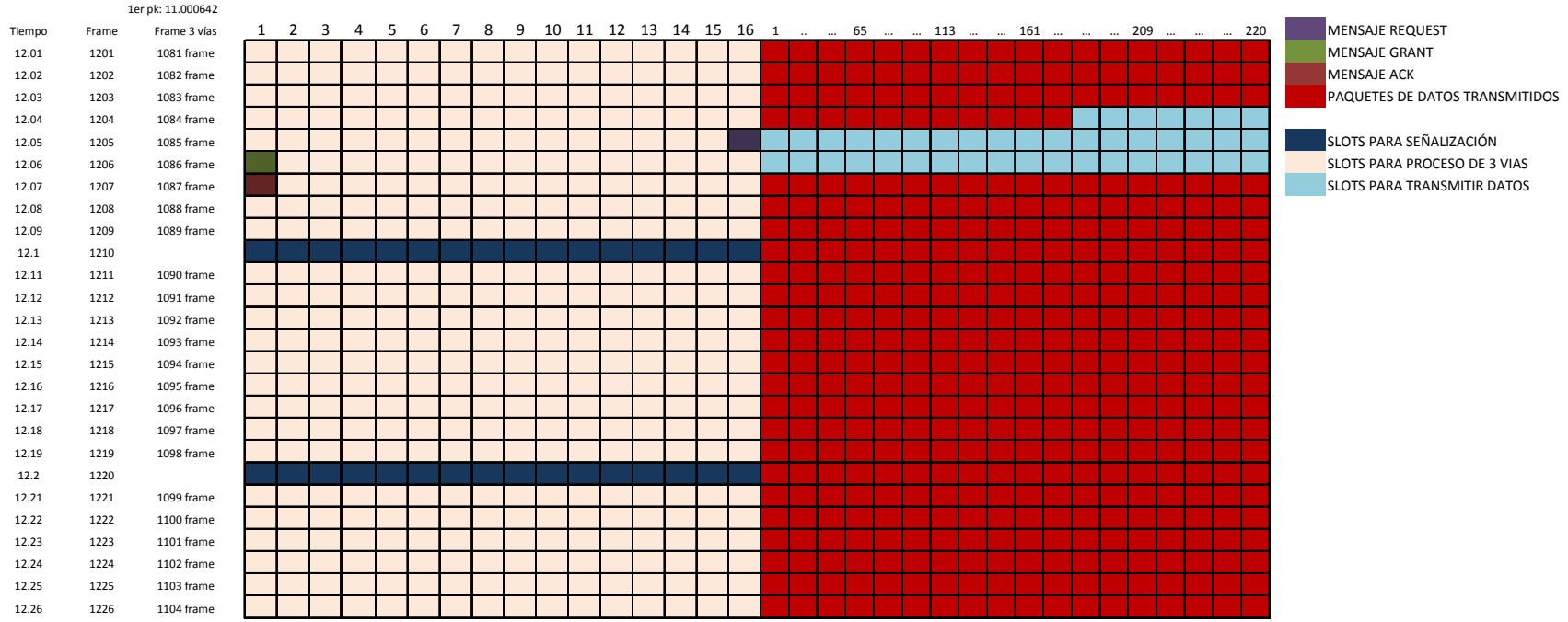
16. Da Teng Shoubao Yang, Weiqing He, Yun Hu. **" TEOS: A Throughput-Efficiency Optimal Distributed Data Subframe Scheduling Scheme in Mesh Networks".** IEEE.2008.

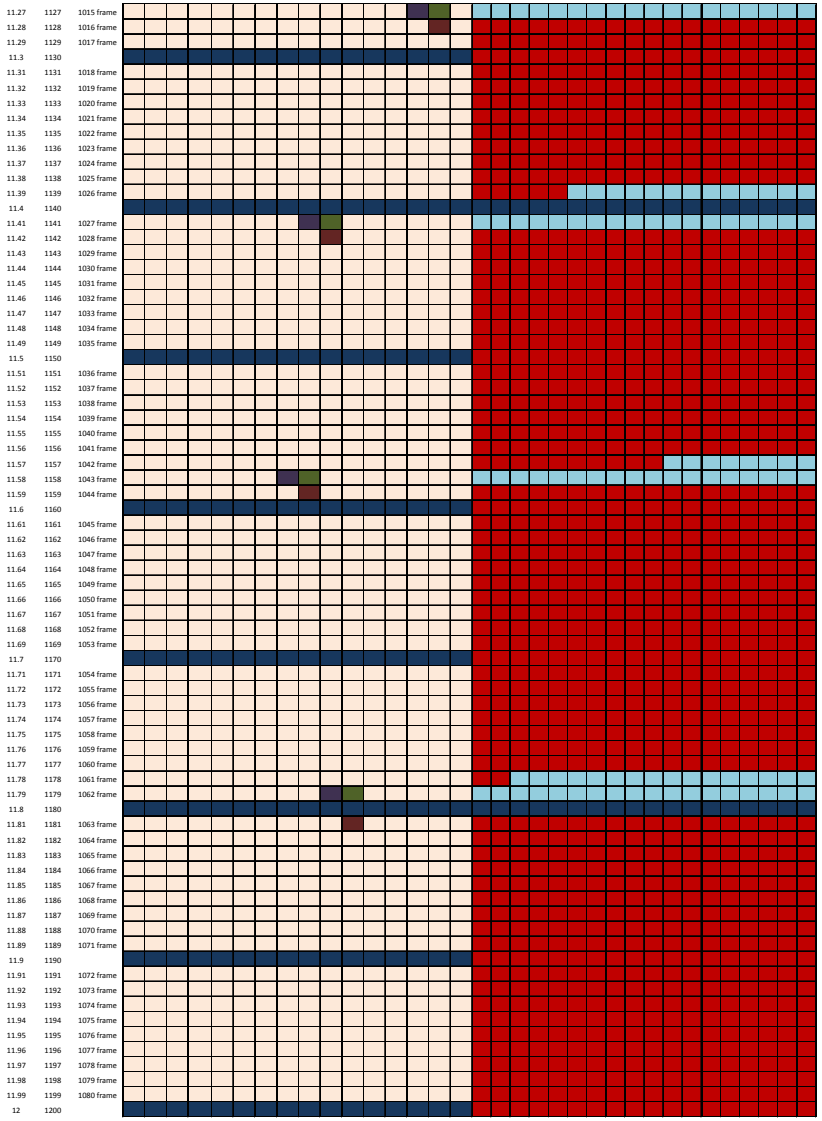
Apéndice

Apéndice A. Lista de acrónimos

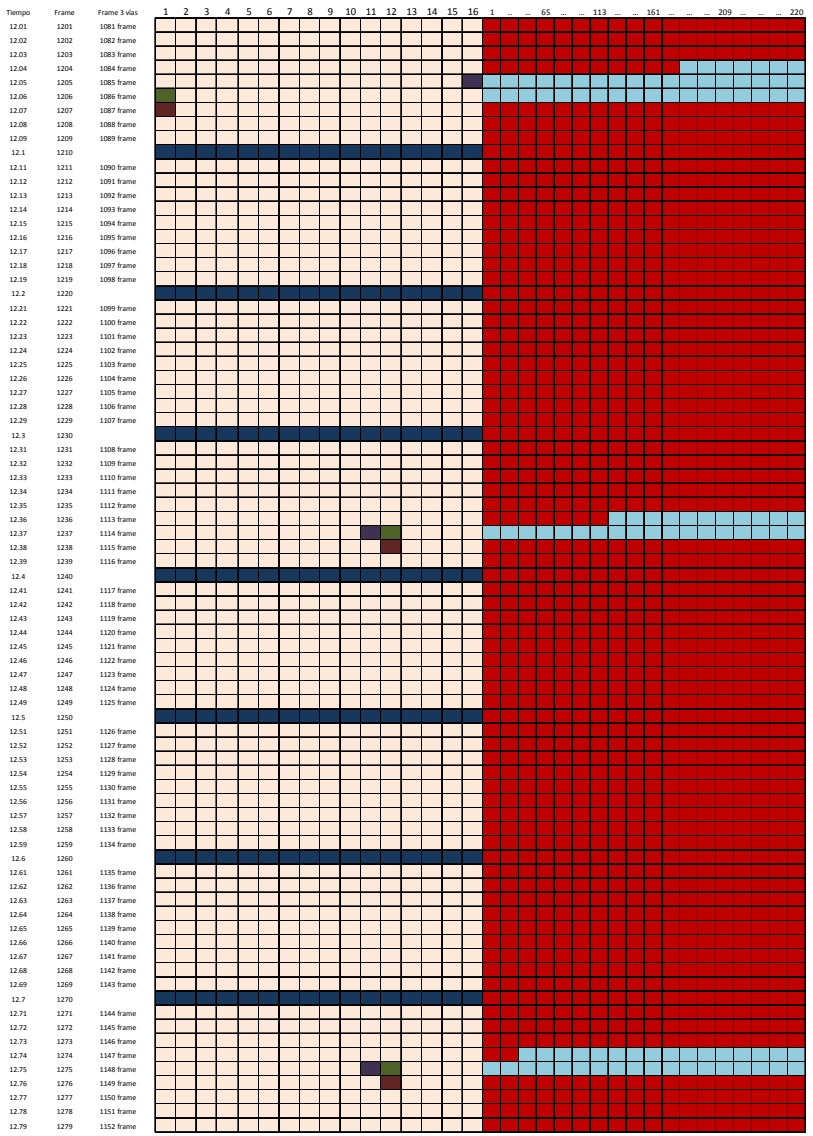
AP: Access Point
BS: Base Station
CDMA: Code Division Multiple Access
CPE: Customer Premises Equipment
Davic : Digital Audio Video Council
DSL: Digital Suscriber Line
DVB: Digital Video Broadcasting
ETSI: European Telecommunications Standards Institute
FFT: Fast Furier Transform
FIFO: First In, First Out
GSM: Global System for Mobile Communications
IP*:* Internet Protocol
LAN: Local Area Network
LMDS: Local Multipoint Distribution Service
MAC: Media Access Control
MBWA: Mobile Broadband Wireless Access
MIMO: Multiple Input Multiple Output
MMDS: Multichannel Multipoint Distribution Service
OFDM: Orthogonal Frecuency Division Multiplexing
PMP: Point to Multipoint
QAM: Quadrature Amplitud Modulation
QPSK: Quadrature Phase Shift Keying
SMS: Short Message Service
SS: Suscriber Station
TDD: Time Division Duplex
UMTS: Universal Mobile Telecommunication System
WAN: Wide Area Network
WiMAX : Worldwide Interoperability for Microwave Access
WMAN: Wireless Metropilitan Acces Network

Apéndice B. Diagrama de envío de paquetes

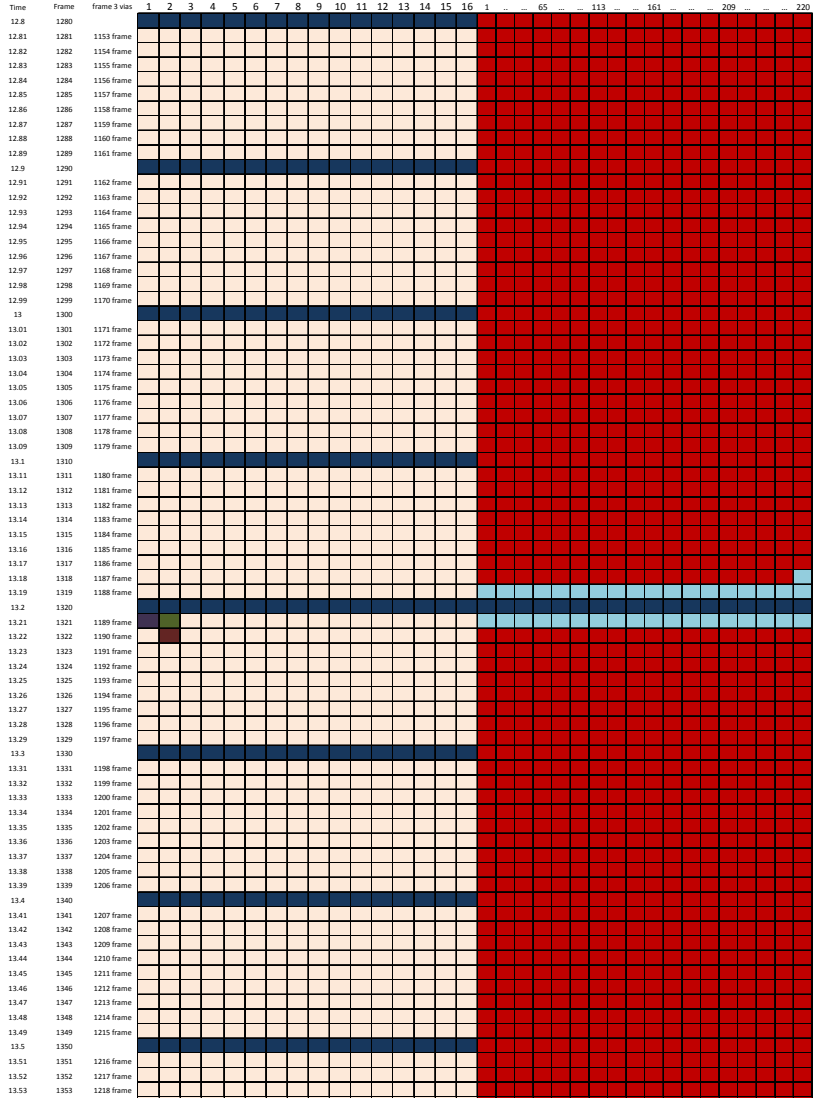


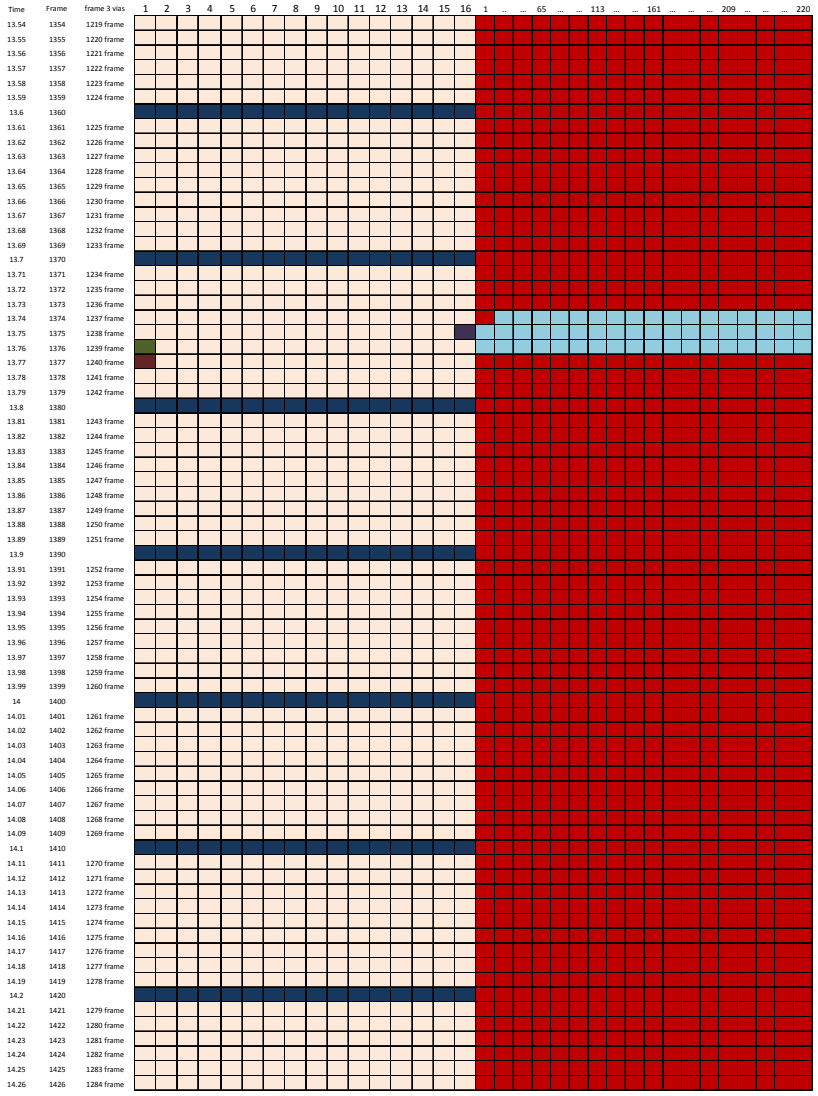


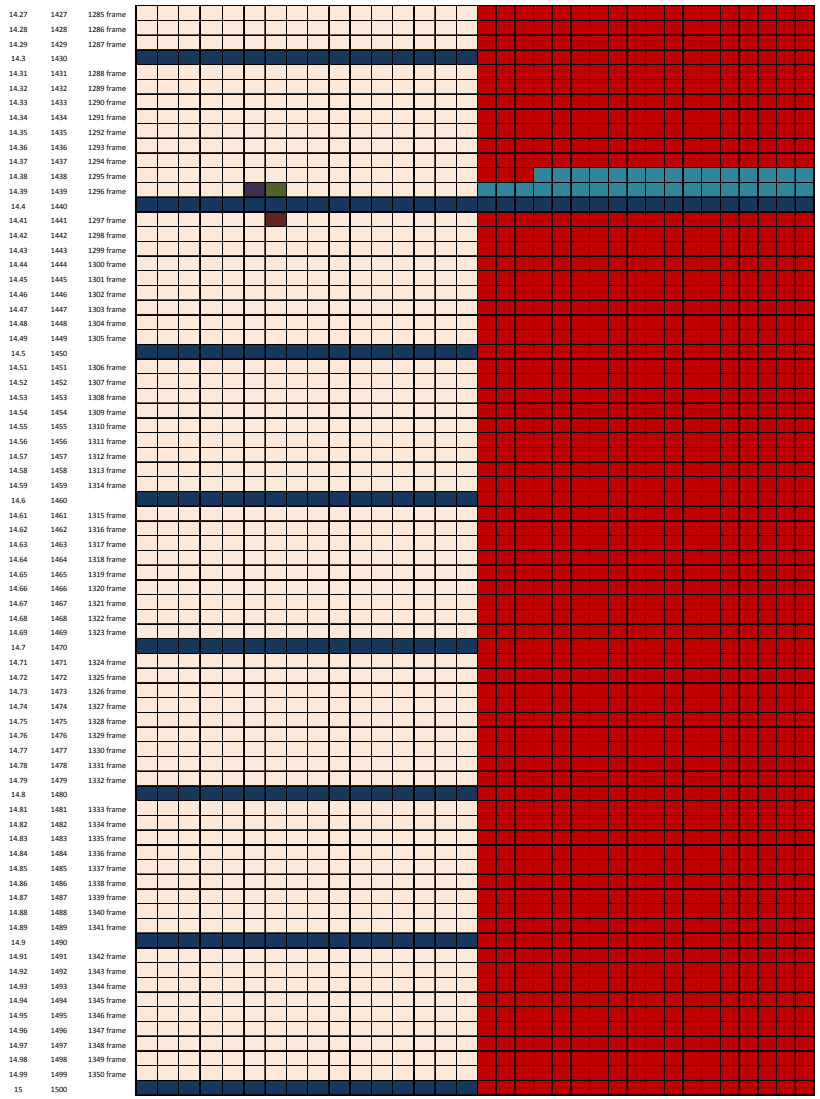
11.27	1127	1015 frame
11.28	1128	1016 frame
11.29	1129	1017 frame
11.3	1130	
11.31	1131	1018 frame
11.32	1132	1019 frame
11.33	1133	1020 frame
11.34	1134	1021 frame
11.35	1135	1022 frame
11.36	1136	1023 frame
11.37	1137	1024 frame
11.38	1138	1025 frame
11.39	1139	1026 frame
11.4	1140	
11.41	1141	1027 frame
11.42	1142	1028 frame
11.43	1143	1029 frame
11.44	1144	1030 frame
11.45	1145	1031 frame
11.46	1146	1032 frame
11.47	1147	1033 frame
11.48	1148	1034 frame
11.49	1149	1035 frame
11.5	1150	
11.51	1151	1036 frame
11.52	1152	1037 frame
11.53	1153	1038 frame
11.54	1154	1039 frame
11.55	1155	1040 frame
11.56	1156	1041 frame
11.57	1157	1042 frame
11.58	1158	1043 frame
11.59	1159	1044 frame
11.6	1160	
11.61	1161	1045 frame
11.62	1162	1046 frame
11.63	1163	1047 frame
11.64	1164	1048 frame
11.65	1165	1049 frame
11.66	1166	1050 frame
11.67	1167	1051 frame
11.68	1168	1052 frame
11.69	1169	1053 frame
11.7	1170	
11.71	1171	1054 frame
11.72	1172	1055 frame
11.73	1173	1056 frame
11.74	1174	1057 frame
11.75	1175	1058 frame
11.76	1176	1059 frame
11.77	1177	1060 frame
11.78	1178	1061 frame
11.79	1179	1062 frame
11.8	1180	
11.81	1181	1063 frame
11.82	1182	1064 frame
11.83	1183	1065 frame
11.84	1184	1066 frame
11.85	1185	1067 frame
11.86	1186	1068 frame
11.87	1187	1069 frame
11.88	1188	1070 frame
11.89	1189	1071 frame
11.9	1190	
11.91	1191	1072 frame
11.92	1192	1073 frame
11.93	1193	1074 frame
11.94	1194	1075 frame
11.95	1195	1076 frame
11.96	1196	1077 frame
11.97	1197	1078 frame
11.98	1198	1079 frame
11.99	1199	1080 frame
12	1200	

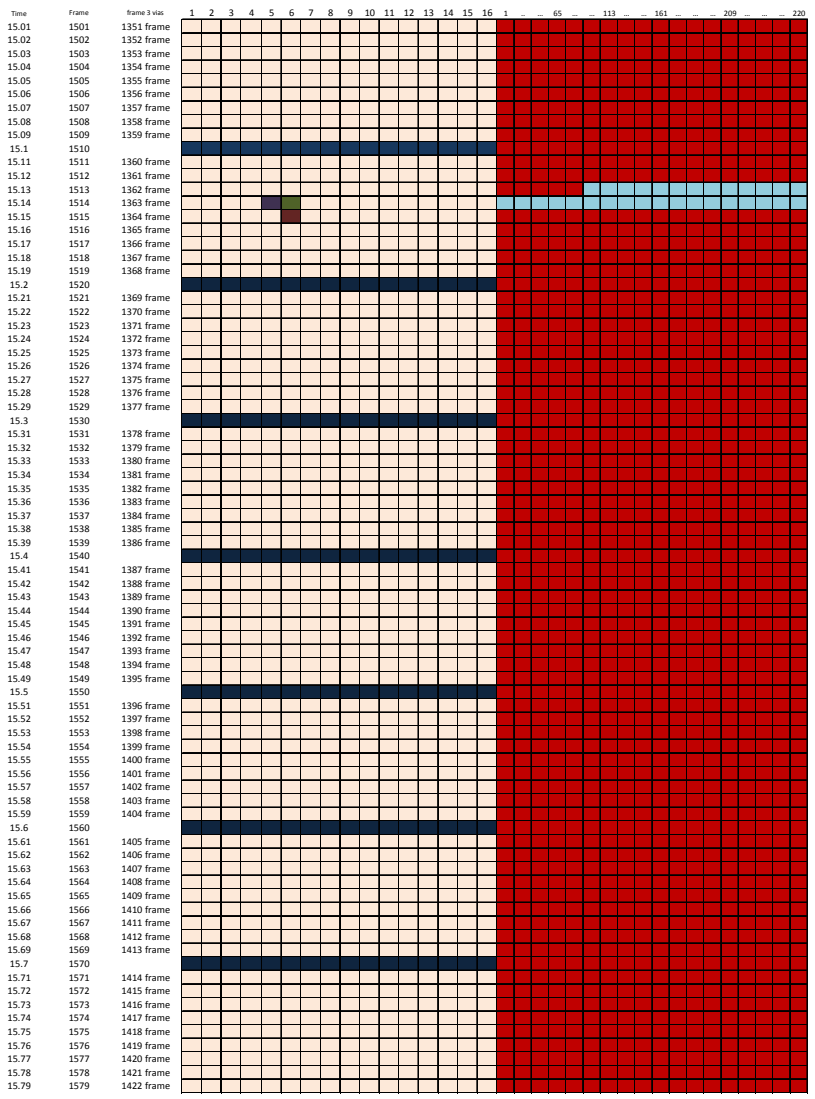


Tiempo	Frame	Frame 3 vías
12.01	1201	1081 frame
12.02	1202	1082 frame
12.03	1203	1083 frame
12.04	1204	1084 frame
12.05	1205	1085 frame
12.06	1206	1086 frame
12.07	1207	1087 frame
12.08	1208	1088 frame
12.09	1209	1089 frame
12.1	1210	
12.11	1211	1090 frame
12.12	1212	1091 frame
12.13	1213	1092 frame
12.14	1214	1093 frame
12.15	1215	1094 frame
12.16	1216	1095 frame
12.17	1217	1096 frame
12.18	1218	1097 frame
12.19	1219	1098 frame
12.2	1220	
12.21	1221	1099 frame
12.22	1222	1100 frame
12.23	1223	1101 frame
12.24	1224	1102 frame
12.25	1225	1103 frame
12.26	1226	1104 frame
12.27	1227	1105 frame
12.28	1228	1106 frame
12.29	1229	1107 frame
12.3	1230	
12.31	1231	1108 frame
12.32	1232	1109 frame
12.33	1233	1110 frame
12.34	1234	1111 frame
12.35	1235	1112 frame
12.36	1236	1113 frame
12.37	1237	1114 frame
12.38	1238	1115 frame
12.39	1239	1116 frame
12.4	1240	
12.41	1241	1117 frame
12.42	1242	1118 frame
12.43	1243	1119 frame
12.44	1244	1120 frame
12.45	1245	1121 frame
12.46	1246	1122 frame
12.47	1247	1123 frame
12.48	1248	1124 frame
12.49	1249	1125 frame
12.5	1250	
12.51	1251	1126 frame
12.52	1252	1127 frame
12.53	1253	1128 frame
12.54	1254	1129 frame
12.55	1255	1130 frame
12.56	1256	1131 frame
12.57	1257	1132 frame
12.58	1258	1133 frame
12.59	1259	1134 frame
12.6	1260	
12.61	1261	1135 frame
12.62	1262	1136 frame
12.63	1263	1137 frame
12.64	1264	1138 frame
12.65	1265	1139 frame
12.66	1266	1140 frame
12.67	1267	1141 frame
12.68	1268	1142 frame
12.69	1269	1143 frame
12.7	1270	
12.71	1271	1144 frame
12.72	1272	1145 frame
12.73	1273	1146 frame
12.74	1274	1147 frame
12.75	1275	1148 frame
12.76	1276	1149 frame
12.77	1277	1150 frame
12.78	1278	1151 frame
12.79	1279	1152 frame

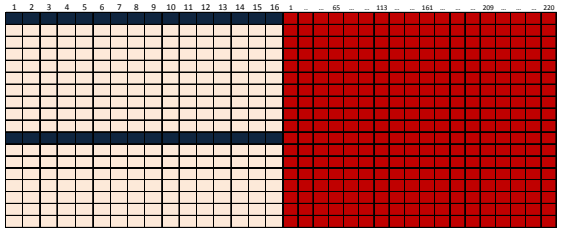

Time Frame frame 3 vias 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 1 ... ... 65 ... ... 113 ... ... 161 ... ... 209 220
12.8 1280
12.81 1281 1153 frame
12.82 1282 1154 frame
12.83 1283 1155 frame
12.84 1284 1156 frame
12.85 1285 1157 frame
12.86 1286 1158 frame
12.87 1287 1159 frame
12.88 1288 1160 frame
12.89 1289 1161 frame
12.9 1290
12.91 1291 1162 frame
12.92 1292 1163 frame
12.93 1293 1164 frame
12.94 1294 1165 frame
12.95 1295 1166 frame
12.96 1296 1167 frame
12.97 1297 1168 frame
12.98 1298 1169 frame
12.99 1299 1170 frame
13 1300
13.01 1301 1171 frame
13.02 1302 1172 frame
13.03 1303 1173 frame
13.04 1304 1174 frame
13.05 1305 1175 frame
13.06 1306 1176 frame
13.07 1307 1177 frame
13.08 1308 1178 frame
13.09 1309 1179 frame
13.1 1310
13.11 1311 1180 frame
13.12 1312 1181 frame
13.13 1313 1182 frame
13.14 1314 1183 frame
13.15 1315 1184 frame
13.16 1316 1185 frame
13.17 1317 1186 frame
13.18 1318 1187 frame
13.19 1319 1188 frame
13.2 1320
13.21 1321 1189 frame
13.22 1322 1190 frame
13.23 1323 1191 frame
13.24 1324 1192 frame
13.25 1325 1193 frame
13.26 1326 1194 frame
13.27 1327 1195 frame
13.28 1328 1196 frame
13.29 1329 1197 frame
13.3 1330
13.31 1331 1198 frame
13.32 1332 1199 frame
13.33 1333 1200 frame
13.34 1334 1201 frame
13.35 1335 1202 frame
13.36 1336 1203 frame
13.37 1337 1204 frame
13.38 1338 1205 frame
13.39 1339 1206 frame
13.4 1340
13.41 1341 1207 frame
13.42 1342 1208 frame
13.43 1343 1209 frame
13.44 1344 1210 frame
13.45 1345 1211 frame
13.46 1346 1212 frame
13.47 1347 1213 frame
13.48 1348 1214 frame
13.49 1349 1215 frame
13.5 1350
13.51 1351 1216 frame
13.52 1352 1217 frame
13.53 1353 1218 frame


Time Frame frame 3 vias 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 1 ... ... 65 ... ... 113 ... ... 161 ... ... 209 220
13.54 1354 1219 frame
13.55 1355 1220 frame
13.56 1356 1221 frame
13.57 1357 1222 frame
13.58 1358 1223 frame
13.59 1359 1224 frame
13.6 1360
13.61 1361 1225 frame
13.62 1362 1226 frame
13.63 1363 1227 frame
13.64 1364 1228 frame
13.65 1365 1229 frame
13.66 1366 1230 frame
13.67 1367 1231 frame
13.68 1368 1232 frame
13.69 1369 1233 frame
13.7 1370
13.71 1371 1234 frame
13.72 1372 1235 frame
13.73 1373 1236 frame
13.74 1374 1237 frame
13.75 1375 1238 frame
13.76 1376 1239 frame
13.77 1377 1240 frame
13.78 1378 1241 frame
13.79 1379 1242 frame
13.8 1380
13.81 1381 1243 frame
13.82 1382 1244 frame
13.83 1383 1245 frame
13.84 1384 1246 frame
13.85 1385 1247 frame
13.86 1386 1248 frame
13.87 1387 1249 frame
13.88 1388 1250 frame
13.89 1389 1251 frame
13.9 1390
13.91 1391 1252 frame
13.92 1392 1253 frame
13.93 1393 1254 frame
13.94 1394 1255 frame
13.95 1395 1256 frame
13.96 1396 1257 frame
13.97 1397 1258 frame
13.98 1398 1259 frame
13.99 1399 1260 frame
14 1400
14.01 1401 1261 frame
14.02 1402 1262 frame
14.03 1403 1263 frame
14.04 1404 1264 frame
14.05 1405 1265 frame
14.06 1406 1266 frame
14.07 1407 1267 frame
14.08 1408 1268 frame
14.09 1409 1269 frame
14.1 1410
14.11 1411 1270 frame
14.12 1412 1271 frame
14.13 1413 1272 frame
14.14 1414 1273 frame
14.15 1415 1274 frame
14.16 1416 1275 frame
14.17 1417 1276 frame
14.18 1418 1277 frame
14.19 1419 1278 frame
14.2 1420
14.21 1421 1279 frame
14.22 1422 1280 frame
14.23 1423 1281 frame
14.24 1424 1282 frame
14.25 1425 1283 frame
14.26 1426 1284 frame


14.27 1427 1285 frame
14.28 1428 1286 frame
14.29 1429 1287 frame
14.3 1430
14.31 1431 1288 frame
14.32 1432 1289 frame
14.33 1433 1290 frame
14.34 1434 1291 frame
14.35 1435 1292 frame
14.36 1436 1293 frame
14.37 1437 1294 frame
14.38 1438 1295 frame
14.39 1439 1296 frame
14.4 1440
14.41 1441 1297 frame
14.42 1442 1298 frame
14.43 1443 1299 frame
14.44 1444 1300 frame
14.45 1445 1301 frame
14.46 1446 1302 frame
14.47 1447 1303 frame
14.48 1448 1304 frame
14.49 1449 1305 frame
14.5 1450
14.51 1451 1306 frame
14.52 1452 1307 frame
14.53 1453 1308 frame
14.54 1454 1309 frame
14.55 1455 1310 frame
14.56 1456 1311 frame
14.57 1457 1312 frame
14.58 1458 1313 frame
14.59 1459 1314 frame
14.6 1460
14.61 1461 1315 frame
14.62 1462 1316 frame
14.63 1463 1317 frame
14.64 1464 1318 frame
14.65 1465 1319 frame
14.66 1466 1320 frame
14.67 1467 1321 frame
14.68 1468 1322 frame
14.69 1469 1323 frame
14.7 1470
14.71 1471 1324 frame
14.72 1472 1325 frame
14.73 1473 1326 frame
14.74 1474 1327 frame
14.75 1475 1328 frame
14.76 1476 1329 frame
14.77 1477 1330 frame
14.78 1478 1331 frame
14.79 1479 1332 frame
14.8 1480
14.81 1481 1333 frame
14.82 1482 1334 frame
14.83 1483 1335 frame
14.84 1484 1336 frame
14.85 1485 1337 frame
14.86 1486 1338 frame
14.87 1487 1339 frame
14.88 1488 1340 frame
14.89 1489 1341 frame
14.9 1490
14.91 1491 1342 frame
14.92 1492 1343 frame
14.93 1493 1344 frame
14.94 1494 1345 frame
14.95 1495 1346 frame
14.96 1496 1347 frame
14.97 1497 1348 frame
14.98 1498 1349 frame
14.99 1499 1350 frame
15 1500


Time Frame frame 3 vías 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 1 .. .. 65 .. .. 113 .. .. 161 .. .. .. 209 220
15.01 1501 1351 frame
15.02 1502 1352 frame
15.03 1503 1353 frame
15.04 1504 1354 frame
15.05 1505 1355 frame
15.06 1506 1356 frame
15.07 1507 1357 frame
15.08 1508 1358 frame
15.09 1509 1359 frame
15.1 1510
15.11 1511 1360 frame
15.12 1512 1361 frame
15.13 1513 1362 frame
15.14 1514 1363 frame
15.15 1515 1364 frame
15.16 1516 1365 frame
15.17 1517 1366 frame
15.18 1518 1367 frame
15.19 1519 1368 frame
15.2 1520
15.21 1521 1369 frame
15.22 1522 1370 frame
15.23 1523 1371 frame
15.24 1524 1372 frame
15.25 1525 1373 frame
15.26 1526 1374 frame
15.27 1527 1375 frame
15.28 1528 1376 frame
15.29 1529 1377 frame
15.3 1530
15.31 1531 1378 frame
15.32 1532 1379 frame
15.33 1533 1380 frame
15.34 1534 1381 frame
15.35 1535 1382 frame
15.36 1536 1383 frame
15.37 1537 1384 frame
15.38 1538 1385 frame
15.39 1539 1386 frame
15.4 1540
15.41 1541 1387 frame
15.42 1542 1388 frame
15.43 1543 1389 frame
15.44 1544 1390 frame
15.45 1545 1391 frame
15.46 1546 1392 frame
15.47 1547 1393 frame
15.48 1548 1394 frame
15.49 1549 1395 frame
15.5 1550
15.51 1551 1396 frame
15.52 1552 1397 frame
15.53 1553 1398 frame
15.54 1554 1399 frame
15.55 1555 1400 frame
15.56 1556 1401 frame
15.57 1557 1402 frame
15.58 1558 1403 frame
15.59 1559 1404 frame
15.6 1560
15.61 1561 1405 frame
15.62 1562 1406 frame
15.63 1563 1407 frame
15.64 1564 1408 frame
15.65 1565 1409 frame
15.66 1566 1410 frame
15.67 1567 1411 frame
15.68 1568 1412 frame
15.69 1569 1413 frame
15.7 1570
15.71 1571 1414 frame
15.72 1572 1415 frame
15.73 1573 1416 frame
15.74 1574 1417 frame
15.75 1575 1418 frame
15.76 1576 1419 frame
15.77 1577 1420 frame
15.78 1578 1421 frame
15.79 1579 1422 frame

Time	Frame	frame 3 vias
15.8	1580	
15.81	1581	1423 frame
15.82	1582	1424 frame
15.83	1583	1425 frame
15.84	1584	1426 frame
15.85	1585	1427 frame
15.86	1586	1428 frame
15.87	1587	1429 frame
15.88	1588	1430 frame
15.89	1589	1431 frame
15.9	1590	
15.91	1591	1432 frame
15.92	1592	1433 frame
15.93	1593	1434 frame
15.94	1594	1435 frame
15.95	1595	1436 frame
15.96	1596	1437 frame
15.97	1597	1438 frame



FIN DE LA SIMULACIÓN

Apéndice C. Programa para encontrar la Oportunidad de Transmisión de cada nodo.

Este programa se realizó para calcular la oportunidad de transmisión que tiene un nodo para enviar los mensajes REQUEST, GRANT y ACK.

Consta de 3 Funciones y la función q las manda a llamar.

1.- Obtener el número de frame y el número de slot en el que se encuentra el paquete de datos: **frame_slot_data.**
2.- Función para obtener el esxmt del dato: **esxmt_data.**
3.- Función para obtener el número de frame y el número de slot en el que se encuentra la oportunidad de transmisión del nodo: **REQUEST_DSCH** .
4.- Función principal para obtener el tiempo de transmisión de los mensjaes DSCH para un paquete de datos que llegó de la capa de aplicación: **tiempos_calendarizacion.**

```
static void frame_slot_data(double tiempo_data)
{
//Variables para calcular el slot y el frame para el dato q llego
double tiempo_frame=0.0;
int no_frame=0;
double tiempo_restante=0.0;
double comparar=0.0;
double tiempo_no_frame=0.0;
double time_slot=0.0;
double no_slot=0.0, no_slot_1=0.0;
int multiframe=0;
int no_multiframe=0;
int frame_gral=0.0;
int e=2;

FIN(frame_slot_data());

time_slot = 0.000068;
comparar=16.0 * time_slot;
tiempo_frame = 0.01;
multiframe = 10.0; // 10 frames por multiframe

frame_senalizacion_data =-1;
no_frame_total = 0.0;
no_slot_total =0.0;

no_frame = tiempo_data / tiempo_frame;
tiempo_restante = tiempo_data - ( no_frame * tiempo_frame);
if(tiempo_restante < 0){
```

```
        tiempo_restante = 0.0;
}
no_multiframe = no_frame / multiframe;

//Verificar si el dato cayó en un frame de señalización

if(tiempo_restante > 0 ){
         frame_gral =  no_frame + 1;
} else if(tiempo_restante == 0 ){
                frame_gral =  no_frame;
        }
if(frame_gral == (1+ no_multiframe*10) ){
                 frame_senalizacion_data = no_multiframe;
}else {
   // Verificar si el dato se encuentra en el subframe de control o de datos

          if(tiempo_restante == 0 || tiempo_restante > comparar){
                      no_slot_total= 0;
                      if( no_frame % multiframe != 0){
                            no_frame_total = frame_gral - (no_multiframe+1);
                            }else{
                                  no_frame_total = frame_gral - no_multiframe;
                              }
       }else{
              //printf("\n++++++El dato cayo en el subframe de control+++++");
              if( no_frame % multiframe != 0){
                           no_frame_total = frame_gral - (no_multiframe+1);
             }else {
                           no_frame_total = frame_gral - no_multiframe;
                     }
      no_slot= tiempo_restante / time_slot;
      no_slot_1 = no_slot - (int)no_slot;

      if( no_slot_1 > 0){
             no_slot_total = no_slot +1;
      }else if( no_slot_1 == 0){
                       no_slot_total = no_slot;
             }
     }
}
printf("\nel dato se encuentra en el frame: %d y slot: %d\n",no_frame_total, no_slot_total); FOUT;
}

static void esxmt_data(int lim_sup, int exp, int mx, int esxmt_packet)
{
int no_OTX = 16;
int lim_inf;
int H=1;
int I = 1;
int factor;
int exp_1=exp+4;
```

```
int base=2;
int nxmt, nxmt_1 = 0,nxmt_2 = 0;
int lim_sup_1;
int i, j;
int esxmt_actual, esxmt_tx_siguiente, esxmt_anterior;
int inter_exp;
int esxmt_actual_end;
int esxmt_anterior_end;
int esxmt_tx_siguiente_end;

 FIN(esxmt_data());

for(factor=0; factor<exp_1; factor ++)
{
      H=H*base;
}
if(exp==0)
{
    nxmt_1= 1*mx;
    nxmt_2= 1*(mx+1);
}else if(exp !=0)
      {
          for(factor=0; factor<exp; factor ++)
            {
                    I=I*base;
          }
       nxmt_1= I*mx;
       nxmt_2= I*(mx+1);
}
nxmt=nxmt_1 + 1;
inter_exp = pow(base, exp);
if( frame_senalizacion_data != 0){ //llave....0
     if(lim_sup > no_OTX){ //LLAVE 1
          lim_inf = lim_sup - 16 + 1;
          if( lim_inf  % (H+inter_exp) !=0   ){
               i = (lim_inf  / (H+inter_exp) ) + 1;
               esxmt_actual = (i * (H+inter_exp)) + nxmt ;
                  esxmt_anterior = esxmt_actual - (H+inter_exp) ;
               if(esxmt_anterior >= lim_inf ){
                    esxmt_actual = esxmt_anterior;
               }
            esxmt_tx_siguiente = esxmt_actual + (H+inter_exp);
            esxmt_actual_end = esxmt_actual+ inter_exp -1;
            esxmt_anterior_end = esxmt_anterior+ inter_exp -1;
            esxmt_tx_siguiente_end = esxmt_tx_siguiente+ inter_exp -1;
            if((esxmt_packet >= esxmt_anterior) && (esxmt_packet <= esxmt_anterior_end )){//  kiss 1
                esxmt_actual= esxmt_packet;
             }/*      kiss      1*/      else      if((esxmt_packet      >=      esxmt_actual)      &&      (esxmt_packet      <=
esxmt_actual_end )){//kiss 2
   esxmt_actual= esxmt_packet;
 }/* kiss 2*/else if((esxmt_packet >= esxmt_tx_siguiente) && (esxmt_packet <= esxmt_tx_siguiente_end )){// kiss 3
```

```
                        esxmt_actual= esxmt_packet;
                    }//kiss 3


 if(esxmt_actual > lim_sup){
      lim_sup_1 = lim_sup;
      while (esxmt_actual > lim_sup_1){
            lim_sup_1 = lim_sup_1  + no_OTX ;
            frame_request_DSCH = frame_request_DSCH+1;
     }
 }

slot_request_DSCH = esxmt_actual - ( ((lim_sup / 16 ) - 1 )* 16 );
while(slot_request_DSCH > no_OTX){
      slot_request_DSCH= slot_request_DSCH - no_OTX;
}

}else  if( lim_inf  % (H+ inter_exp) == 0   ){
          i = (lim_inf  / (H+ inter_exp) );
          esxmt_actual = (i * (H+ inter_exp)) + nxmt ;
          esxmt_anterior = esxmt_actual - (H+ inter_exp);
          if(esxmt_anterior >= lim_inf ){
              esxmt_actual = esxmt_anterior;
         }
esxmt_tx_siguiente = esxmt_actual + (H+ inter_exp);
esxmt_actual_end = esxmt_actual+ inter_exp -1;
esxmt_anterior_end = esxmt_anterior+ inter_exp -1;
esxmt_tx_siguiente_end = esxmt_tx_siguiente+ inter_exp -1;
if((esxmt_packet >= esxmt_anterior) && (esxmt_packet <= esxmt_anterior_end )){// kiss 1...1
  esxmt_actual= esxmt_packet;
 }/* kiss 1...1*/ else if((esxmt_packet >= esxmt_actual) && (esxmt_packet <= esxmt_actual_end )){//kiss 2...2
                              esxmt_actual= esxmt_packet;
                          }/*    kiss    2...2*/else    if((esxmt_packet    >=    esxmt_tx_siguiente)    &&    (esxmt_packet    <=
esxmt_tx_siguiente_end )){// kiss 3...3
                               esxmt_actual= esxmt_packet;
                                   }//kiss 3...3
if(esxmt_actual > lim_sup){
  lim_sup_1 = lim_sup;
  while (esxmt_actual > lim_sup_1){
        lim_sup_1 = lim_sup_1  + no_OTX ;
        frame_request_DSCH = frame_request_DSCH+1;
  }
}
slot_request_DSCH = esxmt_actual - ( ((lim_sup / 16 ) - 1 )* 16 );
while(slot_request_DSCH > no_OTX){
      slot_request_DSCH= slot_request_DSCH - no_OTX;
}
}
}/*LLAVE 1*/ else if(lim_sup <= no_OTX){
                        slot_request_DSCH = nxmt;
                    }
```

```
}/*llave..0 */  else if(frame_senalizacion_data >= 0 && lim_sup == (-1)){
                        if(frame_senalizacion_data == 0){
                          esxmt_actual= nxmt;
                          slot_request_DSCH = esxmt_actual;
                                frame_request_DSCH = 1;
                          }
}
FOUT;
} //Fin de la función 2

static void REQUEST_DSCH (int xmt_h_exp, int xmt_mx)
{  //empieza funcion 3
int lim_sup;
int num_frames=9;
int x, z;
double sum_slot;
double sum_frame;
double sum_frame_1;
double time_copy;
int inter_exp_func3;
int esxmt_pk=0;

FIN(REQUEST_DSCH ());

inter_exp_func3 = pow(2, xmt_h_exp);
if( frame_senalizacion_data ==(-1)){ // llave 1
   if(no_slot_total==0){//llave...2
     frame_request_DSCH = no_frame_total + 1 ;
     lim_sup= frame_request_DSCH * 16;
     esxmt_pk = (no_frame_total*16)+1;
     esxmt_data(lim_sup, xmt_h_exp, xmt_mx, esxmt_pk );
     if ((frame_request_DSCH % num_frames)==0){
         x= frame_request_DSCH / num_frames;
         x=x-1;
     }else{
            x=frame_request_DSCH / num_frames;
          }
sum_slot =  slot_request_DSCH*0.000068000;
sum_frame = frame_request_DSCH + x;
sum_frame_1 = sum_frame * 0.010000000;
time_request_DSCH = 0.0;
time_copy = sum_slot + sum_frame_1;
time_request_DSCH = sum_slot + sum_frame_1;
printf("\n\n ++++++La Oportunidad de Tx esta en el frame:%i, en el slot=%i y su tiempo de tx es: %f
",frame_request_DSCH, slot_request_DSCH , time_request_DSCH);
} else if(no_slot_total!=0){//cierra llave...2 abrre llave ...3
         frame_request_DSCH = no_frame_total;
         lim_sup= frame_request_DSCH * 16;
         esxmt_pk = ((no_frame_total-1)*16 ) + no_slot_total;
         esxmt_data(lim_sup, xmt_h_exp, xmt_mx ,esxmt_pk);
         if(frame_request_DSCH == no_frame_total){// 60
```

```
            if(no_slot_total > slot_request_DSCH){ //61
               frame_request_DSCH = frame_request_DSCH + 1;
            lim_sup= frame_request_DSCH * 16;
            esxmt_data(lim_sup, xmt_h_exp, xmt_mx ,esxmt_pk);
            if ((frame_request_DSCH % num_frames)==0){
                x= frame_request_DSCH / num_frames;
                 x=x-1;
            }else{
                   x=frame_request_DSCH / num_frames;
                 }
           sum_slot =  slot_request_DSCH*0.000068000;
           sum_frame = frame_request_DSCH + x;
           sum_frame_1 = sum_frame * 0.010000000;
           time_request_DSCH = 0.0;
           time_request_DSCH = sum_slot + sum_frame_1;
           } else if(no_slot_total <= slot_request_DSCH ){ // cierra 61 y abre 64
                    if ((frame_request_DSCH % num_frames)==0){
                        x= frame_request_DSCH / num_frames;
                        x=x-1;
                     }else{
                            x=frame_request_DSCH / num_frames;
                          }
           sum_slot =  slot_request_DSCH*0.000068000;
           sum_frame = frame_request_DSCH + x;
           sum_frame_1 = sum_frame * 0.010000000;
           time_request_DSCH = 0.0;
           time_request_DSCH = sum_slot + sum_frame_1;
           printf("\n\nLa Oportunidad de Tx esta en el frame:%i, en el slot=%i y su tiempo de tx es: %f ",frame_request_DSCH,
slot_request_DSCH , time_request_DSCH);
  }//64
 }//if(frame_request_DSCH == no_frame_total) ...60
 else if(frame_request_DSCH > no_frame_total){  /*llave *7*/
         if ((frame_request_DSCH % num_frames)==0){
             x= frame_request_DSCH / num_frames;
             x=x-1;
          }else{
                 x=frame_request_DSCH / num_frames;
                }
sum_slot =  slot_request_DSCH*0.000068000;
sum_frame = frame_request_DSCH + x;
sum_frame_1 = sum_frame * 0.010000000;
time_request_DSCH = 0.0;
time_request_DSCH = sum_slot + sum_frame_1;
printf("\n\nLa Oportunidad de Tx esta en el frame:%i, en el slot=%i y su tiempo de tx es: %f ",frame_request_DSCH,
slot_request_DSCH , time_request_DSCH);
} // if(frame_request_DSCH > no_frame_total)... /*llave *7*/
} //llave...3
}/*llave 1 */ else if(frame_senalizacion_data >= 0){ //Cuando el dato cayo en un frame de señalizacion
                        if(frame_senalizacion_data == 0){
                           lim_sup = -1;
                        }else {
```

```
                                z = frame_senalizacion_data - 1;
                                frame_request_DSCH = 10+(z*9);
                                lim_sup= frame_request_DSCH * 16;
                               }
    esxmt_data(lim_sup, xmt_h_exp, xmt_mx,esxmt_pk);
    if ((frame_request_DSCH % num_frames)==0){
        x= frame_request_DSCH / num_frames;
         x=x-1;
    }else{
            x=frame_request_DSCH / num_frames;
          }
sum_slot =  slot_request_DSCH*0.000068000;
sum_frame = frame_request_DSCH + x;
sum_frame_1 = sum_frame * 0.010000000;
time_request_DSCH = 0.0;
time_request_DSCH = sum_slot + sum_frame_1;
 }
printf("\n\nLa Oportunidad de Tx esta en el frame:%i, en el slot=%i y su tiempo de tx es: %f ",frame_request_DSCH,
slot_request_DSCH , time_request_DSCH);

 FOUT;
}//Termina la funcion 3

static void tiempos_calendarizacion(double dato_generado, int xmt_h_exp, int xmt_mx)
{

   FIN(tiempos_calendarizacion());

frame_slot_data(dato_generado);
REQUEST_DSCH ( xmt_h_exp, xmt_mx);
FOUT;
}
```

Apéndice D . Programa para determinar el tiempo de transmisión de los paquetes de datos de cada nodo.

Este programa se realizó para calcular el tiempo de transmisión que le corresponde a un paquete de datos.

Cuando un nodo desea transmitir un paquete de datos, este solicita recursos al nodo receptor; este nodo receptor de acuerdo a la disponibilidad de ancho de banda y los parámetros del transmisor, asigna los recursos.

Esta asignación de recursos se encuentra cuando se programa el mensaje GRANT en el nodo receptor.

```
static Packet* encapsulate_GRANT(int tipo, int g_size)
{
Packet* pkt_msh_dsch_ptr;
Packet* hdr_ptr;
REQUEST_IE *request_ie_g;
GRANT_IE_LIST *grant_list_struct;
PACKET_STRUCT* pk;
PKSEND_STRUCT* pk_send;
int i,p = 0;
int  grant_part1_v, grant_part2_v = 0;
int cont;
int index;
double time;
int index_ack;
int index_ack_neigh;
NEIGHBOR_INIT_STRUCT* data_ack;
NEIGHBOR_INIT_STRUCT* data_ack_neigh;
int exp_source; //13-mayo
int mx_source;  //13-mayo
int frame_ack; //13-mayo
int gano = 0;
int listSize=0;
List* static_extended_neigh_ack;
int mac; //30-mayo
int listSize_req; //30-mayo
int j;//30 mayo
int no_pk_grant; //30-mayo
int direction; //30 mayo
int w; //23 junio
int index_ack_nodo; //25 junio
int q; //26junio
REQUEST_IE* pk_req_enc;
int c_nodo_link;
```

```
int c_pk_list_req;
int  c_mac_source;
int c_pk_req;
int c_pk;
int  star_nodo;
int mac_sources_arr[4];
int cont_slot;
int cont_gano;
int qos;
int mini_slot_voip;
int frame_voip;
int cont_voip;
double time_voip;
PKSEND_VOIP_STRUCT* pk_send_voip;

FIN(encapsulate_GRANT());

grant_list_struct =  (GRANT_IE_LIST*) op_prg_mem_alloc (sizeof (GRANT_IE_LIST));
if(tipo == 2) //Grant
{
 g_size = op_prg_list_size(request_list);
 grant_list_struct->grant_ie_list =  (GRANT_IE*) op_prg_mem_alloc ((g_size) *sizeof (GRANT_IE));
 c_pk = 0;
 for(w=0; w< 3; w++){
     mac_sources_arr[w]= 0;
}
for(c_pk_list_req = 0; c_pk_list_req < g_size;  c_pk_list_req++ ){// FOR MAC_SOURCE
     request_ie_g = (REQUEST_IE*) op_prg_list_access(request_list,c_pk_list_req);
     c_nodo_link = 0;
     while(request_ie_g->linkID !=  static_topology[c_nodo_link].linkID){
              c_nodo_link = c_nodo_link +1;
     }
    mac_source = static_topology[c_nodo_link].node_from;
    c_mac_source= 0;
    while(op_prg_list_access(req_grant_list,c_mac_source) != mac_source){
            c_mac_source= c_mac_source+1;
    }
    if(op_prg_list_access(req_grant_list,c_mac_source) ==  mac_source){//if comp mac_sources
       c_pk =  c_pk+1;
       c_pk_req = op_prg_list_access(req_grant_pk_list,c_mac_source);
       if( mac_sources_arr[c_mac_source]== 0){//if sacar star frame
           index_ack = mac_source -1000;
           index_ack_nodo = index_ack-1;
           data_ack = op_prg_list_access(init_neigh ,index_ack);
           exp_source = data_ack -> xmt_holdoff_exp;
           mx_source = data_ack ->xmt_mx ;
           tiempos_calendarizacion(op_sim_time()+ 0.01, exp_source, mx_source);
           static_extended_neigh_ack = getVecindadExtendida(mac_source);
           listSize =  op_prg_list_size(static_extended_neigh_ack);
           gano =0;
           cont_gano = 0;
```

```
        time_grant_global->time_grant_global_list[index_ack_nodo].cont_gano_1 = 0;
        while(gano != 1){// while .. gano
              gano = selected_time_DSCH(listSize, mac_source, time_request_DSCH, static_extended_neigh_ack);
              cont_gano = cont_gano +1;
              if(gano == 1  && cont_gano== 1 ){//0
                 if((no_frame_total == frame_request_DSCH) && (no_slot_total == slot_request_DSCH) ){//1
                      tiempos_calendarizacion(time_request_DSCH + 0.000068 , exp_source, mx_source);
                      static_extended_neigh_ack = getVecindadExtendida(mac_source);
                       listSize =  op_prg_list_size(static_extended_neigh_ack);
                       gano = selected_time_DSCH(listSize, mac_source, time_request_DSCH, static_extended_neigh_ack);
                       if(gano == 1){
                         time_grant_global->time_grant_global_list[index_ack_nodo].cont_gano_1 = cont_gano;
                       }
                    }//1
              }//0
              if(gano != 1){
              tiempos_calendarizacion(time_request_DSCH + 0.000068, exp_source, mx_source);
              }
         }// while .. gano
        frame_ack= frame_request_DSCH;
        time_grant_global->time_grant_global_list[index_ack_nodo].time_ack_nodo_win = time_request_DSCH;
        if(frame_ack > ( star_frame_minislot_global->star_frame_minislot_list[ index_ack_nodo].star_frame_number_nodo)
){
           star_frame_minislot_global->star_frame_minislot_list[index_ack_nodo].minislot_star_nodo = 1;
           star_frame_minislot_global->star_frame_minislot_list[ index_ack_nodo].star_frame_number_nodo  = frame_ack;
        }
        mac_sources_arr[c_mac_source]= 1;
      }/*if sacar star frame*/
     star_nodo= (mac_source - 1000)-1;
     start_Frame_number= star_frame_minislot_global->star_frame_minislot_list[star_nodo].star_frame_number_nodo;
     minislot_start_c = star_frame_minislot_global->star_frame_minislot_list[star_nodo].minislot_star_nodo;
     for(cont_slot= 0; cont_slot< 6; cont_slot++){
       if(start_Frame_number<=star_frame_minislot_global->star_frame_minislot_list[cont_slot].star_frame_number_nodo){
//1
          if((start_Frame_number                                    ==                                    tar_frame_minislot_global-
>star_frame_minislot_list[cont_slot].star_frame_number_nodo)    &&    (minislot_start_c    <    star_frame_minislot_global-
>star_frame_minislot_list[cont_slot].minislot_star_nodo )){// if2
              if(star_nodo != cont_slot ){
                 minislot_start_c    =    star_frame_minislot_global->star_frame_minislot_list[cont_slot].minislot_star_nodo;

                 if(minislot_start_c > 220){
                    minislot_start_c = 1;
                    start_Frame_number= start_Frame_number+1;
                 }
              }
           }else
if(start_Frame_number<star_frame_minislot_global->star_frame_minislot_list[cont_slot].star_frame_number_nodo){// if2 ...
if3
           if(star_nodo != cont_slot ){
              minislot_start_c = star_frame_minislot_global->star_frame_minislot_list[cont_slot].minislot_star_nodo;
```

```
                start_Frame_number=                                                    star_frame_minislot_global-
>star_frame_minislot_list[cont_slot].star_frame_number_nodo;
                star_frame_minislot_global->star_frame_minislot_list[star_nodo].star_frame_number_nodo                =
start_Frame_number;
                star_frame_minislot_global->star_frame_minislot_list[star_nodo].minislot_star_nodo=minislot_start_c ;



                if(minislot_start_c > 220){
                    minislot_start_c = 1;
                    start_Frame_number= start_Frame_number+1;
                }
                      }
        }//if3
      star_frame_minislot_global->star_frame_minislot_list[star_nodo].star_frame_number_nodo = start_Frame_number;
      star_frame_minislot_global->star_frame_minislot_list[star_nodo].minislot_star_nodo=minislot_start_c ;
    }//1
}
if( request_ie_g->priority != 3){
    grant_list_struct->grant_ie_list[c_pk_list_req].linkID = request_ie_g->linkID;
    grant_list_struct->grant_ie_list[c_pk_list_req].minislot_range = request_ie_g->demand__level;
    grant_list_struct->grant_ie_list[c_pk_list_req].direction = 1;
    grant_list_struct->grant_ie_list[c_pk_list_req].start_frame_number = start_Frame_number;
    minislot_start_c += request_ie_g->extra; //08 junio para recorrer los slots q van en el mismo grant para otros nodos
    grant_list_struct->grant_ie_list[c_pk_list_req].minislot_start = minislot_start_c;
    if(cont_frame_ut == start_Frame_number ){
       cont_slot_ut = cont_slot_ut + request_ie_g->demand__level;
       if(cont_slot_ut > 220){
          slot_restantes= cont_slot_ut-220;
          cont_slot_ut = 220;
          util_frame = (double) (cont_slot_ut *100)/220;
          op_stat_write (util_dsch, util_frame);
          cont_slot_ut= slot_restantes;
          cont_frame_ut = cont_frame_ut +1;
          util_frame_rest = (double) (cont_slot_ut *100)/220;
       }
       util_frame = (double) (cont_slot_ut *100)/220;
       op_stat_write (util_dsch, util_frame);
    }else if(cont_frame_ut < start_Frame_number){
                if(util_frame_rest > 0){
                    util_frame = util_frame_rest;
                    op_stat_write (util_dsch, util_frame);
                    util_frame_rest = 0.0;
                }
               cont_frame_ut = start_Frame_number;
               cont_slot_ut = request_ie_g->demand__level;
               if(cont_slot_ut > 220){
                   slot_restantes= cont_slot_ut-220;
                   cont_slot_ut = 220;
                   util_frame = (double) (cont_slot_ut *100)/220;
                   op_stat_write (util_dsch, util_frame);
```

```
                cont_slot_ut= slot_restantes;
                cont_frame_ut = cont_frame_ut +1;
            }
            util_frame = (double) (cont_slot_ut *100)/220;
            op_stat_write (util_dsch, util_frame);
        }
util_frame_ave= util_frame_ave + request_ie_g->demand__level;
avail_mark(start_Frame_number, request_ie_g->demand__level, 1,request_ie_g->linkID);
if((minislot_start_c + request_ie_g->demand__level) >220){
    minislot_start_c = (minislot_start_c + request_ie_g->demand__level) - 220;
    start_Frame_number=start_Frame_number+1;
}else{
       minislot_start_c = minislot_start_c + request_ie_g->demand__level;
      }
star_frame_minislot_global->star_frame_minislot_list[star_nodo].star_frame_number_nodo= start_Frame_number;
star_frame_minislot_global->star_frame_minislot_list[star_nodo].minislot_star_nodo = minislot_start_c;
}else if(request_ie_g->priority == 3){// voip ..1
      }// voip ..1
}//if comp mac_sources
}// FOR MAC_SOURCE
print_local_avail();
pkt_msh_dsch_ptr = op_pk_create_fmt(MSH_DSCH_FMT);
pk_nfd_set(pkt_msh_dsch_ptr, "Management Message Type", 41);
op_pk_nfd_set(pkt_msh_dsch_ptr, "Coordination Flag", 1);
op_pk_nfd_set(pkt_msh_dsch_ptr, "Grant/Request Flag", (tipo==1)?0:1);
op_pk_nfd_set(pkt_msh_dsch_ptr, "Sequence counter", 1);
op_pk_nfd_set(pkt_msh_dsch_ptr, "No Requests", 0);
op_pk_nfd_set(pkt_msh_dsch_ptr, "No Availabilities", 0);
op_pk_nfd_set(pkt_msh_dsch_ptr, "No Grants", g_size);
op_pk_nfd_set(pkt_msh_dsch_ptr,  "MSH-DSCH_Grant_IE",grant_list_struct,  op_prg_mem_copy_create,  op_prg_mem_free,
sizeof (GRANT_IE_LIST));
hdr_ptr = op_pk_create_fmt(MSH_HD_PK_FMT);
op_pk_nfd_set(hdr_ptr, "MAC Source", mac_address);
op_pk_nfd_set(hdr_ptr, "MAC Destination", mac_destination);
pkt_send = op_pk_create_fmt(MSH_PK_FMT);
op_pk_nfd_set(pkt_send, "Header", hdr_ptr);
op_pk_nfd_set(pkt_send, "Payload", pkt_msh_dsch_ptr);
op_pk_total_size_set (pkt_send, 208);

FRET(pkt_send);
```