UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"ANÁLISIS DEL COMPORTAMIENTO DINÁMICO DE REDES INÁLAMBRICAS DE BANDA ANCHA CON ARQUITECTURA MESH"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRA EN INGENIERÍA
(COMPUTACIÓN)

P R E S E N T A:

YASMINE MACEDO REZA

DIRECTOR DE TESIS:

DR. VICTOR RANGEL LICEA

México, D.F. Agosto 2008.

Dedicatoria

A mi familia y a mi novio.

A mi familia por apoyarme en cada etapa de mi vida.

A mis padres Damián y Dina por ser mi guía e impulsarme a mejorar.

A mis hermanas, cuñados y sobrinos por ser incondicionales en mi vida.

A mi novio Efrén Arturo por todo el cariño, apoyo y comprensión a lo largo de estos años.

A todos mis grandes amigos que me han dado la alegría para salir adelante en cada una de las etapas de la vida.

Agradecimientos

A la Universidad Nacional Autónoma de México por impulsar mi desarrollo profesional y humano.

Al Posgrado en Ciencia e Ingeniería de Computación por el apoyo recibido para la adquisición de licencias para el software de OPNET.

Al Instituto de Investigaciones de Matemáticas Aplicadas y Sistemas, sus profesores y administración por ayudar a realizar mis estudios de maestría.

Al Dr. Víctor Rangel Licea por su apoyo y asesoría para la realización del trabajo realizado. Ya que sin su guía y paciencia no habría sido posible concluir este trabajo. Por el tiempo dedicado

Al CONACYT por la beca otorgada durante los estudios y a la DGAPA-UNAM por el apoyo recibido, por parte del proyecto de investigación PAPIIT No. IN-104907 "Diseño de técnicas de mejoramiento de capacidad en redes inalámbricas de banda ancha tipo Mesh".

Índice general

Índice de figuras

Índice de tablas

Resumen

Las redes inalámbricas han crecido de modo exponencial, debido al crecimiento de número de usuarios con consecuencia una gran demanda de ancho de banda. Por lo que es necesario utilizar nuevas tecnologías, que puedan satisfacer a un mayor número de usuarios.

El estándar IEEE 802.16 describe el protocolo para utilizar redes inalámbricas de banda ancha. Este estándar utiliza un mecanismo central basado en acceso múltiple por división de tiempo, en la versión 2004, se describe estas redes con una topología en malla (Mesh). Y la división de tiempos entre transmisiones de la estación base y de las estaciones suscriptoras para este tipo de arquitectura.

Las redes Mesh brindan un servicio que puede ser no centralizado y brindar servicio en zonas de difícil acceso como áreas rurales, zonas montañosas y áreas que por alguna razón no pueden ser cubiertas por compañías que brindan servicios celulares o de Internet. Ya que brevemente descrita, su funcionalidad consiste en que el nodo que esta conectado con la red exterior tiene la funcionalidad de estación base y es el proveedor de servicio hacia los demás nodos, y de esta forma todos los nodos tengan contacto hacia al exterior.

En el presente trabajo se desarrolló un modelo de simulación utilizando el software OPNET, de la inicialización de una red Mesh de hasta 100 nodos divididos en 3 grupos de 1,2 y 3 saltos de la estación proveedora de servicio probando el correcto ingreso a la red, de todos los nodos. Y proponiendo los principales parámetros de tiempo para un uso eficiente del canal de comunicación en los mensajes de control de inicialización de la red.

Capítulo 1

Introducción

1.1. Antecedentes

Actualmente existe una gran gama de tecnologías que están apoyando el crecimiento y expansión del mundo de las telecomunicaciones, este fenómeno conlleva una adquisición enorme de usuarios demandantes del servicio de red. Al haber una multitud de peticiones de servicio y tráfico de información, se requieren tecnologías que puedan cubrir dichos requerimientos.

Por otro lado, las tecnologías inalámbricas han incrementado fuertemente su mercado tanto en la industria como en usuarios finales, cada vez es más común que un usuario particular tenga acceso a servicios inalámbricos a bajo costo.

Una mezcla de los dos puntos anteriores ha impulsado a investigadores de todo el mundo enfocar en una sola tecnología llamada BWA *(Acceso Inalámbrico de Banda Ancha)*[1] que permita brindar un servicio con alta tasa de transferencia de datos , bajo costo y con menos errores de transmisión, así como extender el alcance del servicio de manera que todos los usuarios que lo deseen tengan acceso al servicio celular o internet en cualquier lugar que se encuentren.

Existen principalmente dos tecnologías BWA. La primera es la propuesta dada por el ETSI *(Instituto Europeo de Normas de Telecomunicaciones)*[19]
ETSI BRAN actualmente producen especificaciones para tres áreas principalmente:

HiperLAN2 Una red móvil de banda ancha de corto alcance

HIPERACCESS Una red inalámbrica fija de banda ancha de corto alcance

HIPERMAN Una red inalámbrica la cual opera por debajo de los 11 GHz

La segunda tecnología es la propuesta por la IEEE [2], en la que se enfoca de lleno el presente trabajo que comenzó en 1998 cuando se creó el grupo de trabajo denominado IEEE 802.16.

[1]Se incluye glosario de acrónimos en inglés, véase Apéndice A

Este grupo ha creado un estándar, comúnmente conocido como WiMAX *(Worldwide interoperability for Microwave Access)* para redes BWA. A su vez este estándar se ha desplegado en varias versiones, que se identificaran más adelante, cabe mencionar a modo de preámbulo que la versión que se utilizará para el presente trabajo será IEEE802.16-2004 que define una tecnología de acceso inalámbrico fijo, que esta diseñada para servir como una tecnología de reemplazo DSL, o para proveer acceso básico de voz y también una solución viable para redes backhaul inalámbrico.

El 802.16 utiliza OFDM *(Orthogonal Frequency Division Multiplexing)* para servir a múltiples usuarios en una forma de división temporal en una especie de técnica circular de tal forma que los usuarios tienen la sensación de que siempre están recibiendo o transmitiendo.

El estándar IEEE 802.16 define dos formas de operación, Punto-a-Multipunto (PMP) y Mesh. En el modo de PMP el tráfico se dirige de la estación base (BS) a las estaciones suscriptoras (SSs), o viceversa. En el modo Mesh por el contrario, el tráfico puede ocurrir directamente entre SSs, es decir los usuarios no tienen que tener una conectividad directa con la BS si no conectarse con una serie de usuarios vecinos e ir formando la ruta hacia la BS.

1.2. Definición del problema

La BS Mesh es la entidad que interconecta la red inalámbrica a las ligas externas. Actúa como una BS en modo PMP, excepto en que las SSs no tiene que estar conectadas directamente con la BS Mesh. Se espera que esta característica de la interfaz aérea ayude que las tasas de datos puedan ser más altas. Por tanto, se investigará las redes Mesh que especifica la comunicación directa entre SSs y la capa MAC y permite realizar conexiones multi-hop. La conexión puede ser centralizada o distribuida. Las redes Mesh cuya topología no se basa en puntos de acceso que hacen una ruta hacia una red ordenada, sino que trabajan similar a las redes punto a punto. La arquitectura de las redes Mesh utiliza conexiones redundantes entre los dispositivos de red. Cada cliente es un punto de acceso.

El estándar IEEE802.16 con arquitectura Mesh provee de varias ventajas además de incrementar el alcance y el ancho de banda. Su mecanismo central basado en TDMA *(Time Division Multiple Access)* permite reservar slots centrales para una utilización eficiente de los recursos globales apropiada para redes inalámbricas fijas tipo backhaul (conectan redes de datos, redes de telefonía celular y constituyen una estructura fundamental de las redes de comunicación). Sin embargo la interferencia representa un problema mayor en redes multi-hop WiMAX tipo Mesh. Para proveer un alto rendimiento espectral se necesita un algoritmo eficiente para reservación de slots. El nivel de interferencia depende de cómo los datos son enrutados en la red WiMAX.

Por lo que se menciona anteriormente las redes Mesh amplían la capacidad de expansión de la red, pero para esto se necesita definir algoritmos de ruteos que optimicen la comunicación entre nodos.

Para poder brindar servicios de comunicación en un área más amplia de cobertura, y teniendo en cuenta los costos y desventajas de poner toda una infraestructura en un área donde no existen suficientes usuarios que la requieran, se propone el uso de redes Mesh como una ampliación de cobertura a bajo costo.

Las redes Mesh al proveer un servicio no centralizado están colocadas en la mejor posición para brindar servicios en zonas de difícil acceso, como zonas rurales, zonas montañosas, zonas que por alguna razón no pueden ser cubiertas por compañías que brindan servicios celulares o de Internet.

La arquitectura Mesh es una solución para proveer a las redes de banda ancha con una cobertura mayor, por ejemplo llevar internet e información a las áreas rurales que se encuentren el diferentes países. Estas redes están siendo diseñadas para que puedan proveer de una cobertura extendida que aún no han sido cubiertas por la tecnología existente xDSL o Cable Modem.

La expansión de la red se asemejaría a un fenómeno de biología celular o neuronal. Los sistemas de redes Mesh son una forma de enrutar datos, voz e instrucciones entre nodos. Esto permite conexiones continuas y reconfiguraciones alrededor de vías bloqueadas mediante el "salto"de un nodo a otro hasta que se pueda establecer una conexión.

Estas redes se organizan y se configuran automáticamente: entre los nodos de la red se establecen los enlaces y mantiene la conectividad en malla entre ellos. Además este proceso se debe realizar periódicamente, permitiendo la aparición de nuevos nodos y la desaparición de nodos existentes. Estas características aportan grandes ventajas como robustez, fiabilidad y un mantenimiento sencillo de la red.

Las redes de malla son confiables debido a que cada nodo esta conectado a otros nodos. Si un nodo se retira de la red por daño, sus vecinos simplemente encuentran otra ruta.

Para lograr un funcionamiento eficiente y correcto de estas redes, es recomendable crear primero modelos que simulen el comportamiento de cada sus diferentes fases.

1.3. Objetivos

Análisis, diseño e implementación un modelo de simulación de la inicialización de redes Mesh de banda ancha BWA, en el que se detalle el inicio y registro dentro de la red de todos los nodos que la integran y se analicen los parámetros que intervienen en este proceso.

Construir un modelo que sirva de base para proponer técnicas de mejoramiento en el rendimiento de las redes Mesh, búsqueda de rutas óptimas, técnicas de planeación de tiempo y una futura implementación de envío de datos. Este modelo se utilizará referencia para el estudio del comportamiento dinámico de la red, tanto para los usuarios fijos como usuarios móviles.

Al mismo tiempo tiene como objetivo realizar investigación de vanguardia que permita incrementar y extender el conocimiento actual de redes Mesh con tecnología inalámbrica de banda ancha. Así mismo, contribuir al fortalecimiento y desarrollo de la investigación científica y tecnológica en el país.

1.4. Metodología

Para la creación del modelo se utilizó la versión 8 del software OPNET *(Optimun Network Performance Modeler)*, con licencia para el proyecto PAPIIT No IN104907 "Diseño de técnicas de mejoramiento de capacidad en redes inalámbricas de banda ancha tipo Meshz el Posgrado de Ciencia e Ingeniería de la Computación.

Se desarrolló un modelo que fue analizado, diseñado y construido desde el principio, es decir, se programaron todas las funcionales de la capa MAC que se presentan en este trabajo, a partir del estándar IEEE 802.16-2004. El comportamiento de la red Mesh, se estudió a lo largo de la realización del modelo con el fin de obtener los parámetros de configuración óptimos de acuerdo al comportamiento del modelo. Algunos de estos parámetros también se obtuvieron de la lectura de múltiples artículos relacionados con el tema.

Se analizaron algoritmos para sincronización de tiempos para envío de mensajes de configuración y el intercambio de mensaje entre los diferentes niveles de nodos para inicio y registro de cada uno de los nodos dentro de la red, todo esto a nivel capa MAC.

La funcionalidad fue construida primero para un número pequeño de nodos y después para un número mayor. Con el modelo más complejo se concluyeron los resultados que a su vez fueron comparados con los existentes.

Se implementó un programa que desarrollaba los algoritmos de tiempos para el envío de mensajes, para tener una primera aproximación a los tiempos en que se enviaban cada uno de los mensajes de inicialización. Este programa se utilizó para la comparación de tiempos con el modelo realizado en OPNET el cual incluyó el manejo de colisiones.

Posteriormente se evaluó el comportamiento de la inicialización de la red con diferentes parámetros.

1.5. Contribución y relevancia

El principal aporte de esta tesis es proporcionar un modelo base que implemente el registro e inicialización de los usuarios para que posteriormente se pueda llevar a cabo el estudio del comportamiento dinámico de las redes con topología Mesh bajo el estándar IEEE802.16.

Este modelo colaborará para que se realice investigación sobre mejoras y especificaciones en el uso de este tipo de topología. Existen trabajos que simulan el comportamiento con los planificadores de tiempo, o la optimización de rutas, ya sea de forma teórica o por medio de un modelo. Pero no hay un modelo que englobe el comportamiento desde la inicialización que es imprescindible para el funcionamiento de la red.

Será la base y texto de consulta para futuras tesis sobre redes BWA con arquitectura Mesh basadas en el estándar IEEE 802.16.

Este modelo tiene varias características para ser explotado, ya que puede servir como punto de referencia para la estudio de algoritmos de enrutamiento, estudio de la capacidad de la red, envío de paquetes de control para planificadores, movilidad, entre otros.

Por otro lado será anexado al consorcio nacional de academias de OPNET *Technologies*, para que pueda ser utilizado por cualquier centro de investigación perteneciente al grupo de investigación de OPNET, actualmente este grupo está formado por más 2000 universidades de todo el mundo.

1.6. Estudios relacionados

Hasta la fecha en que se escribe este trabajo aún no se ha implementando una propuesta para el estudio dinámico de los tiempos para la inicialización de las redes en topología mesh, existen varios artículos que proponen algunos parámetros y algoritmos en los diferentes tipos de planeación ya sea de forma centralizada (todos los paquetes pasan por la estación base) o distribuida (la administración de tiempos se re-envía desde cada uno de los nodos).

El artículo [12] se relaciona fuertemente a la inicialización de la red, pero no hay referencia a los tiempos en los mensajes de control para el inicio en la red, enfocando más a fondo al siguiente paso dentro del funcionamiento de la arquitectura Mesh que es establecer vínculos con los nodos vecinos.

En [10] se hace una comparación entre las diferentes propuestas para implementar planeadores de tiempo *schedulers* tanto centralizados y distribuidos, además de que muestra la división de tiempos a la que se hace referencia en este trabajo más adelante.

En [6] y [8] coinciden en los parámetros que utilizados para definir los tiempos para el estudio de los planificadores centralizados, aunque estas propuestas se hicieron para 10 nodos dentro de la red, en esta tesis se planea observar un comportamiento masivo, aunque claro en la inicialización de la red, que tiene muchas coincidencias con el planificador centralizado, ya que todos los mensajes de control de inicialización tienen que pasar por la estación base.

1.7. Estructura de la tesis

A continuación se presenta la estructura que formará el presente trabajo:

Capítulo 2 En este capítulo se presenta un panorama general de las redes de acceso inalámbrico de banda ancha. Se describe la evolución que las redes inalámbricas han tenido en los últimos años, pasando desde las redes de 2^a generación hasta la 4^a. Al final se explica el comportamiento de las redes de banda ancha haciendo hincapié en el estándar IEEE 802.16.

Capítulo 3 Se describe el protocolo IEEE 802.16, un entorno general del estándar y los procesos de comunicación entre los diferentes tipos de nodos. Las diferentes tipos de

topologías que se pueden utilizar en el estándar, resaltando la operación del modo Mesh que servirá para entender el modelo desarrollado.

Capítulo 4 Se expone el modelo propuesto de las redes Mesh, se realiza un análisis detallado de los módulos, procesos, estados y directivas que determinan el comportamiento de los nodos involucrados, así como el diseño e implementación para que el modelo funcione, el cual estará implementando en Proto-C [2] utilizando OPNET.

Capítulo 5 Se exponen los resultados obtenidos del modelo de simulación descrito en el capítulo 4, con cada uno de los diferentes modo de sincronizar los tiempos durante la inicialización de la red.

Capítulo 6 Se realizará un resumen de los principales resultados, en este capítulo se hace un breve recuento de los objetivos y de los resultados obtenidos, también se realiza una proyección en donde se puede ver los posibles trabajos a futuro para esta tecnología.

[2] El lenguaje Proto-C es una combinación de lenguaje C y funciones propias del modelador OPNET

Capítulo 2

Redes inalámbricas de banda ancha

2.1. Introducción

Desde las décadas finales del siglo XX, las redes de datos han tenido un crecimiento vertiginoso, debido a la utilización cada vez más común de red y de elementos que demandan cada vez más capacidad de la misma. Típicamente la tasa máxima de datos de un acceso de banda ancha compartido para usuarios residenciales y SOHO (*small office/home office*) son alrededor de 5-10 Mbps en el canal de bajada y 0.5-2 Mbps en el canal de subida. Esta asimetría ocurre por la naturaleza del tráfico web y su dominio.

Voz y videoconferencia presentan un tráfico simétrico. Mientras la evolución de los servicios de Internet y el tráfico resultante es difícil de predecir, la demanda de las tasas de datos y la calidad de los servicios se incrementarán dramáticamente en un futuro cercano.

Actualmente el acceso de banda ancha es ofrecido a través de líneas digitales suscriptoras (xDSL, *Digital Subscriber Line*), por cable y acceso inalámbrico de banda ancha (BWA, *Broadband Wireless Access*). Cada una de estás técnicas tiene diferentes costos, desempeño y balance efectivo.

BWA conjunta a los dos de las tecnologías con más crecimiento en los últimos dos años. Ambos, redes inalámbricas y redes de banda ancha tienen en sí mismos la adopción de mercado en masa.

Mientras que el servicio de cable y DSL están siendo efectivas a gran escala, BWA está emergiendo como una tecnología con varias ventajas. Estas incluyen la evasión de los limites de distancias de DSL y altos costos de cableado, rápida ejecución, alta escalabilidad, bajo mantenimiento y costos de actualizaciones e inversiones granulares para ser compatibles con el crecimiento del mercado.

Las redes de acceso inalámbrico son muy atractivas para los nuevos operadores debido a que no es necesaria la infraestructura de una red cableada, una gran ventaja es la rápida implementación y la baja inversión inicial.

2.2. Breve historia de la tecnología BWA

2.2.1. Distribución de video: LMDS, MMDS y DVB

El sistema de distribución local multipunto (LMDS)[22] es un sistema de red de acceso inalámbrico fijo especificado en Estados Unidos por el Council Audio-Visual Digital (Davic), un consorcio proveedor de equipo de video, operadores de redes y otras industrias de telecomunicaciones. Davic fue creada en 1993. LMDS es una tecnología con comunicación punto-a-multipunto de banda ancha. LMDS opera en la banda de frecuencia de 28 GHz en Estados Unidos

El servicio de distribución multipunto y multicanal (MMDS), también conocido como cable inalámbrico es teóricamente una tecnología BWA. Es principalmente utilizada como un método alternativo de televisión por cable. Opera en frecuencias más bajas que LMDS, 2.5 GHz y 2.7 GHz, etc. para tazas de datos más bajas y frecuencias de ancho de banda más pequeños.

Para estandarizar la televisión digital se inició en Europa el proyecto de Broadcasting Digital de Video (Digital Video Broadcasting, DVB). DVB distribuye datos por varios medios: Televisión terrestre (DVC-T), television terrestre para "handhelds"(DVD-H), satélite (DVB-S) y cable (DVB-C). El estándar define la capa física y capa de ligado de datos del sistema distribuido de televisión.

El uso de estas tecnologías de banda ancha se muestra en la figura 2.1.

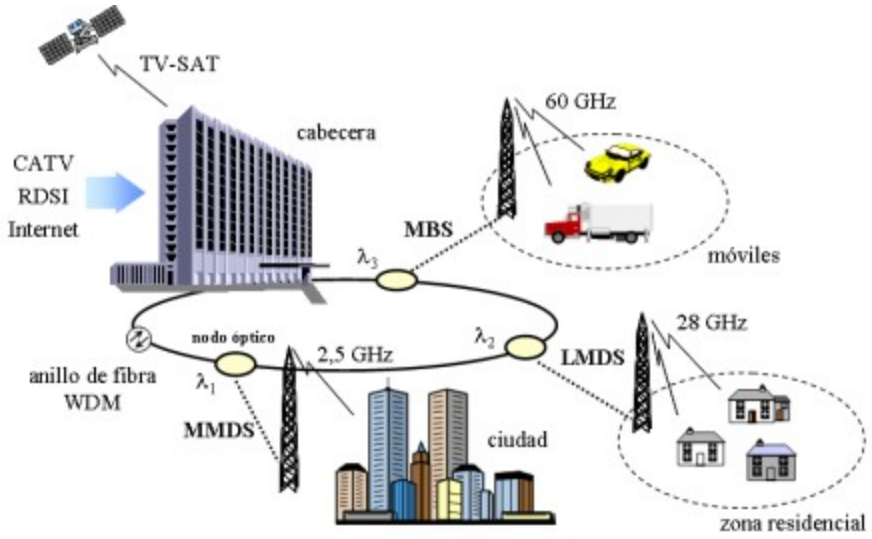


Figura 2.1: Arquitectura de Banda Ancha

2.2.2. Sistemas pre-WiMAX

La primera version del estándar IEEE 802.16 apareció en el 2001. La primera versión completa fue publicada en el 2004. Hubo evidentemente una necesidad de utilizar banda ancha antes de estas fechas. Muchas compañías tenían equipo inalámbrico de banda ancha de tecnología propietario desde los 90ś. Evidentemente estos productos no eran interoperables.

Con la llegada del estándar IEEE 802.16, muchos de estos productos afirmaron basarse en el estándar. Pero no fue posible la verificación y pruebas de interoperabilidad IEEE 802.16/WiMAX hasta el 2006. Estos productos se conocen como productos pre-WiMAX.

2.3. Arquitectura de las redes de banda ancha (BWA)

Un sistema BWA está compuesto por al menos una estación base BS (*Base Station*) y una o más estaciones suscriptoras SS's (*Suscriber Stations*). La BS y las SS's conforman la interfaz aérea del sistema ODU (*Outdoor Unit*), en donde se incluye los transmisores, receptores y antenas. Por otro lado, se encuentra la interfaz alambrica IDU (*Indoor Unit*) que es la interfaz que permite el acceso hacia y desde las redes backbone de datos y PSTN (*Public Switched Telephone Network*). Las dos interfaces se interconectan a una frecuencia intermedia.

La BS asigna un canal de radio a cada una de las SS's de acuerdo a las políticas de control de acceso al medio (MAC). Todos los sistemas BWA constan de tres subsistemas, los cuales se citan a continuación:

1. **Sistema de radiofrecuencia:** Es la parte inalámbrica de alta capacidad que permite la transmisión y recepción de las señales con cada SS. En la estación base, los módulos de transmisión y recepción están conectados en un lado con el equipo de acceso de datos de la BS y por otro lado con una antena cuyas características dependen de los requisitos del sistema. Por otro lado, las estaciones remotas un módulo trasmisor-receptor recibe la señal de bajada de la BS y la pasa al módem de cada SS. Asimismo, recibe el tráfico de la SS y lo trasmite hacia la BS.

2. **Sistema de acceso:** Representa la interfaz de comunicación entre la SS y la BS, es el responsable de la estructuración y manejo del tráfico de señal de bajada o de subida con dirección al sistema de radiofrecuencia. En una red BWA, las SS's comparten en el tiempo los canales de subida UL (*uplink*) y de bajada DL (*downlink*). En el enlace de subida, el canal es usualmente ranurado permitiendo así el acceso al canal mediante la técnica TDMA, mientras que en el canal de bajada se emplea el esquema TDM. Cada SS puede entregar voz y datos utilizando interfaces comunes, por ejemplo, Ethernet, E1/T1, etc.

3. **Interfaz de red en la BS:** Representa la conexión entre el proveedor de servicios y la red IP, las principales aplicaciones que se usan con el sistema de acceso a Internet, acceso a alta velocidad a servicios multimedia, además del acceso a la PSTN para aplicaciones VoIP. En el lado de la SS, es la red de conexión con el sistema inalámbrico.

2.4. Redes inalámbricas y BWA

Los sistemas móviles de comunicaciones han sido testigos de un crecimiento exponencial en demanda durante las últimas décadas. Esta aceptación no había sucedido en los servicios móviles de telefonía. Las aplicaciones de datos móviles fueron iniciadas principalmente para transportar servicios financieros. Actualmente, sin embargo, las aplicaciones en Internet son los principales movilizadores. Este paradigma incrementa la realización de más reglas, que hagan más accesible a las computadoras o aún más rentables para un gran número de gente. Debido al paradigma antes mencionado, algunos centros de voz, redes y sistemas de comunicación de baja tasa de datos han sido mejorados para proveer servicios de altas tasas para servicios móviles de comunicación multimedia en este milenio. ETSIs GSM no han sido la excepción y han experimentado un continúo mejoramiento en tasas bajas de circuitos de datos conmutados CSD (*Circuit-Switched Data*) a través de alta velocidad de circuitos conmutados HSCSD (High-speed Circuit-Switched Data) y de paquetes generales de servicio de radio GPRS (*General Packet Radio Service*) y posteriormente por el mejoramiento de tasa de datos de la evolución de GSM EDGE (*Enhanced Data rate for GSM Evolution*).

2.4.1. 1era Generación (1G)

Apareció en 1979, ésta generación engloba a todas aquellas tecnologías de comunicaciones móviles analógicas con capacidad para transmitir únicamente. La calidad de los enlaces de voz era muy baja (2400 bauds), la transferencia entre celdas era muy imprecisa, tenían baja capacidad (basadas en FDMA) y no existía seguridad. El primer sistema desarrollado y puesto en servicio, fue basado en la normativa NMT-450 (*Nordic Mobile Telephone*) desarrollado en los países nórdicos de Europa. Posteriormente se especificó el AMPS (*Advanced Mobile Phone System*) en los Estados Unidos de América en la banda de los 800 MHz.

Más tarde surgieron toda una serie de estándares diferentes en diversos países: NTT (*Nippon Telegraph and Telephone*) en Japón, y posteriormente dos más, uno como versión del AMPS conocido como TACS (*Total Access Communications System*) desarrollado para Europa en la banda de los 900 MHz y otro NMT-900 basado en su predecesor NMT-450. En las redes 1G cada estación trabaja con un rango de frecuencias, que delimita el número máximo de llamadas simultáneas que puede soportar, puesto que a cada llamada se le asigna un par de frecuencias diferentes: una para cada sentido de la comunicación. Las celdas colindantes no pueden utilizar las mismas frecuencias, para que no se produzcan interferencias. Pero las celdas que están algo más alejadas, si que podrían reutilizar estas frecuencias. Y esto, es lo que se hace. Se parte de una determinada cantidad de frecuencias disponibles. Luego, teniendo en cuenta la densidad estimada de llamadas por área, tanto el tamaño de la celda, como las frecuencias por celda y la reutilización de frecuencias son determinadas.

Cuando DSL y los cable modems comenzaron a ser desarrollados, los sistemas inalámbricos tenían que soportan altas velocidades para ser competitivos. Estos sistemas comenzaron a desarrollarse para frecuencias altas tales como las bandas 2.5GHz y 3.5 GHz. Los sistemas de alta velocidad, conocidos como LMDS (*Local Multipoint Distribution*

System), soportando arriba de varios cientos de megabits por segundo, fueron también desarrollados en bandas de frecuencia con milimetros de ondas, tales como en bandas de 24GHz y 39GHz. Los servicios basados en LMSD estuvieron enfocados en negocios y a finales de los 90's pero el éxito fue corto. Se obtuvieron problemas para acceder a las azoteas para la instalación de antenas, unido al poco rango de capacidad, detuvo su expansión.

En los años 70's, los laboratorios Bell propusieron el concepto celular, un idea mágica que permitía la cobertura tan grande como se necesitara. Desde entonces muchas tecnologías inalámbricas han sido utilizadas, la más exitosa hasta ahora resulto ser GSM (*Global System Mobile*), originalmente fue sistema de segunda generación en europa, GSM es una tecnología utilizada principalmente para transmisión de voz en conjunto con transmisión de datos de baja velocidad como los mensajes SMS (*Short Message Service*).

El GSM actualmente es utilizado en muchos países. Ha ido evolucionando destinado a facilitar relativamente comunicación de datos de alta velocidad en redes basadas en el funcionamiento de GSM. Las evoluciones más importantes, se analizarán en las generaciones posteriores.

2.4.2. 2da Generación (2G)

La segunda generación solucionó los problemas de LOS y proveer mayor capacidad, además de ser digital, haciendo su arribo en la década de los 90s. Las limitaciones de la primera generación de telefonía móvil condujeron al desarrollo de sistemas como GSM , IS-136 TDMA (conocido también como TIA/EIA-136 o ANSI-136), CDMA (*Code Division Multiple Access*) y PDC (*Personal Digital Communications*). Los protocolos empleados en los sistemas de 2G soportan velocidades de información más altas para voz pero limitados en comunicaciones de datos. Se pueden ofrecer servicios auxiliares tales como datos, fax y SMS (*Short Message Service*). El sistema 2G, utiliza protocolos de codificación más sofisticados y son los sistemas de telefonía celular mayormente utilizados aún en la actualidad. La mayoría de los protocolos de 2G ofrecen diferentes niveles de encriptación. En los Estados Unidos y otros países se les conoce como PCS (*Personal Communications Services*).

GSM

GSM es una tecnología móvil de radio que ofrece una solución total de red inalámbrica. A pesar de que fue iniciada por ETSI ha tenido una penetración global y es ampliamente PLMN(Public Land Mobile Network) adoptada. El canal de radio GSM esta especificado para velocidad máxima en la terminal de 250-300 km/h, justificando con esto un alto escalafón en al tecnología. GSM divide su espectro de frecuencia en portadoras de radio frecuencia (RF) de 200-kHz, y cada portadora es dividida en ocho slots de tiempo TDMA (en el canal de físico) de 577-ps de duración. Cada transmisión de slot tiempo 'normal' es de ráfagas 156.25 bits (comprimiendo 3 bits en la cola, 57 bits de payload, 26 de secuencia de entrenamiento de bits alojados en un pequeño espacio (comprimido) de un bit de toggle, 57 bits de payload, 3 bits de cola seguidos por 8.25 bits de guarda en ese orden). Esto da como resultado un promedio de datos de 270.9 kbps por frame. Funcionalmente una red GSM esta dividida en estación móvil MS (*Mobile Station*), subsistema de estación base BSS (Base Station Subsystem) que

consiste en una base de radio transmisora y receptora BTS (*Base Transceiver Station*), la estación base controladora BSC (Base Station Controller), y el centro de servicio conmutado MSC (*Mobile Services Switching Center*). GSM emplea cuatro bases de datos llamadas HLR (*Home Location Register*), VLR (*Visitor Location Register*), AUC (*Authentication Center*) y EIR (*Equipment Identity Register*).

2.4.3. Generación (2.5G)

Aquí se incluyen todas aquellas tecnologías de comunicaciones móviles digitales que permiten una mayor capacidad de transmisión de datos y que surgieron como paso previo a las tecnologías 3G. Muchos de los proveedores de servicios de telecomunicaciones (*carriers*) se moverán a las redes 2.5G antes de entrar masivamente a 3G debido a que la tecnología 2.5G es más rápida y más económica para actualizar a 3G. Una de estas tecnologías 2.5G es GPRS, basada en la transmisión de paquetes y donde los canales de comunicación se comparten entre los distintos usuarios de forma dinámica. GPRS coexiste con GSM, compartiendo gran parte de la infraestructura desplegada en el mismo, pero ofreciendo al usuario un servicio portador más eficiente para las comunicaciones de datos, especialmente en el caso de los servicios de acceso a redes IP como Internet. La velocidad teórica máxima que puede alcanzar GPRS es de 171.2 kbps.

GPRS (*General Packet Radio Service*)

GPRS es una mejora del sistema GSM con la introducción de servicios basados en la técnica de conmutación de paquetes. Este servicio provee eficientemente el uso de los recursos de radio, por medio del acomodo de recursos de datos que son impredecibles por naturaleza, tal como aplicaciones de Internet. La estandarización de las especificaciones GPRS han sido ejecutado por el ETSI, al igual que se han realizado estudios específicos para analizar el comportamiento de GPRS para decidir la calidad del servicio (QoS) o medidas relativas.

Las redes inalámbricas de paquete de datos, como GPRS, son planeadas para soportar una variedad de aplicaciones particulares por diferentes características, tales como navegación en Internet, transferencia de archivos, servicios de correo electrónico, etc.

2.4.4. 3era Generación (3G)

Las tecnologías denominadas de tercera generación (comúnmente llamadas 3G) son un conjunto de nuevos procedimientos de comunicación, estándares y dispositivos que mejorarán la calidad y velocidad de los servicios actualmente disponibles en teléfonos móviles. Las terminales 3G combinan la funcionalidad de un teléfono móvil con la de un PDA y una PC con conexión de banda ancha a Internet.

Mientras que las redes de comunicación, según la ITU (*International Telecommunications Union*), permitirán a los dispositivos preparados para ello transmitir y recibir datos a una velocidad superior a los 144 kbps (en la práctica la tecnología está permitiendo velocidades cercanas a los 384 kbps, muy por encima de los 14.4 kbps de GSM o 53.6 kbps de GPRS).

Algunas de las funcionalidades más interesantes de la tecnología 3G, a parte de mejorar la calidad de las transmisiones de voz, están las capacidades de comunicación instantánea (fax, e-mail, transmisión de grandes archivos, de imágenes, etc.), conexión a Internet con banda ancha, videoconferencia, multimodalidad, capacidades de procesamiento que permitan ejecutar complejas aplicaciones en el teléfono como si fuera un PDA(Personal Digital Assistant), funcionalidades de GPS, sistemas de pago, sistemas de identificación, comunicación con radiofrecuencia, infrarrojos, transmisión por conmutación de paquetes mejor que punto a punto, roaming global, etc. Hay que tener en cuenta que, aunque 3G hace referencia principalmente a la disposición de más ancho de banda para transmitir datos y voz a través de dispositivos móviles, el desarrollo de estas redes coincide con el aumento de las capacidades de procesamiento, memoria y contenidos multimedia de los terminales, lo que conlleva la aparición de paquetes más atractivos a la hora de diseñar nuevos usos en el mundo de la movilidad.

La ITU ha aprobado como estándares oficiales de 3G una serie de sistemas, surgidos de la colaboración entre distintas compañías, agrupados bajo el nombre genérico IMT2000 (International Mobile Telecommunication 2000).

EDGE (*Enhanced Data Rates for Global Evolution*)

EDGE es una tecnología de radio con red móvil que permite que las redes actuales de GSM ofrezcan servicios de 3G dentro de las frecuencias existentes. Como resultado evolutivo de GSM/GPRS, EDGE es una mejora a las redes GPRS y GSM. GPRS es una tecnología portadora de datos que EDGE refuerza con una mejora de la interfaz de radio, y proporciona velocidades de datos tres veces mayores que las de GPRS. Añadir EDGE a la red de GPRS significa aprovechar en toda su extensión las redes de GSM. EDGE puede aumentar el rendimiento de la capacidad y producción de datos típicamente al triple de GPRS, proporcionando así un servicio de 3G espectralmente eficiente. En particular, EDGE permitirá que se exploren todas las ventajas de GSM/GPRS, con el establecimiento de una rápida conexión, mayor amplitud de banda y velocidades en la transmisión de datos medios de 80-130 kbps y tan rápidas como 473 kbps.

Al ser una tecnología de radio de banda angosta (canales de 200 kHz), EDGE permite que los operadores de telecomunicaciones, ofrezcan servicios de 3G sin la necesidad de comprar una licencia 3G. Al desarrollar la infraestructura inalámbrica ya existente, EDGE permite que los proveedores de servicios brinden al mercado servicios de 3G en un lanzamiento rápido.

En la mayoría de los casos sólo se necesitan cambios secundarios para pasar de GPRS a EDGE. Además, EDGE reduce el costo al implementar sistemas de 3G a nivel nacional porque está diseñada para integrarse a una red de GSM ya existente. Así, EDGE representa una solución fácil, incremental del costo, una de las rutas más rápidas a desplegar los servicios de 3G.

CDMA 2000

CDMA 2000 parte del legado de la tecnología CDMAONE para ofrecer servicios de datos eficientes y de gran capacidad de voz en una cantidad mínima de espectro. Permite mayores velocidades de datos de usuario y un uso más eficiente del espectro de radio que las técnicas de radio actuales.

CDMA 2000 es una tecnología de interfaz de radio compatible con el estándar IMT-2000 y ofrece una evolución simple y eficaz en función de costos hacia 3G para operadores internacionales que operan actualmente redes CDMAONE, CDMA 2000 ofrece mejoras en la calidad y capacidad de voz respecto a CDMAONE, junto con servicios de datos multimedia de alta velocidad. La norma CDMA 2000 está dividida en fases para permitir la pronta instalación de la nueva tecnología. Este enfoque permite a los operadores introducir más capacidad para servicios de voz junto con incrementos en la velocidad de los datos en intervalos que coinciden con la demanda emergente del mercado. La primera fase de CDMA 2000 o CDMA 2000 1x realiza la transmisión de datos a 144 kbps. La fase dos, llamada CDMA 2000 1xEV provee transmisiones mayores a los 2 Mbps.

CDMA 2000 1X

El estándar IS-2000 (CDMA2000 1X) fue publicado por la TIA (*Telecommunications Industry Association*). 1X ofrece aproximadamente el doble de capacidad para voz que CDMAONE. Las transmisiones de datos promedio son de 144 kbps. CDMA 2000 1X se refiere a la implementación de CDMA 2000 dentro del espectro existente para las portadoras de 1.25MHz de CDMAONE. El termino técnico se deriva de N = 1 (es decir, el uso de la misma portadora de 1.25MHz de cdmaOne) y el 1X significa una vez 1.25MHz.

CDMA 2000 1xEV

La evolución de CDMA 2000 después de 1x es conocida como CDMA 2000 1xEV. Esta versión está dividida en dos categorías: 1xEV-DO (*1X Evolution Data Only*) y 1xEV-DV (*1X Evolution Data and Voice*). La primera mejora el volumen de transmisión de datos y alcanza velocidades pico de 2.4 Mbps sin requerir más de 1.5 MHz de ancho de banda. Esta fase está optimizada para lograr un enfoque eficiente que reúne los mejores esfuerzos para la entrega de datos. Mientras tanto, 1xEV-DV, se centra en las capacidades de voz y datos en tiempo real y en los aumentos de rendimiento para lograr eficiencia tanto de voz como de datos.

WCDMA

WCDMA es la tecnología de interfaz de aire en la que se basa la UMTS (*Universal Mobile Telecommunications Service*), el cual es un estándar europeo 3G para los sistemas inalámbricos. Se trata de una tecnología pensada para ofrecer elevados anchos de banda de voz y datos, alcanzando velocidades de hasta 2 Mbps, adecuados para aplicaciones tales como videoconferencia.

Esta tecnología, constituye una buena elección cuando se piensa a medio-largo plazo, ya que ofrece mayores posibilidades aunque también es más compleja. La razón de que la especificación WCDMA sea algo más compleja y menos eficiente de lo que debería puede estar en la participación de multitud de agentes tecnológicos diferentes en su desarrollo, principalmente fabricantes de equipos, lo que obliga a adoptar numerosas soluciones de compromiso en su implementación.

La tecnología WCDMA está altamente optimizada para comunicaciones de alta calidad de voz y comunicaciones multimedia, como pueden ser las videoconferencias. También es posible acceder a diferentes servicios en una sola terminal, por ejemplo, podemos estar realizando una videoconferencia y al mismo tiempo estar haciendo una descarga de archivos muy grande, etc.

UMTS HSPDA

UMTS (*Universal Mobile Telecommunications Service*) es una tecnología de voz y datos a alta velocidad que integra la familia de normas inalámbricas de tercera generación. La tecnología radial utilizada en UMTS es la WCDMA (*Wideband Code Division Multiple Access*), que entrega velocidades de datos pico de hasta 2 Mbps.

UMTS utiliza una combinación de las tecnologías Acceso Múltiple por División de Código(CDMA) y Acceso Múltiple por División de Tiempo (TDMA) para hacer un uso altamente eficiente del espectro. El HSDPA (High Speed Downlink Packet Access) es la primera evolución 3G, del UMTS/WCDMA.

Una tecnología de transmisión, que puede dar una velocidad pico teórica de hasta 14 Mbps y soportar tasas de throughput promedio cercanas a 1 Mbps. HSDPA lleva a WCDMA a su máximo potencial en la prestación de servicios de banda ancha, y la capacidad de datos celulares definida con throughput más elevado. De la misma manera en que EDGE incrementa la eficiencia espectral en comparación con GPRS, HSDPA incrementa la eficiencia espectral en un factor de hasta 3.5 veces en comparación con WCDMA.

La eficiencia espectral y las velocidades aumentadas no sólo habilitan nuevas clases de aplicaciones, sino que además dan soporte para que la red sea accedida por un mayor número de usuarios. HSDPA alcanza sus elevadas velocidades mediante las mismas técnicas que amplifican el rendimiento de EDGE superando a GPRS. Estas incluyen el agregado de modulación de mayor orden (Modulación de Amplitud en Cuadratura 16-16 QAM), codificación variable de errores, y redundancia incremental, así como el agregado de nuevas y potentes técnicas tales como programación rápida. Además, HSDPA emplea un eficiente mecanismo de programación para determinar qué usuario obtendrá recursos. Finalmente, HSDPA comparte sus canales de alta velocidad entre los usuarios del mismo dominio de tiempo, lo que representa el enfoque más eficiente.

2.5. Algunos estándares IEEE relacionados a BWA

2.5.1. IEEE 802.11 (WiFi)

WiFi (*Wireless Fidelity*) es el nombre popular del estándar IEEE 802.11 para redes inalámbricas de área local (WLAN) que operan utilizando espectro sin licencia en la banda de 2.4 Ghz. La generación actual de WLAN soporta velocidades para datos de hasta 11 Mbps dentro de los 90 metros a partir de la estación base en donde se transmite la señal.

La aparición repentina de 802.11 se llevo a cabo como resultado como de una decisión tomada en 1985 por la comisión federal de comunicaciones FCC (*Federal Communications Commission*) de liberar varias bandas del espectro inalámbrico para la utilización sin licencia del gobierno.

En 1990, un nuevo comité IEEE llamado 802.11 fue establecido para buscar el comienzo de un estándar. Sin embargo, no fue hasta 1997, ocho o nueve años después, que este nuevo estándar fue publicado (aunque dispositivos con pre-estándar se encontraban ya en el mercado).

Dos variantes fueron ratificadas en los dos siguientes años 802.11b el cual opera en la banda de la industria médica y científica ISM (*Industry Medical and Scientific*) de 2.4 GHz y 802.11a el cual opera las bandas de 5.3 GHz y 5.8 GHz de la infraestructura de información nacional sin licencia.

La popularidad de 802.11 despegó con el crecimiento con el acceso a Internet de banda ancha a alta velocidad en el hogar. Fue de esta manera y permanece la forma fácil de compartir un enlace de banda ancha entre varios equipos esparcidos en una casa. El crecimiento de *hotspots*, y puntos de acceso access points han agregado popularidad ha 802.11. La última variante fue 802.11g. Esta tecnología 802.11, como 802.11a, utiliza una forma más avanzada de modulación llamada OFDM (*Orthogonal Frequency-Division Multiplexing*), esta habilitada para ser utilizada en la banda de 2.4 GHz. 802.11 puede alcanzar una velocidad de hasta 54 Mbps.

En una red WiFi, cada componente, ya sean estaciones o puntos de acceso AP (*Access Point*), requieren un radio transmisor y una antena. Las estaciones pueden ser incorporadas a una tarjeta LAN instalada en una PC de escritorio, un adaptador USB, una PCMCIA o puede ser integrada a un dispositivo PDA.

Actualmente, 802.11a/b/g para redes inalámbricas de área local proveen un desempeño adecuado para las aplicaciones que hoy en día se utilizan, por lo tanto la conveniencia de las conexiones inalámbricas poseen un gran valor. Las siguientes generaciones de aplicaciones inalámbricas requerirán altas tasas de transmisión de datos (*higher WLAN data throughput*) y por lo tanto la gente demandará mayor rango. En respuesta a estas necesidades el grupo de trabajo IEEE 802.11n Task Group al igual que las alianzas Wi-Fi se encuentran trabajando en ello.

El objetivo del grupo de trabajo 802.11n es definir las modificaciones a la capa física y a la capa de control de acceso al medio PHY/MAC (*Physical Layer and Media Access Control Layer*) que proporciona un mínimo de la máxima capacidad de transmisión a 100 Mbps en el punto de acceso al servicio de la MAC SAP (Service Access Point). El requerimiento mínimo de la máxima capacidad de transmisión aproximadamente cuadruplica el desempeño de la máxima capacidad de transmisión WLAN comparada con las redes de hoy en día 802.11a/g. El objetivo principal sobre de la máxima capacidad de transmisión aérea es exceder 200 Mbps para alcanzar los requerimientos de máxima capacidad de transmisión de 100 Mbps MAC SAP. Otras mejoras necesarias incluyen un rango de acuerdo a la máxima capacidad de transmisión dada, que sea robusto respecto a la interferencia y una mejora y más servicios uniformes dentro del área de cobertura de un punto de acceso AP. Canales amplios de ancho de banda y múltiples configuraciones de antenas que puedan permitir tasas de datos de 500 Mbps.

Mientras la trasferencia de datos y audio han tenido una atracción primaria para la mayoría de los usuarios 802.11, la siguiente generación de aplicaciones será el video. En el futuro es Internet una herramienta para bajar películas. Poder ver streamig video en una computadora portátil o un PDA sobre una conexión inalámbrica. Actualmente, 802.11g es capaz de soportar streaming para video, pero los problemas respecto a la calidad de servicio aún dañan el desempeño. La imagen de video puede ser (jittery and halts) cuando la red se encuentra saturada. El grupo de trabajo de las especificaciones de los servicios multimedia 802.11e ha desarrollado uno fijo. Una velocidad mejorada llamada tecnología de ráfagas de paquetes que incrementa la máxima capacidad de transmisión del canal permitiendo que múltiples paquetes viajen a través de las ondas aéreas sin el encabezado extra de espacio entre los paquetes, esto incrementa la velocidad de la red. Las especificaciones 802.11e toma lugar la capa MAC y por ende será común la capa física (PHY) para todas las tecnologías WLAN 802.11. Consecuentemente, 802.11e las ráfagas de paquetes (packet-bursting) estarán disponibles en 802.11g.

2.5.2. IEEE 802.16 (WiMAX, *Worldwide Interoperability for Microwave Access*)

WiMAX es el nombre que se le dio al estándar IEEE 802.16 que describe la interfaz aérea para sistemas fijos de acceso inalámbrico de banda ancha. El objetivo de este nuevo protocolo es reemplazar o competir directamente con el Internet por Cable y ADSL (*Asymmetric Digital Subscriber Line*), ya que mediante una sola torre se tendrá cobertura de hasta 50 kilómetros a tasas de transmisión de hasta 75 Mbps. Por su capacidad, fue diseñado como una solución de último kilómetro, es decir, el último tramo en redes metropolitanas (MAN). Actualmente, lo impulsan 130 organizaciones (fabricantes de equipo y componentes, proveedores de servicio, desarrolladores de software, etc.), entre los que destaca Intel Corporation, Nokia y Siemens.

El estándar IEEE 802.16 es realmente una especificación para sistemas inalámbricos de acceso de banda ancha que usan una arquitectura de punto a multipunto (PMP). La primera versión del estándar 802.16 lanzó ambientes de línea de vista LOS direccionados en bandas

de frecuencia alta que operan en el rango de 10-66 GHz, mientras que la rectificación del estándar 802.16a fue diseñado para sistemas que operen en bandas entre 2 GHz y 11 GHz. La significante diferencia entre estas dos bandas de frecuencia se encuentra en la habilidad de soportar la operación de NLOS (*Non-Line-of-Sight*) en las frecuencias más bajas, algo que no es posible en las frecuencias altas.

El estándar IEEE 802.16 cubre más aspectos que HIPERACCESS, incluyendo Redes WMAN (*Wireless Metropolitan Area Network*) y HUMAN (*High-Speed Unlicensed Metropolitan Area Networks*). El estándar inicial 802.16 opera en la banda de frecuencias de 10 a 66 GHz. En estas frecuencias, IEEE 802.16, requiere de una línea de vista directa entre los transmisores y receptores. Esto reduce la distorsión multitrayectoria, lo que aumenta el ancho de banda. Teóricamente IEEE 802.16 puede proveer velocidades de transmisión arriba de 75 Mbps en los canales downlink y uplink.

Los proveedores pueden usar múltiples canales para una transmisión simple para proveer velocidades arriba de los 350 Mbps. IEEE 802.16 divide su MAC en diferentes subcapas que soportan diferentes tecnologías de transporte, incluyendo IPv4, IPv6, Ethernet y ATM.

La IEEE ha desarrollado 802.16a para el uso de bandas de frecuencias en el rango de 2 a 11 GHz. En las frecuencias bajas, las señales pueden penetrar barreras y no se requiere una línea de vista entre el transmisor y la antena. El estándar IEEE 802.16a soporta un desarrollo tipo malla, en donde una BS puede pasar una comunicación simple a otra BS extendiendo así el rango de funcionamiento del protocolo.

La extensión IEEE 802.16b incrementa el espectro y tecnología que puede usarse en las bandas de frecuencia de 5 a 6 GHz y provee calidad de servicio (QoS). Proveer calidad de servicio asegura la prioridad de transmisión para la transmisión de voz y video en tiempo real y ofrece diferentes niveles de servicio a diferentes tipos de tráfico. La IEEE 802.16c representa para la banda de 10 a 66 GHz el perfil de sistema que estandariza más detalles de la tecnología, asegura mayor consistencia para la implementación e interoperabilidad del sistema.

En IEEE 802.16d se incluyen pequeños detalles que ayudan a mejorar la extensión 802.16a. Esta extensión también crea los perfiles del sistema para lograr que los dispositivos basados en 802.16a sean probados.

La tecnología basada en IEEE 802.16e estandariza la intercomunicación entre las estaciones base fijas y los dispositivos móviles. IEEE 802.16e habilitará las funciones de hand-off de alta velocidad necesarias para la comunicación con usuarios moviéndose sobre automóviles a altas velocidades.

La Unión Internacional de Telecomunicaciones (ITU) inició un grupo de trabajo llamado ITU JRG 8A-9B para encargarse de la estandarización de los sistemas BWA. Este grupo recibe información por parte de BRAN y de IEEE 802.16 y trata de desarrollar una conciencia global sobre esta tecnología en base al punto de vista y función de la ITU [21].

Estándar	Descripción
802.16f-2005	Administración de una red WiMAX
802.16g-2007	Servicios y Procedimientos de Administración
802.16k-2007	Puenteo en 802.16
802.16h	Mecanismo de coexistencia para mejorar la operación
802.16i	Administración de una red WiMAX móvil
802.16j	Especificación de Multihop Relay
802.16Rev2	Consolida 802.16-2004, 802.16e, 802.16f, 802.16g y posiblemente 802.16i dentro de un nuevo documento
802.16m	Interfaz área avanzada

Tabla 2.1: Otros Estándares IEEE 802.16

En la tabla 2.1 se presentan una serie de estándares IEEE 802.16 que se han terminado recientemente o se encuentran aún en fase de desarrollo.

2.5.3. IEEE 802.20 (MBWA, *Mobile Broadband Wireless Access*)

El estándar 802.20 promete una combinación de muchas características deseadas del estándar 802.16e con aquellas de las redes de datos celulares de 3G, mientras reduce las limitaciones de ambas para dichas modalidades. Por lo tanto, las soluciones 802.20 podrá implementar la necesidad de un espectro amplio funcional para negocios móviles e implementación personal de computadoras.

En diciembre 2002, el *IEEE Standards Board* aprobó el establecimiento de IEEE 802.20, el grupo de trabajo para acceso móvil inalámbrico de banda ancha MBWA Working Group.

La misión de IEEE 802.20 es desarrollar las especificaciones para una eficiente interfase aérea basada en paquetes, para optimizada para el transporte de servicios basados en IP. El objetivo es habilitar el desarrollo worldwide sostenible, presente en todos lados, interoperando siempre para multiples vendedores de acceso a redes inalámbricas de banda ancha que permita conocer las necesidades del mercado de usuarios de negocios y de usuarios finales residenciales.

La siguiente generación de conectividad móvil detrás de WiMAX estará basada en el estándar 802.20, el cual esta siendo desarrollado para la tierra como una especificación móvil. La capa física y la capa de enlace de datos son diseñadas específicamente para los requerimientos móviles, tales como arreglos de antenas adaptativas. El concepto que sustenta el concepto de adaptativo, ó de antenas 'inteligentes', es proveer el procesamiento de la señal que permita señales de múltiples antenas, como en una red celular, para trabajar juntas producir una señal fuerte con un mínimo de interferencia. Las especificaciones 802.20 es el primer estándar IEEE que explícitamente plantea la necesidad de clientes móviles en vehículos también en movimiento.

2.5.4. IEEE 802.21 (MIH, *Media Independent Handover*)

IEEE 802.21 es un nuevo estándar, que es interesante sin duda para los equipos de telecomunicaciones para tener la posibilidad de realizar handover entre dos tecnologías inalámbricas.

El estándar IEEE 802.21 analiza habilitar handover e interoperatibilidad entre diferentes tipos de redes, lo cual es conocido como MIH. Los tipos de redes pueden estar o no dentro de los estándares IEEE. Por ejemplo IEEE 802.21 podría permitir handover entre redes 3G y 802.11/WiFi.

Capítulo 3

Protocolo IEEE 802.16 con arquitectura Mesh

3.1. Introducción

El estándar IEEE 802.16, conocido comúnmente como WiMAX ha sido diseñado específicamente para proveer acceso a redes inalámbricas de banda ancha en las Redes de Área Metropolitana (MAN), para ofrecer un desempeño comparable al que ofrece las redes tradicionales con cable, DSL o TI.

En el año 2000, el grupo IEEE inició el IEEE 802.16 *Working Group* para crear estándares que regulen el acceso inalámbrico de banda ancha con el objeto de ofrecer una solución escalable, de bajo costo y alta capacidad, la cual incluye tanto usuarios como tasa de transmisión, para extender el backbone de fibra óptica.

El estándar IEEE 802.16 cuenta con distintas versiones de los que sobresalen el 802.16 que emplea el espectro entre 10 y 66 GHz y que requiere línea de vista LOS (*Line of Sight*) entre el transmisor y el receptor, IEEE 802.16a [3] que opera en la banda de frecuencias entre 2 y 11 GHz, y el 802.16e [4] que está enfocado en proveer movilidad al estándar y que funciona en bandas menores de 6 GHz. Las dos últimas versiones mencionadas con anterioridad, no requieren LOS.

El estándar IEEE 802.16 permite alcanzar tasas de transferencia máxima que fluctúan entre 32 y 130 Mbps dependiendo del ancho del canal de frecuencia y de la técnica de modulación. Adicionalmente se destaca que el estándar IEEE 802.16 posee la capacidad de utilizar ventajas ofrecidas por los sistemas de antenas avanzados que actualmente se encuentran disponibles. Un servicio inalámbrico 802.16 ofrece un camino de comunicaciones entre los abonados y la red central a la que proporciona acceso, ya sea la red telefónica pública o Internet, ofreciendo una interfaz aire entre las estaciones de los suscriptores o abonados SS (*Suscriber Stations*) y la estación base BS (*Base Station*). Los protocolos definidos específicamente para la transmisión inalámbrica dan respuesta a todos los aspectos relacionados con la transmisión de bloques de datos por una red.

En este capítulo se presenta un entorno general del protocolo de comunicaciones de acceso inalámbrico de banda ancha IEEE 802.16-2004 [2].

3.2. Antecedentes

Durante los últimos años ha crecido el interés por las distintas modalidades de redes de acceso inalámbricas, conocidas como WLL (*Wireless Local Loop*) como una alternativa al bucle local de cobre tradicional. A fin de proporcionar un enfoque estandarizado a estas diversas tecnologías, el comité IEEE 802 estableció el grupo de trabajo 802.16 en 1999 con el cometido de normalizar los accesos inalámbricos de banda ancha.

IEEE 802.16, también conocido como WiMAX (*Worldwide Interoperability for Microwave Access*), estandariza tanto la interfaz aérea como las funciones relacionadas con los bucles locales inalámbricos o bucles inalámbricos fijos. Para ello se crearon tres distintos grupos de trabajos: IEEE 802.16.1, 802.16.2 y 802.16.3. El primero tiene como objetivo la interfaz aérea para 10-66 GHz, mientras que el segundo está centrado en la coexistencia de los diversos sistemas de acceso inalámbrico de banda ancha. IEEE 802.16.3, finalmente, se ocupa de la interfaz de aire para frecuencias de 2-11 GHz, que requieren licencias. De todos ellos, los desarrollos del grupo 802.16.1 serán los que generen mayor interés en la industria, dado que se centran en las bandas de frecuencias disponibles.

Históricamente, las actividades 802.16 fueron iniciadas en agosto 1998, en una reunión por parte de N-WEST (*Nacional Wireless Electronics Systems Tested*) de la U.S. *Nacional Institute of Standard and Technology*. El esfuerzo fue respaldado en IEEE 802, la cual abrió el grupo de estudio.

El desarrollo de el estándar 802.16 es responsabilidad de el grupo de trabajo 802.16 (*IEEE Working Group Access (BWA) Standards*). El interés inicial del grupo fue el rango de frecuencias 10-66 GHz. El proyecto 2-11 GHz de el grupo IEEE 802.16a fue aprobado en Marzo 2000. El proyecto 802.16a en la etapa primaria cubría el desarrollo las especificaciones de la capa física, el cual soportaba el mejoramiento de la MAC básica. Adicionalmente, el grupo de trabajo había completado el estándar IEEE Standard 802.16.2 para direccionar la coexistencia 10-66 GHz y a través del proyecto 802.16.2a, la recomendación se expandió para incluir las bandas bajo licencia de 2 a 11 GHz.

3.3. Evolución

El estándar 802.16-2001 [1] completado en octubre del año mencionado y publicado en abril 8 de 2002 define las redes inalámbricas MAN (*WirelessMAN TM*). La finalización de este estándar anuncio la era del acceso inalámbrico de banda ancha como una gran herramienta para el mejoramiento de enlaces caseros y de negocios hacia redes de telecomunicaciones en el mundo.

Actualmente la definición del estándar 802.16, es una red inalámbrica MAN que provee el acceso a redes a través de antenas exteriores que se comunican con estaciones radio base. Las redes inalámbricas MAN ofrecen una alternativa a las redes de acceso cableadas, tales como enlaces de fibra óptica, sistemas de cable modems usando cable coaxial y enlaces DSL. Esto es debido a que los sistemas inalámbricos tienen la capacidad de direccionarse en áreas geográficas sin el costo de la infraestructura y el desarrollo que requiere el cableado individual, esta tecnología provee de un costo menor para desarrollarse y puede ser más omnipresente el acceso de banda ancha.

El estándar IEEE 802.16 fue diseñado para evolucionar como un conjunto de interfaces aéreas basadas en un protocolo MAC común pero con especificaciones en la capa física dependientes del espectro en uso así como de la asociación reguladora. El estándar fue aprobado en 2001 las frecuencias son 10 a 66 GHz, donde el espectro esta disponible en todo el mundo pero que las el tamaño de las ondas es corto y esto introduce retos significativos que desarrollar. En la tabla 3.2 se resume las principales características de los diferentes pasos en la evolución del estándar.

3.4. Entorno de operación del protocolo IEEE 802.16

IEEE 802.16 opera en 10-66 GHz para Linea-de-Vista (LOS), y 2-11 GHz para conexiones sin LOS. En la capa física, el estándar emplea Multiplexación por División de Frecuencias Ortogonales (OFDM), y soporta modulación adaptativa y cuentan con codificación en las condiciones del canal, provee un data rate por encima de los 134 Mbps (por estación base) en cada canal de 28 MHz.

Una red IEEE 802.16 consiste de una Estación Base (BS) y multiples estaciones suscriptoras (SSs). La BS actúa como un gateway para las SSs para la red externa, y cada SS actúa como un punto de acceso que agrega tráfico desde los usuarios finales en cierta área geográfica.

Una red que utiliza un medio compartido debe proveer un mecanismo eficiente para compartir este medio. PMP y Mesh son ejemplos de topologías de redes inalámbricas para la distribución del medio en redes inalámbricas. En este contexto el medio de propagación es el espacio a través de ondas de radio.

El estándar IEEE 802.16 describe un control de acceso al medio que puede compartir el canal de radio entre cientos de usuarios. Éste estándar usa un mecanismo del tipo solicitud/entrega (Request/Grant Protocol) similar al utilizado por los sistemas de cable modem (DOCSIS)[20]. El acceso al medio se puede configurar por TDD, con un único canal de comunicación, como por FDD con varios canales en distintas portadoras. Para ambos casos, el protocolo de acceso utiliza TDMA el cual tiene una ranura el tiempo tanto en el canal de subida como en el de bajada. Para el caso de TDMA sobre FDD, 802.16 permite una comunicación *Full Duplex*, mientras que para TDMA sobre TDD la comunicación es *Half-Duplex.* En ambos casos el sistema permite reservar un determinado ancho de banda para aplicaciones que necesitan ancho de banda garantizado como VoIP, video, etc.

	802.16	**802.16-2004**	**802.16e-2005**
Status	Completado Diciembre 2001	Completado Junio 2004	Completado Diciembre 2005
Banda de Frecuencia	10GHz-66GHz	2GHz-11GHz	2GHz-11GHz para fijo 2GHz-6GHz para aplicaciones móviles
Aplicación	LOS Fijo	NLOS Fijo	Fijo y móvil NLOS MAC
Arquitectura	PMP, Mesh	PMP, Mesh	PMP, Mesh
Esquema de transmisión	Portadora sencilla	Portadora 256 OFDM o 2,048 OFDM	Portadora 256 OFDM o OFDM escalable con 128, 512, 1,024, o 2,048 subcarriers
Modulación	QPSK, 16 QAM, 64 QAM	QPSK, 16 QAM, 64 QAM	QPSK, 16 QAM, 64 QAM
Tasa de datos burda	32Mbps-134.4Mbps	1Mbps-75Mbps	1Mbps-75Mbps
Trama Multiplexaje	TDM/TDMA	TDM/TDMA/ OFDMA	TDM/TDMA/ OFDMA
Bidireccional	TDD and FDD	TDD and FDD	TDD and FDD
Channel bandwidths	20MHz, 25MHz, 28MHz	1.75MHz, 3.5MHz, 7MHz, 14MHz, 1.25MHz, 5MHz, 10MHz, 15MHz, 8.75MHz	1.75MHz, 3.5MHz,7MHz, 14MHz, 1.25MHz, 5MHz, 10MHz, 15MHz, 8.75MHz
Designación interfaz-aérea	WirelessMAN-SC	WirelessMAN-SCa WirelessMAN-OFDM WirelessMAN-OFDMA WirelessHUMANa	WirelessMAN-SCa WirelessMAN-OFDM WirelessMAN-OFDMA WirelessHUMANa
Implementación WiMAX	None	256 - OFDM WiMAX Fijo	OFDMA escalable WiMAX Móvil

Tabla 3.1: Principales características de estándares IEEE 802.16

3.4.1. Capa física

La capa física de WiMAX esta basada en OFMD (multiplexación por división de frecuencias ortogonales) , que es un esquema de transmisión a altas velocidades.

Las capas físicas definidas en el IEEE 802.16 son:

WirelessMAN SC. Una portadora planeada para frecuencias más allá de los 11GHz requiere LOS. Este tipo de capa física esta contemplada en la especificación original del IEEE 802.16.

WirelessMAN SCa. Una portadora para frecuencias entre 2GHz y 11 GHz para operación punto-a-multipunto.

WirelessMAN OFDM. Basada en OFDM con FFT de 256 puntos para operaciones punto-a-multipunto en condiciones sin LOS a frecuencias entre 2GHz y 11GHz. La capa física especificada en IEEE 802.16-2004, ha sido aceptada por WiMAX en operaciones fijas, a menudo se hace referencia a este tipo como **WiMAX fijo**.

WirelessMAN OFDMA. Basada en OFDMA con FFT de 2048 puntos para operaciones punto-a-multipunto en condiciones NLOS a frecuencias entre 2GHz y 11GHz. En la especificación IEEE 802.16-2005 , esta capa ha sido modificada a SOFDMA (*Scalable OFDMA*), donde el tamaño FFT es variable y puede tomar cualquiera de los siguientes valores: 128, 512, 1024 y 2048. El tamaño variable FFT permite una operación e implementación optima en los sistemas con un rango amplio de canales en condiciones de ancho de banda y radio. Ha sido aceptada por WiMAX para operaciones móviles y portables, también se hace referencia a esta capa como **WiMAX móvil**

En el presente trabajo se utilizará la tercera especificación de capa física ya que se trata de un sistema WiMAX fijo.

3.4.2. Capa MAC

La tarea principal de la capa MAC de WiMAX es proporcionar una interfaz entre las capas superiores de transporte y la capa física- La capa MAC toma los paquetes de la capa superior llamados unidades de servicio de datos MAC (MSDUs) y se organizan dentro de unidades de datos (MPDUs) para la transmisión aérea. El MAC PDU (*MAC Protocol Data Unit*) es la unidad básica de comunicación intercambiada entre la capa MAC de la estación base y las estaciones suscriptoras.

Para un eficiente uso de los recursos físicos, múltiples MAC PDUs destinados al mismo receptor puede ser concatenado y portado sobre una sola oportunidad de transmisión o región de datos como se muestra en la figura 3.1. Para habilitar conexiones ARQ, el SDU es primero particionado dentro de bloques ARQ de tamaño fijo, y un número secuencial es asignado a cada bloque ARQ. Un ACK selectivo para cada BSN indica que el bloque ARQ se ha recibido sin errores.

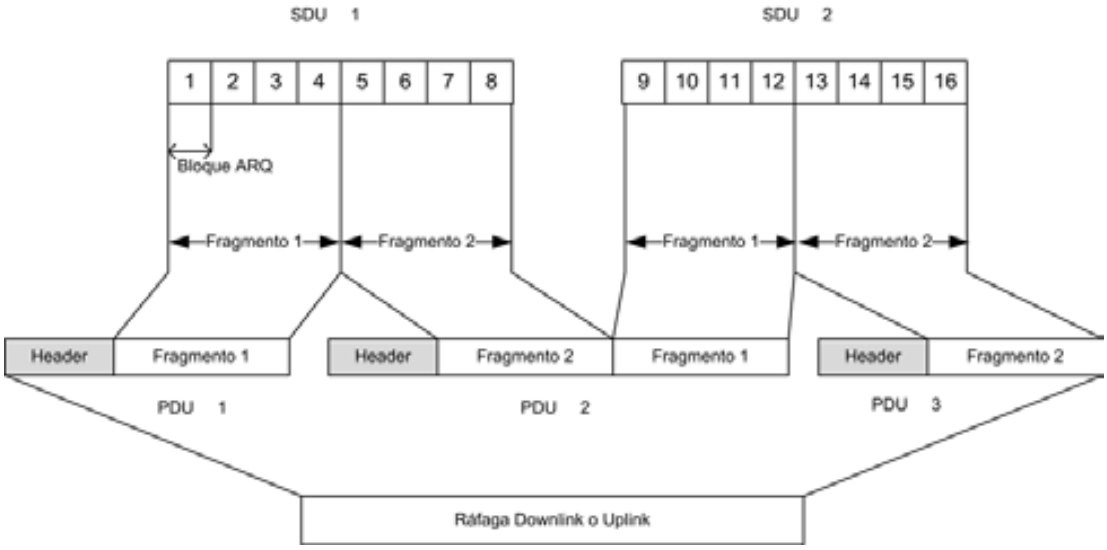


Figura 3.1: Segmentación y Partición de SDU en MAC PDU

El MAC PDU posee tres campos como se puede apreciar en la Figura 3.2. El primer campo es de tamaño fijo y corresponde al encabezado genérico (*Generic MAC Header*), el segundo campo es opcional, de longitud variable, y corresponde a la carga útil o de datos (*Payload*) que puede tener sub-encabezados, mientras que el tercer campo correspondiente al CRC (*Cyclic Redundancy Check*), es también opcional y protege tanto al *Generic MAC Header* como al *Payload*.

Figura 3.2: MAC PDU

3.5. Topologías WiMAX

IEEE 802.16 soporta dos modos de operación de acuerdo a su topología:

- PMP Punto-a-Multipunto
- Topología Mesh o modo Mesh.

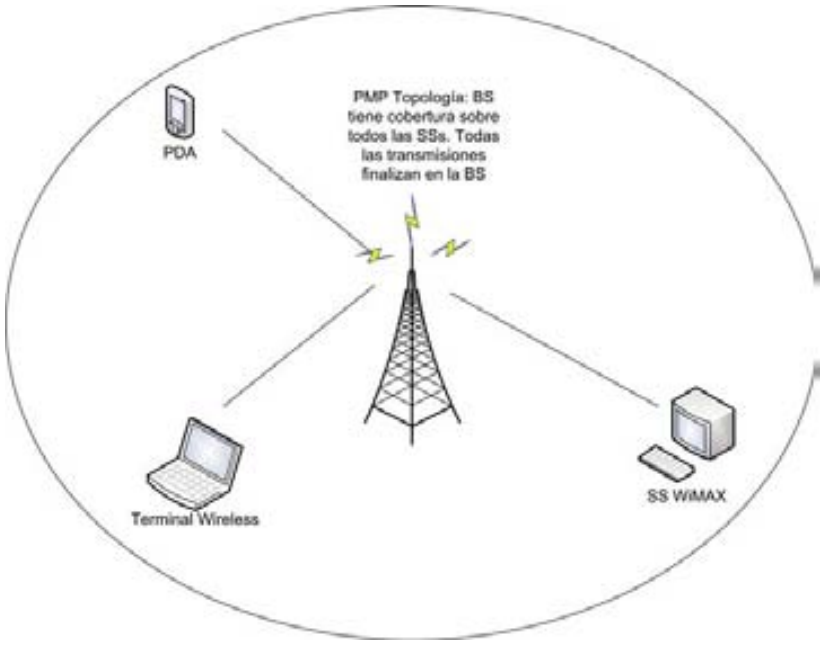


Figura 3.3: Topología PMP

3.5.1. PMP

En PMP cada SS comunica directamente con la BS a través de un enlace de un solo salto, el cual requiere que todos los nodos o estaciones suscriptoras estén en rango de transmisión de la BS, es decir, PMP es una topología centralizada donde la BS es el centro del sistema.

El *downlink* , de la BS a el usuario, opera en una base Punto-a-Multipunto. El radio link del IEEE 802.16 opera como ya se ha mencionado con una BS central y una antena sectorizada que es capaz de manejar multiples sectores independientes simultáneamente. Sin un canal de frecuencia dada y sin un sector de antena, todas las estaciones reciben la misma transmisión o parte de las mismas. La BS solo opera en esta dirección sin tener que coordinarse con otras estaciones salvo que se utilice TDD que puede dividir el tiempo en uplink y *downlink* en periodos de transmisión por tiempo. El *downlink* generalmente es generalmente *broadcast*.

Las estaciones suscriptoras (SSs) están identificadas por una dirección única de 48 bits de la capa MAC. Esta dirección permite distinguirlas tomando en cuenta tanto al fabricante como el tipo de equipo, por consiguiente se les considera como direcciones universales. La dirección MAC se emplea durante los procesos de registro así como en los de autenticación. Durante el registro, esta dirección permite a la BS el establecimiento de todas las conexiones apropiadas que correspondan a una SS en particular. Por otro lado, durante el proceso de autenticación la dirección MAC es requerida para la identificación mutua tanto de la BS como de la SS. La proporción de servicios de conectividad es posible mediante conexiones realizadas entre la BS y la SS. Dichas conexiones se identifican mediante el CID (*Connection*

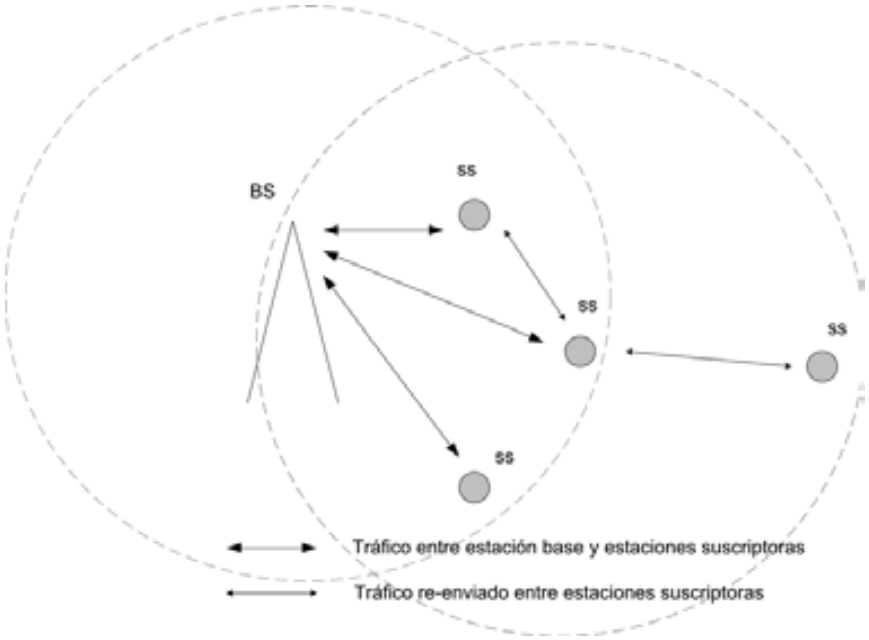


Figura 3.4: Topología Mesh

ID), que consta de 16 bits, permitiendo hasta 65 536 conexiones en cada canal ascendente y descendente. Durante la inicialización de los servicios, la BS proporciona 3 conexiones en ambas direcciones mediante los mensajes de control RNG-REQ (*Ranging Request*) y REG-RSP (*Registration Response*).

El CID funciona como un apuntador hacia el destino de cada conexión y también proporciona información acerca del contexto en la que se realiza. El tipo de servicio que la BS brinda está implícito en el CID. Adicionalmente, cada solicitud de asignación de ancho de banda que realizan las SSs está basada en el CID, por lo que el ancho de banda disponible será distinto para conexiones diferentes, es posible que una sola conexión soporte tráfico proveniente de distintas sesiones de capas superiores.

3.5.2. Mesh

La principal diferencia entre PMP y modo Mesh es que en el modo PMP el tráfico solamente se lleva a cabo entre BS y SSs, mientras que en el modo Mesh el tráfico puede se enrutado a través de las SSs y puede estar presente directamente entre SSs. Dependiendo del algoritmo de transmisión utilizado, si se utiliza en base a la igualdad de se utiliza scheduling distribuido, o si se basa en la superioridad de la Mesh BS, que resulta en un scheduling centralizado, o en una combinación de ambos. La mayor ventaja de las redes Mesh es que el alcance de la BS es mucho mayor, depende del número de saltos, hasta el nodo más distante.

En la figura 3.4 se muestra un ejemplo de arquitectura Mesh. A continuación se presenta más detalladamente el modo de operación de las redes Mesh bajo el estándar mencionado, ya

que es el tipo de operación que se analizó e implementó en el presente trabajo.

3.6. Arquitectura Mesh en redes WiMAX

En el presente trabajo se elaboró la inicialización de estas redes, es decir, cada uno de los nodos que puedan ser alcanzados por la cobertura de otros nodos que a su vez estén conectados con otros más de tal forma que con que un solo nodo de esta red ad-hoc pueda conectarse con una BS , planteado de otra forma un nodo que tenga acceso a internet o alguna otra red, pueda dar acceso a cada uno de los nodos ad-hoc.

3.7. Operación de topología Mesh

Primero se definirán los términos utilizados en la especificación del modo de operación, para una mejor compresión del funcionamiento:

Término	**Concepto**
Mesh BS	Estación Base Mesh (MBS) que tiene acceso a un servicio backhaul fuera de la red Mesh
Mesh SS	Sistemas que se encuentran aparte de la BS dentro de la red Mesh se denominan, estación suscriptoras de tipo Mesh (MSS)
nodo	Cada uno de los sistemas de la red Mesh
uplink	Información en dirección a la MBS
downlink	Información en dirección contraria a la MBS
nodo vecino	cualquier sistema que este a un salto del nodo
vecindad	el conjunto de nodos que son vecinos.
vecindad extendida	vecinos de la vecindad de un nodo

Tabla 3.2: Términos básicos de la operación de redes Mesh

En un sistema con topología Mesh los nodos y la MBS tienen que coordinarse para transmitir la información una vez inicializados.

El protocolo IEEE 802.16 utiliza TDMA (accesso múltiple por división de tiempo) para acceder al canal tanto de la MBS como MSSs, donde el canal de radio esta divido en frames. Cada frame esta divido en slots de tiempo que puede ser asignado a MBS y nodos MSS. En la fig 3.5 muestra la estructura de un frame en modo mesh. Un frame consiste en un subframe de control y un subframe de datos. Cada frame esta divido dentro de 256 minislots de para transmisión de datos del usuario y mensajes de control. En el subframe de control esta formado de segmentos de tiempo conocidas como oportunidades de transmisión , cada oportunidad de transmisión puede constar de uno o más minislots, y son utilizadas para enviar mensajes de configuración de la red o del planeador de transmisión (*scheduler*). Un árbol de ruteo es establecido para la comunicación entre MBS y cada uno de los nodos (MSS's).

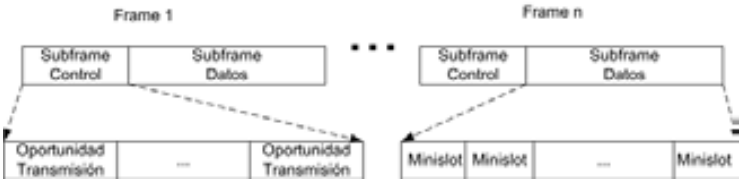


Figura 3.5: Estructura Frame IEEE 802.16 Mesh

3.8. Capa MAC

Cada nodo tiene un dirección MAC asociada de 48 bits , como se definió en el estándar IEEE 802-2001. Esta dirección es utilizada durante el proceso de entrada del nodo a la red y como parte del proceso de autorización para cada nodo candidato y validar la identificación de cada uno de los nodos.

Cada nodo también debe contar con un número que identifique cada nodo (nodeID)que es un número de 16 bits. NodeID es la forma de identificar los nodos durante la operación normal.

3.8.1. Mensajes de Control MAC

El estándar define un conjunto de mensajes de control. Estos mensajes deben ser instanciados dentro del *Payload* del MAC PDU como lo vimos en la figura 3.2 que a su vez contiene un subframe en modo mesh (figura 3.5). Todos los mensajes de control inician con un campo *Management Message Type* y pueden contener más parámetros, cabe mencionar que sólo se revisarán los tipos de mensajes que se utilizan en la arquitectura Mesh.

Los códigos de los tipos de mensajes de control se muestran en la tabla 3.3 :

Tipo	Nombre	Descripción	Conexión
39	MSH-NCFG	Mesh Network Configuration	Broadcast
40	MSH-NENT	Mesh Network Entry	Basic
41	MSH-DSCH	Mesh Distributed Schedule	Broadcast
42	MSH-CSCH	Mesh Centralized Schedule	Broadcast
43	MSH-CSCF	Mesh Centralized Schedule Configuration	Broadcast

Tabla 3.3: Mensajes de control MAC para Mesh

3.8.2. Mensaje de control MSH-NCFG (Configuración de Red)

Un mensaje MSH-NCFG es enviado vía *downlink*, el primer tipo de mensaje *MSH-NCFG:Network Descriptor* es un mensaje broadcast que invita a todos los nodos que estén en la cobertura de MBS en un inicio. Posteriormente este mensaje es re-enviado a los nodos consecuentes del nodo que ha recibido el mensaje, además de generarse periódicamente el mensaje de éste tipo actualizado la información de la red y buscando nuevos nodos que quieran integrarse a ella.

En la tabla 3.4 se muestra el formato para el mensaje MSH-NCFG

Sintaxis	**Tamaño**	**Notas**
Management Message Type = 39	8 bits	
NumNbrEntries	5 bits	Número de vecinos reportados
NumBSEntries	2 bits	Número de Mesh BS reportados
Embedded Packet Flag	1 bit	0 = No, 1 = presente
Xmt Power	4 bits	Cada 2dB, comenzando en 8 dB
Xmt Antenna	3 bits	Antena lógica utilizada para el envío de este mensaje
NetEntry MAC Address Flag	1 bit	0= No, 1= presente
Network base channel	4 bits	El canal base que esta siendo utilizado para este nodo
reserved	4 bits	Debe ser 0
NetConfig Count	4 bits	Contador de mensaje MSH-NCFG enviados por este nodo
Timestamp		
Frame Number	12 bits	
Network Control Slot Number in frame	4 bits	
Synchronization Hop Count	8 bits	
NetConfig schedule info		
Next Xmt Mx	3 bits	
Xmt Holdoff Exponent	4 bits	
Net Entry MAC Address	48 bits	
for (i=0; i < NumBSEntries; ++i) {		
BS Node ID	16 bits	
Number of hops	3 bits	
Xmt energy/bit	5 bits	
}		
for(i=0; i < NumNbrEntries; ++i) {		
Nbr Node ID	16 bits	
MSH-Nbr_Physical_IE	16 bits	
if (Logical Link Info Present Flag)		
MSH-Nbr_Logical_IE	16 bits	
}		
if (Embedded Packet Flag)		
MSH-NCFG_embedded_data	16 bits	Ver tabla 3.5

Tabla 3.4: Formato mensaje MSH-NCFG

A continuación se describe los campos que contiene el mensaje MSH-NCFG:

NumNbrEntries Representa el número de vecinos dados de alta actualmente en la red.

NumBSEntries Número de estaciones base que se encuentran activas.

Xmt Power En pasos de 2 dBm, empezando de 8 dbm, por ejemplo se puede poner 1111 que indica que son 38 dBm

Xmt Antenna ocho posibles direcciones a las que puede ir direccionada la antena

Network base cannel Se usa para enviar información en broadcast.

Netconfig count Es un contador de los mensajes MSH-NCFG que se han enviado.

Frame Number Se incrementará en uno cada vez que contabilice un frame y tiene un valor máximo de 2^{12}.

Synchronization hop count Es un número usado para la sincronización entre todos los nodos que existen en la red.

Xmt Holdoff Exponent Es el número de oportunidades de transmisión para enviar un mensaje MSH-NCFG después del *Next Xmt Time*.
El valor de esta variable se utiliza para calcular el siguiente valor:

$$XmtHoldoffTime = 2^{(XmtHoldoffExponent+4)}$$

Este parámetro se usará para la realización del algoritmo para seleccionar quién va a transmitir y poder evitar colisiones.

Next Xmt Mx Es el rango de tiempo en que un nodo puede enviar mensajes MSH-NCFG, que esta definido por:

$$2^{XmtHoldoffExponentNext} * XmtMx \leq NextXmtTime 2^{XmtHoldoffExponent} * (NextXmtMx+1)$$

NetEntry MAC Address Nos indica la presencia de un nuevo nodo en la red.

BS node ID Es el número de nodo que distingue a la estación base.

Number of hops Es el número de saltos, es decir, el número de estaciones suscriptoras involucradas para llegar de la estación base a la estación suscriptora de interés

Xmt energy/bit factor Nos indica la energía / bit requerida para enviar la información

Nbr node ID El número identificador del nodo de la estación suscriptora.

En la tabla 3.5 se muestra el siguiente nivel del mensaje MSH-NCFG en donde se establece el tipo que se esta enviando y posteriormente se muestran los diferentes valores que se pueden adquirir en campo *Type* dependiendo en que tiempo del proceso de entrada a la red se encuentre.

Sintaxis	Tamaño	Notas
MSH-NCFG_embedded_data		
Extended embedded_data	1 bit	Indica si el IE embebido esta seguido por otro 0 = No, 1 = Sí
reserved	3 bits	Deber ser 0
Type	4 bits	
Length	8 bits	Tamaño en bytes de embedded_IE(), excluyendo esta cabecera
Embedded_data_IE	variable	Depende de *Type, ver tabla 3.6*

Tabla 3.5: Formato MSH-NCFG_embedded_data

Type

Los siguientes tipos están definidos:

0x0 *reservado*

0x1 *Network Descriptor*

0x2 *Network Entry Open*

0x3 *Network Entry Reject*

0x4 *Network Entry Ack (Embedded_data_IE() = NULL)*

0x5 *Neighbor Link Establishment Protocol*

3.8.2.1. MSH-NCFG:Network Descriptor

Este mensaje es enviado por medio de un mensaje broadcast, es transmitido periódicamente y es el que inicia la comunicación con los nodos que desean ingresar a la red, enviando la información de la configuración de la red. Su formato se muestra en la tabla 3.6.

3.8.2.2. MSH-NCFG:Network Entry Open

Este mensaje se utiliza para responder al mensaje *MSH-NENT:Request*, la tabla 3.7 contiene el formato de este paquete.

3.8.2.3. MSH-NCFG:Network Reject

Este mensaje es enviado cuando se rechazan los mensajes *MSH-NENT:Request*, este mensaje debe contener los parámetros listados en la tabla 3.8.

Posibles códigos de rechazo:

0x0 Autenticación de operador con valor no válido.

0x1 Exceso en retardo de propagación.

0x2 Selecciona un nuevo nodo proveedor de servicio.

Sintaxis	Tamaño	Notas
MSH-NCFG_embedded_data_IE		
Frame Length Code	4 bits	4 LSB de Duración del frame
MSH-CTRL-LEN	4 bits	Tamaño de subframe de control
MSH-DSCH-NUM	4 bits	Número de oportunidades en el subframe DSCH
MSH-CSCH-DATA-FRACTION	4 bits	
Scheduling Frames	4 bits	Cuantos frames contienen subframes de control para los planificadores entre dos frames que contienen subframes de control de red en múltiplos de 4
Num_Burst_Profiles	4 bits	Número de tramas de definición de perfiles.
Operator ID	16 bits	
XmtEnergyUnitsExponent	4 bits	
Channels	4 bits	Número de canales lógicos. Un valor de 0 indica que la información del canal no esta siendo portada en este mensaje
MinCSForwardingDelay	7 bits	Número de símbolos OFDM de retardo entre re-enviar la información y recibirla.
ExtendedNeighborhoodType	1 bit	0 = vecinos a 2 saltos, 1 = vecinos a 3 saltos
if (Channels)		
MSH-NCFG_Channel_IE	variable	
for (i=0; i < NumBSEntries; ++i) {		
FEC Code Type	8 bits	
Mandatory Exit Threshold	8 bits	
Mandatory Entry Threshold	8 bits	
}		

Tabla 3.6: Formato MSH-NCFG_embedded_data_IE para mensajes *MSH-NCFG:Network Descriptor*

3.8.2.4. MSH-NCFG:Network Entry ACK

En este caso el campo *MSH-NCFG_embedded_data_IE()* es igual a NIL, este mensaje termina el proceso de entrada a la res, además de ser la respuesta a un mensaje *MSH-NENT:Close*

3.8.2.5. MSH-NCFG: Neighbor Link Establishment Protocol

Después de que un nodo ha entrado a la red, puede establecer enlaces con otros nodos que están comunicados con su nodo proveedor de servicio, los parámetros de este paquete se encuentran en la tabla 3.9.

Sintaxis	Tamaño	Notas
MSH-NCFG_embedded_data_IE		
Minislot Start	8 bits	En que slot comienza el planificador para la capa superior de entrada de red.
Minislot Range	8 bits	Rango del planificador para la capa superior de entrada de red.
Frame number	12 bits	Número de frame en el que el planificador comienza a ser válido.
Channel	4 bits	Canal lógico para transmitir un nuevo nodo de acuerdo al rango del planificador.
Schedule validity	12 bits	Válida el planificador en frames.
Channel	4 bits	Canal lógico para recibir para un nodo nuevo.
Estimated Propagation Delay	4 bits	Medida en μs retardo de propagación estimado.
reserved	4 bits	Debe ser 0.

Tabla 3.7: Formato MSH-NCFG_embedded_data_IE para mensajes tipo *MSH-NCFG: Network Entry Open*

Sintaxis	Tamaño	Notas
MSH-NCFG_embedded_data_IE		
Reject Code	8 bits	Código de razón de rechazo.
Reject Reason	160 bits	Cadena ASCII, comentarios.

Tabla 3.8: Formato MSH-NCFG_embedded_data_IE para mensajes tipo *MSH-NCFG:Reject*

Sintaxis	Tamaño	Notas
MSH-NCFG_embedded_data_IE		
Action Code	2 bits	0x0 Reto 0x1 Respuesta reto 0x2 Aceptado 0x3 Rechazado
reserved	6 bits	Puede ser 0.
if (Action Code == 0x0 o 0x1)		
Nbr Authentication value	32 bits	
if (Action Code == 0x1 o 0x2)		
Link ID	8 bits	Transmite ID de los enlaces del nodo por este enlace

Tabla 3.9: Formato MSH-NCFG_embedded_data_IE() para mensajes tipo *MSH-NCFG:Neighbor Link Establishment IE*

3.8.3. Network Entry

Este mensaje es enviado vía uplink para la inicialización y la sincronización de un nuevo nodo dentro de la red Mesh. En la tabla 3.10 se muestra el formato para el mensaje MSH-NENT

A continuación se describen los elementos que forman el mensaje MSH-NENT:

Sponsor Node ID Es el número de nodo inmediato superior al cual esta accediendo la estación suscriptora para poder incorporarse en la red. Es decir, es el proveedor de servicio para el nodo desde el cual se hacer referencia.

Xmt Power Se asigna en pasos de 2 dBm y empieza en el nivel de 8 dBm.

Xmt Antenna Soporta ocho direcciones.

Sintaxis	**Tamaño**	**Notas**
Management Message Type = 40	8 bits	
Type	3 bits	0x0*reserved* 0x1 NetEntryAck 0x2 NetEntryRequest 0x3 NetEntryClose
XmtCounter for this Type	3 bits	
Sponsor Node ID	16 bits	
Xmt Power	4 bits	
Xmt Antenna	3 bits	
if (Type == 0x2)		
MSH-NENT_Request_IE()	*variable*	Ver tabla 3.11

Tabla 3.10: Formato mensaje MSH-NENT

El formato del mensaje *MSH-NENT:Request* se muestra en la tabla 3.11. Los campos se describen a continuación:

MAC Address Es la dirección MAC del nodo que quiere incorporarse a la red.

OpConfInfo Es información de configuración proporcionada por el operador de la red.

Operator Authentication Value Utiliza el número serial del nodo para generar una llave secreta para registrarse.

3.8.4. Modo de sincronización Mesh

Los paquetes MSH-NCFG y MSH-NENT generan una comunicación básica entre nodos, estos dos tipos de mensajes ayudan a un nodo a sincronizarse e ingresar a la red. La inicialización de la red se realiza en la primera parte del multiframe, y la sincronización de cada uno de los mensajes se describe en esta sección.

Sintaxis	Tamaño	Notas
MSH-NENT_Request_IE		
MAC Address	48 bits	Dirección MAC para la petición de ingreso a la red
OpConfInfo	64 bits	Información de Operador de Configuración
Operator Authentication Value	32 bits	HMAC{ MAC, Número Serial de Nodo, AK}
Node Serial Number	32 bits	

Tabla 3.11: MSH-NENT Request IE

Los nodos activos dentro del periodo de aviso del mensaje *MSH-NCFG:Network Descriptor*, el cual contiene la información de configuración de la red, entre ellas el número de identificador de la MBS y el canal base que se utiliza en ese momento. Un nuevo nodo que planea unirse a la red busca redes activas y escucha, en espera de un mensaje MSH-NCFG. El nuevo nodo establece una sincronización y comienza el proceso de entrada a la red basándose en la información dada en el paquete MSH-NCFG recibido. Un mensaje MSH-NENT es enviado por el nodo candidato con información *NetEntryRequest* para unirse a la red. En la figura 3.6 se muestra la cronología de ingreso a la red.

En el diagrama 3.6 se muestran los pasos que se tienen que llevar a cabo para que ingrese el nodo y se sincronice con la red. Esta secuencia de mensajes son los que debe seguir exactamente un nodo. Cada uno de los tiempos en que un mensaje en concreto es enviado se define en la siguiente se explica posteriormente.

Tiempo de transmisión para el siguiente mensaje MSH-NCFG

El estándar propone un algoritmo de selección utilizando la función *Mesh Election*[1], se transcribe a continuación

Durante el tiempo actual de transmisión de un nodo *Xmt Time* (es decir el tiempo de slot durante el cual el nodo transmite su paquete MSH-NCFG), cada nodo utiliza el algoritmo de elección para determinar su siguiente tiempo de transmisión ***Next Xmt Time*** que se muestra en la tabla 3.12

3.8.5. Esquemas de Planeación de tiempo

La capa MAC cuenta con dos mecanismos de planeación, que se caracterizan principalmente por la manera de distribuir los datos o información, una vez que los nodos han ingresado a la red y están preparados para transmitir datos. Estos tiempos de transmisión de datos en el modo mesh se puede coordinar de acuerdo a planeación centralizada (*centralized scheduling*, MSH-CSCH) o planeación distribuida (*distributed scheduling*, MSH-DSCH).

[1]La implementación de esta función se encuentra en el Apéndice D

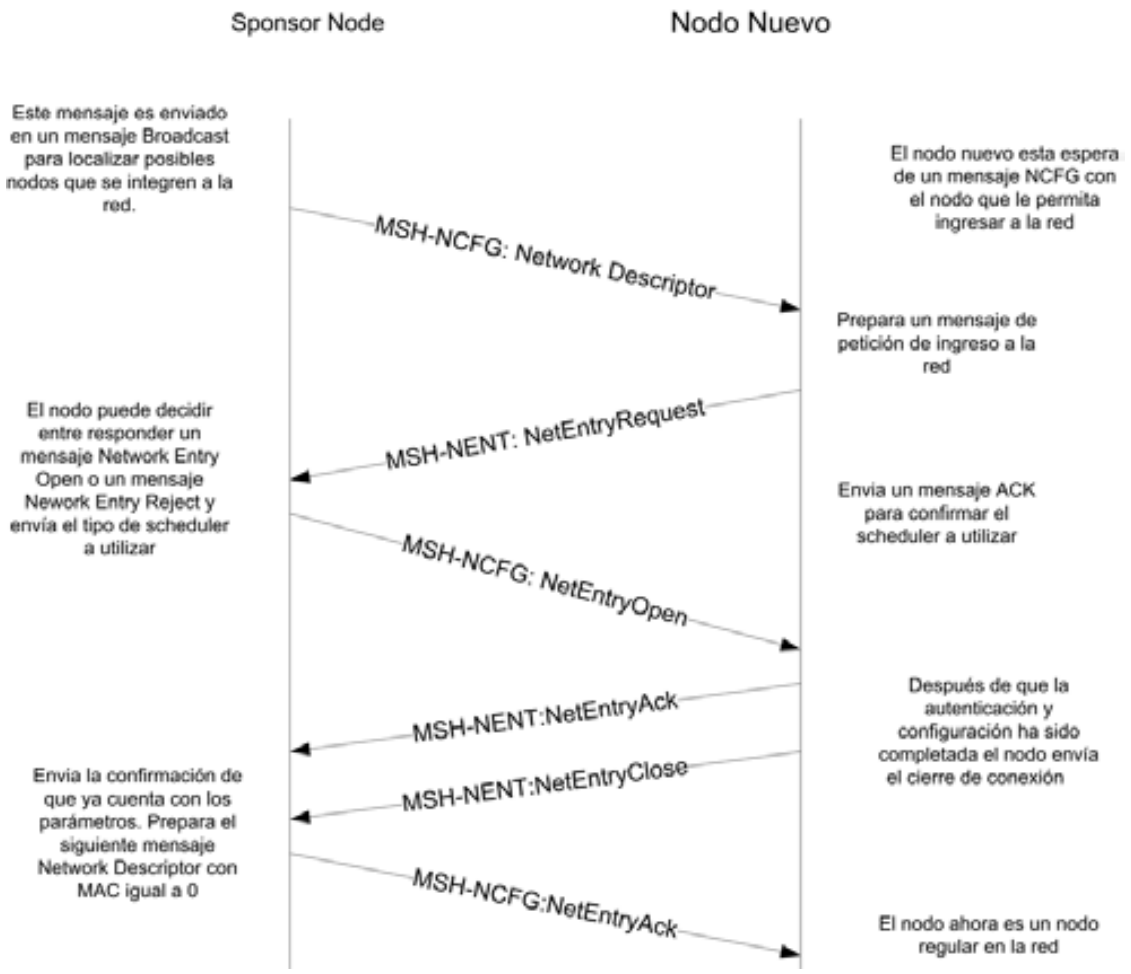


Figura 3.6: Cronología de los paquetes MSH-NCFG y MSH-NENT

Planeación centralizada

MSH-CSCH *(Mesh Centralized Scheduler)* se basa en cómo su nombre lo indica en una organización centralizada que tiene como punto de referencia la BS. Todos los tiempos son administrados desde ésta ya que esta es la que concede slots de transmisión para cada uno de los nodos, éste forma de planeación es más eficiente en largos periodos de tiempo. Cada porción de tiempo es llamado periodo de planeación, el cual generalmente consta de dos frames de longitud.

Hay dos posibles escenarios en cada periodo de planeación. En el primer escenario las MSS's envían peticiones de ancho de banda utilizando un mensaje *MSH-CSCH:Request* que es enviado por medio de sus nodos de expansión o nodos proveedores de servicio, el cuál es encaminado a la MBS por medio del árbol generado para la planeación. Cada nodo envía su propia petición, y la de sus descendientes. Las MSS's transmiten los mensajes *MSH-CSH:Request* en el

Ordena su tabla de vecinos por ***Next Xmt Time.***
Para cada entrada de la tabla de vecinos, agrega el ***Next Xmt Time*** a el ***Xmt Holdoff Time*** del nodo correspondiente junto a su ***Earliest Subsequent Xmt Time***.
Iguala ***TempXmtTime*** a ***Xmt Holdoff Time*** más el tiempo actual: ***Xmt Time***.
success igual a *false*.

> ***While success*** sea igual a *false* ***do***:
> Determina cuales son los nodos que compiten, cual es el conjunto de todos los nodos en la lista de vecinos con un intervalo ***Next Xmt Time*** elegible que incluya ***TempXmtTime*** o con un ***Earliest Subsequent Xmt Time*** igual o menos que ***TempXmtTime***.
> Envía a *Mesh Election* el conjunto de nodos elegibles y el nodo local utilizando ***TempXmtTime*** y la lista de IDs de los nodos que compiten elegidos como entrada.
> *MeshElection (TempXmtTime,MyNodeID,CompetingNodeIDsList [ ])*
> If (Este nodo no ganó)
> Igualar ***TempXmtTime*** a la siguiente tiempo MSH-NCFG.
> Else: Iguala *success* a *true*.
> Iguala ***Next Xmt Time*** a *TempXmtTime*.

Tabla 3.12: Algoritmo de selección utilizando *Mesh Election* para mensajes MSH-NCFG

orden en que se recorren los nodos de expansión del árbol, es decir siempre transmiten después las peticiones de los sus nodos hijos. Así es como la MBS obtiene las peticiones de ancho de banda de todos los nodos de la red.

El segundo escenario la MBS calcula y transmite un mensaje *MSH-CSCH:Grant* por medio de difusión, el cual se propaga por todo el árbol que forma la red. Como todo el control y paquetes de datos necesariamente pasan por MBS el procedimiento es simple, pero el retardo en el envío de datos es grande.

Planeación distribuida

En un sistema distribuido los nodos son organizados como en una red ad-hoc y todos los nodos son puntos que pueden actuar como routers para transmitir a sus vecinos. Cada nodo compite por el acceso al canal utilizando un algoritmo de selección seudoaleatoria, basado en la información de la red de los nodos que tiene a distancia de dos saltos, los frames de datos son otorgados a través de procedimiento de tres vías petición-concesión-confirmación. Esto hace más flexible y eficiente el uso del canal y sobre tiene como principal característica la flexibilidad de la conexión. Utiliza los mensajes de control de tipo MSH-DSCH (*Mesh Distributed Scheduler*).

Capítulo 4

Diseño e implementación del modelo de simulación para redes Mesh

4.1. Introducción

En este capítulo se describe el diseño para el modelo que se realizó para simular el comportamiento dinámico de la inicialización de una red Mesh mediante el software OPNET.

Se analizó a detalle el estándar para generar el modelo de una red Mesh que inicializará cada uno de sus nodos. Haciendo hincapié en la sincronización de tiempos entre mensajes de inicialización, para llevar a cabo un mejor rendimiento de oportunidades de transmisión de cada uno de los mensajes.

4.2. Análisis y diseño

Se utilizaron parámetros analizados en estudios teóricos, del comportamiento dinámico para el uso de los planificadores. Basados en los resultados de artículos tales como [6] y [8] en donde presuponen la inicialización de la red y se enfocan a la distribución equitativa del ancho de banda y en la planeación centralizada, respectivamente. Estos parámetros se muestran en la tabla 4.1utilizaron como previo a la implementación de los planeadores centralizados o distribuidos, y al envío de datos.

La herramienta de modelado OPNET provee de un conjunto de editores para especificar diferentes niveles del análisis y diseño.

El modelo que se implementó se divide en tres fases, que se implementaron dentro de la herramienta OPNET:

Modelo de Red Se construye la topología de la red a alto nivel, en este caso consta de una estación base y 100 nodos.

Parámetros	Valores
Duración de Frame T_F	10 ms
No. de símbolos OFDM / frame	1024
No. de símbolos OFDM / slot	4
Tiempo por oportunidad de transmisión/ slot	68.359 μs
No. de slots / frame	256
No. de bytes / Símbolo OFDM	72
Ancho de Banda	25MHz
Potencia de nodos	0.3 W
Tasa de transferencia de datos	59 Mbps

Tabla 4.1: Parámetros de simulación

Modelo de Nodos Se describe el comportamiento de los diferentes módulos que desempeñan papeles específicos, como pueden ser las diferentes capas por la que pasan los datos en la pila de protocolos.

Modelo de Procesos Se define el comportamiento de cada nodo, basada en una estructura de máquina de estados, que implementadas en lenguaje Proto-C, emulan comportamiento y funcionalidades de sistemas reales, se dividió en 2 partes, estación base y estación suscriptora.

En la tabla 4.2 se muestra un ejemplo de los distintos modelos en OPNET.

Modelo de red	Modelo de procesos
	
	
Modelo de nodos	Código en Proto-C

Tabla 4.2: Fases de modelado en OPNET

El diseño para el entorno de simulación se realizó en diferentes partes. Una de éstas fue la creación de los paquetes definidos en el estándar y descritos en la sección 3.8.1 (MSH-NCFG y MSH-NENT). A continuación se muestran la jerarquía de paquetes que se utilizaron en el modelo.

En la figura 4.1 se muestra las estructuras para los diferentes tipos de mensajes que son posibles para los paquetes MSH-NCFG. En la parte superior derecha de cada trama se muestra el nombre con que el paquete fue creado, dentro del modelo. Los parámetros que difieren en cada tipo están definidos en el paquete *msh_ncfg_emb_data_fmt*, el parámetro ***Type*** indica que tipo de paquete será anidado dentro del campo *Embedded Data IE*. Los diferentes paquetes son anidados dependiendo de la fase de entrada a la red en que se encuentra el nodo, la relación del tipo de paquete y el paquete que es anidado se muestran en las secciones dividas por líneas punteadas.

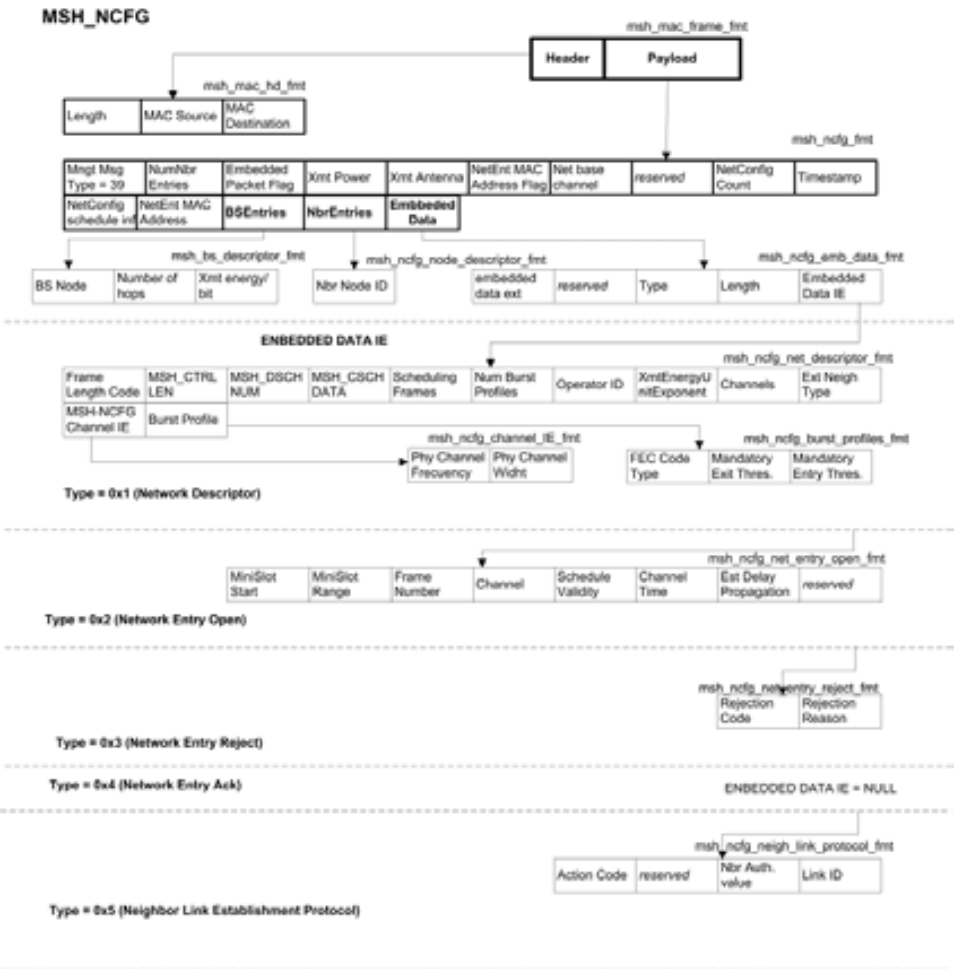


Figura 4.1: Estructura de lo paquetes para diferentes tipos de mensajes MSH-NCFG

En la figura 4.2 se visualiza la estructura general para los mensajes MSH-NENT que únicamente cambia su parámetro *Type* para indicar en que fase de la inicialización se encuentra. En caso de que el tipo sea *MSH-NENT:Request*, es decir, cuando *Type = 2* se anida el paquete *msh_nent_req_fmt*. Este paquete contiene los datos para cuando un nuevo nodo recién va a ingresar a la red.

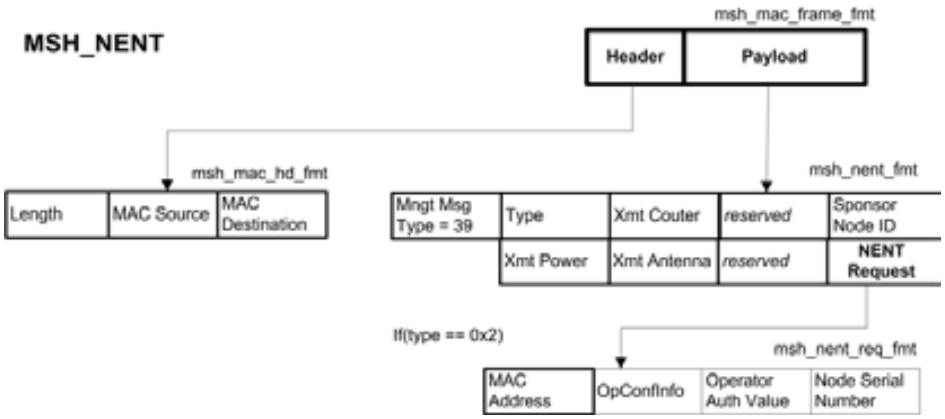


Figura 4.2: Composición de mensajes NENT

La figura 4.3 muestra un ejemplo de la creación de los paquetes que se generaron para el modelo en OPNET a partir de las estructuras anteriores.

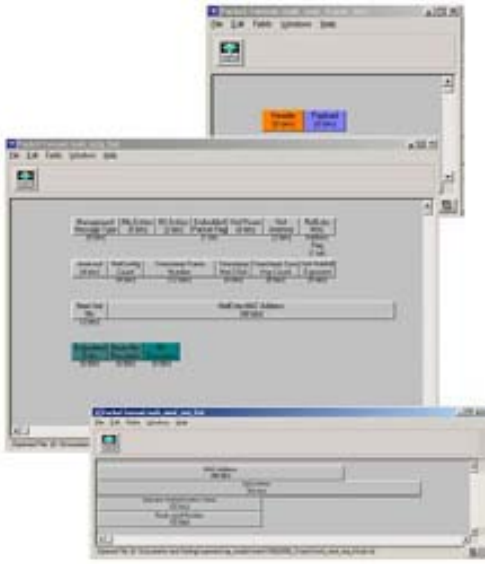


Figura 4.3: Paquetes en OPNET

4.3. Capa física

Para el sustento de la capa física OPNET maneja un procedimiento conocido como *pipeline*, que es una serie de pasos que se realiza en el *radio link* para la transmisión/recepción de un paquete. En la figura 4.4 se muestra este proceso.

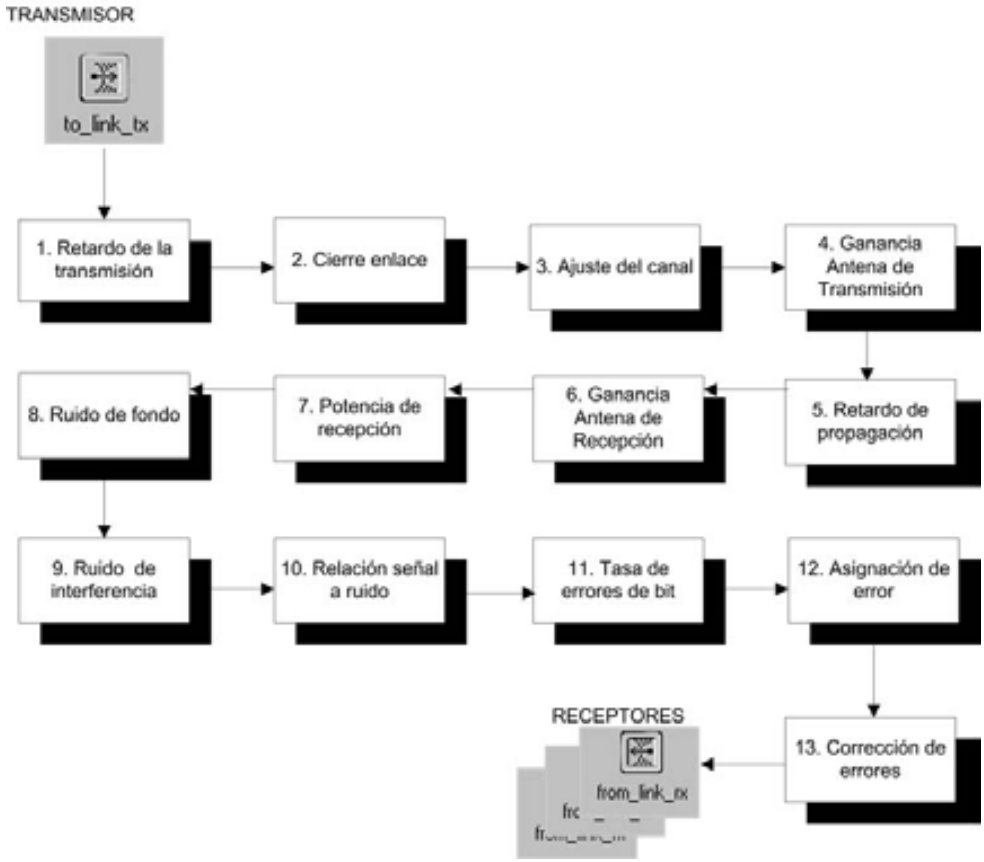


Figura 4.4: Pipeline de radio link

Los archivos que fueron utilizados en el modelo y que corresponden uno a uno con cada fase del diagrama, cada uno de estos programas fueron realizados en Proto-C. A continuación se describen brevemente.

1. **mesh_txdel** Retardo de transmisión. Cuando un nueva transmisión se lleva a cabo es el primer paso a ser ejecutado. Este escenario es invocado para calcular el tamaño del tiempo que es requerido para realizar la transmisión completa del paquete.Este resultado es la diferencia entre el inicio del primer bit y el final del último bit del paquete.

2. **mesh_closure** Cierre del canal. El propósito de esta fase es determinar si un canal receptor puede ser afectado por una transmisión o no. La capacidad de transmisión para el alcance del canal receptor es remitido para el cierre entre el canal de transmisión y el canal receptor.

3. **mesh_chanmatch** Ajuste del canal. El propósito de esta fase es clasificar la transmisión con respecto a el canal receptor. Una de estas tres categorías es asignada como atributo al paquete:

    1. Válido. Paquetes compatibles con el canal receptor
    2. Ruido. Paquetes cuyos datos no pueden ser recibidos, pero tienen algún tipo de impacto en el canal receptor.
    3. Ignorado. Si el paquete no tiene ninguna influencia en el canal receptor se utiliza esta clasificación.

4. **mesh_bs_tagain** Ganancia de transmisión.En esta fase se calcula la ganancia que proveniente de la antena asociada al transmisor, basada en la dirección del vector que va del transmisor al receptor.

5. **mesh_propdel** Retardo de propagación. Esta fase calcula la cantidad de tiempo requerido para que la señal del paquete viaje desde el radio transmisor al radio receptor. Este resultado generalmente depende de la distancia entre origen y destino.

6. **mesh_bs_ragain** Potencia de recepción. Se compara la ganancia asociada a la antena del receptor, basada en la dirección del vector desde el receptor al transmisor.

7. **mesh_bs_power** Se calcula la potencia de recepción de la señal del paquete entrante (en watts).

8. **mesh_bkgnoise** Ruido de fondo. Representa el efecto de todos los orígenes de ruido, excepto de otra transmisión que se realice paralelamente. Típicamente el ruido de fondo incluye ruido termal o galáctico emisiones de aparatos electrónicos y otras señales de radio.

8. **mesh_inoise** Ruido de interferencia. Cuenta la interacción entre transmisiones que llegan paralelamente al mismo canal receptor.

9. **mesh_snr** Relación señal a ruido. El propósito del SNR (*Signal-to-Noise Ratio*) Es calcular la potencia promedio para los paquetes entrantes. Este cálculo se basa generalmente en valores obtenidos durante las fases anteriores, incluyendo la potencia de recepción, ruido de fondo y ruido de interferencia. El SNR del paquete es una medida importante de desempeño que determina la capacidad del receptor para obtener correctamente el contenido del paquete.
    La ecuación para calcular el SNR es la siguiente:
    $SNR[dB] = Prx[dB] - N[dB]$

    En donde:
    $Prx[dB] = EIRP + Grx - PL[dB]$

    $EIRP = Ptx + Gtx - Lc$

Lc representa las pérdidas de cable y conectores.
Gtx Ganancia de transmisión, Prx Potencia de recepción.

$N[dB] = 10 * log_10(T * BW * Ko) + Nf + 30$ es el ruido térmico, calculado en la fase 7.

Los demás parámetros fueron obtenidos de la tabla 4.3 para terreno A y definidos en un archivo de parámetros de simulación.

$BW = 25MHz$
$Ptx = 0{,}3W$
Esta potencia se utilizó para que la cobertura de cada nodo no sobrepasará 1 km como se muestra en la figura 4.5.

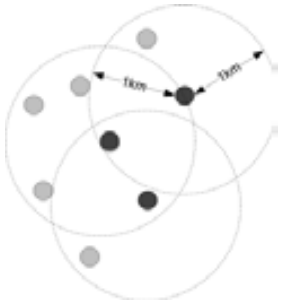


Figura 4.5: Cobertura entre los nodos

10. **mesh_ber** Tasa de error de bit. El BER (*Bit Error Rate*) deriva la probabilidad de errores de bit durante el intervalo anterior de la constante SNR. Este valor esta en función del la modulación que se utiliza para transmitir la señal.

11. **mesh_error** Asignación de errores. Esta fase es para estimar el número de errores de bit en segmento de paquete donde la probabilidad de error de bit ha sido calculada y es constante.

12. **mesh_ecc** Corrección de errores. En esta fase se determina si el paquete entrante puede o nos er aceptado. Esto depende generalmente si el paquete ha colisionado, el resultado calculado en la asignación de errores, y en la capacidad del receptor para corregir los errores que afectan al paquete.

Dentro de este modelo de propagación se consideran tres categorías de terreno que se explican en la tabla 4.3 sus parámetros afectan directamente el comportamiento del modelo de propagación.

Estas categorías se dividen en terrenos para áreas urbanas, semi-urbanas y rurales, la principal diferencia los tipos de terreno se basa en la densidad de obstáculos.

Para el Modelo de Propagación implementado se consideró terreno con Categoría A, basándose en la idea de que una red de este tipo sería más útil dentro de un área rural.

Parámetros	Categoría de Terreno		
	A Accidentado/Densidad de árboles moderada a alta	B Accidentado/Densidad de árboles ligera o Plano/Densidad de árboles alta	C Plano/Densidad de árboles ligera
a	4.6	4.0	3.6
b	.0075	.0065	.0050
c	12.6	17.1	20.0
σ_γ	0.57	0.75	0.59
μ_σ	0.6	9.6	8.2
σ_σ	2.3	3.0	1.6

Tabla 4.3: Parámetros utilizados por el tipo de terreno

4.4. Capa MAC

4.4.1. Diagramas de flujo

Los diagramas de flujo muestran la secuencia de intercambio de mensajes en las figuras 4.6, 4.7 y 4.8, obtenidos del estándar y traducidos para este trabajo[2]. Estos diagramas muestran como fue implementado el comportamiento de la red mesh, para el proceso que siguieron los nodos que iban ingresando a la red. Estos nodos pueden utilizar a un nodo proveedor del servicio que sea el conector entre el y la MBS, e incluso estar conectado a una serie de nodos que se conectan entre sí hasta la MBS. Por lo que esta funcionalidad es ser recursiva hasta llegar a la MBS.

Los diagramas se describen detalladamente en el Modelo de Procesos ya que es la funcionalidad que se implementó.

4.4.2. Capa MAC para estación base

En la figura 4.9 se muestra el modelo referencia a la estación base y posteriormente los módulos que componen el funcionamiento de la estación suscriptora. Primero tenemos el Modelo de Nodos de Estación Base.

El presente trabajo se dedicó básicamente a proveer la función dinámica de la capa MAC, haciendo hincapié en el ingreso de los nodos a la red, como se ha mencionado a lo largo del trabajo.

La inicialización consta de varias etapas:

La primera es un mensaje tipo broadcast que es enviado desde la estación base a los nodos que están dentro de su cobertura.

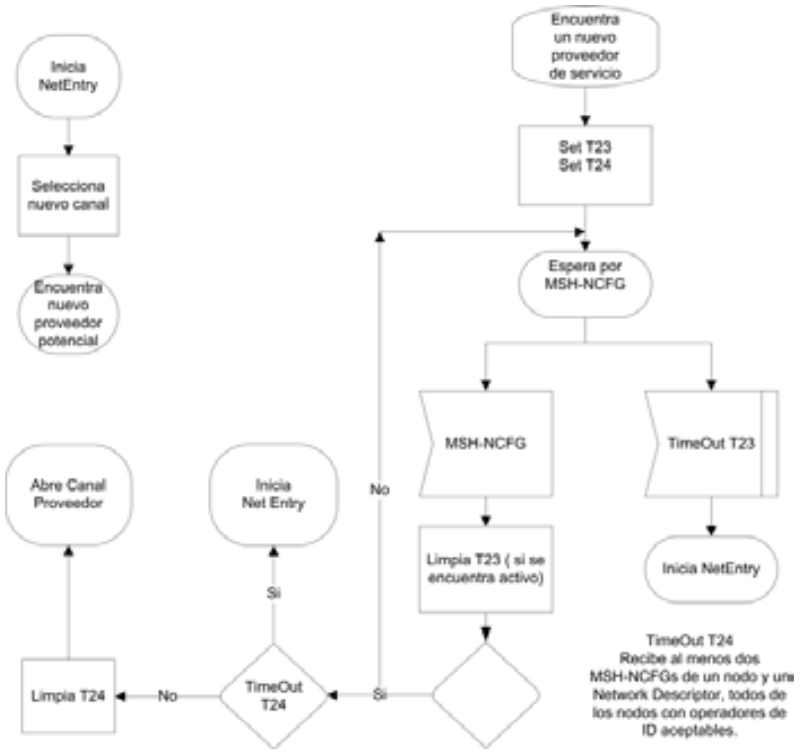


Figura 4.6: Sincronización y entrada de un nuevo nodo en red Mesh- 1era parte

La segunda es una vez que es recibido el mensaje por todos los nodos que están dentro de la cobertura, contestan para realizar la petición a la estación base o a su nodo padre, este proceso se realiza mediante 2 intercambios de mensajes por cada una de las partes, subsecuentes a la petición.

La tercera es la finalización del ingreso a la red, una vez que el nodo ya forma parte de la red, se prepara para re enviar los broadcast a los nodos que a su vez estén dentro de su rango de alcance.

La siguiente etapa en cuando los nodos que han recibido el mensaje re-enviado por los nodos que ya están dados de alta previamente.

mesh_bs_mac

El módulo mesh_bs_mac determina el comportamiento de la capa MAC de la estación base, consta de una cola que procesa todos los mensajes MSH-NENT que entran. Se encarga de generar todos los mensajes MSH-NCFG de bajada y enviar la información de la red en paquetes *MSH-NCFG:Network Descriptor* que son enviados en mensajes broadcast periódicamente en búsqueda de nuevos nodos dados de alta.

La máquina de estados se muestra en la figura 4.10, donde se muestra el flujo que lleva el

Figura 4.7: Sincronización y entrada de un nuevo nodo en red Mesh- 2da parte

programa y que es determinada por el valor de las macros que se programaron de acuerdo a la acción requerida.

A continuación se describe brevemente lo que se realiza en cada uno de los pasos de este diagrama:

Init

1. Determina la ubicación física de la estación BS, definidas por las coordenadas x e y.

2. Verifica que se cumpla la interrupción inicial al inicio de la simulación.

Figura 4.8: Sincronización y entrada de un nuevo nodo en red Mesh-Comportamiento de nodo proveedor de Servicio

3. Obtiene los parámetros de la red.
4. Inicializa la lista que contendrá a todos los nodos pertenecientes a la red.

Idle

1. Maneja diferentes tipos de interrupción, que pueden ocurrir, cuando: OPC_INTRPT_SELF: Se ha programado un mensaje a ser enviado, se ha programado un timeout para cumplirse en cierto plazo. OPC_INTRPT_STRM: Ha llegado un nuevo paquete

Ctrl_Arrival

1. Identifica si el mensaje que llegó son datos o son mensajes de control.

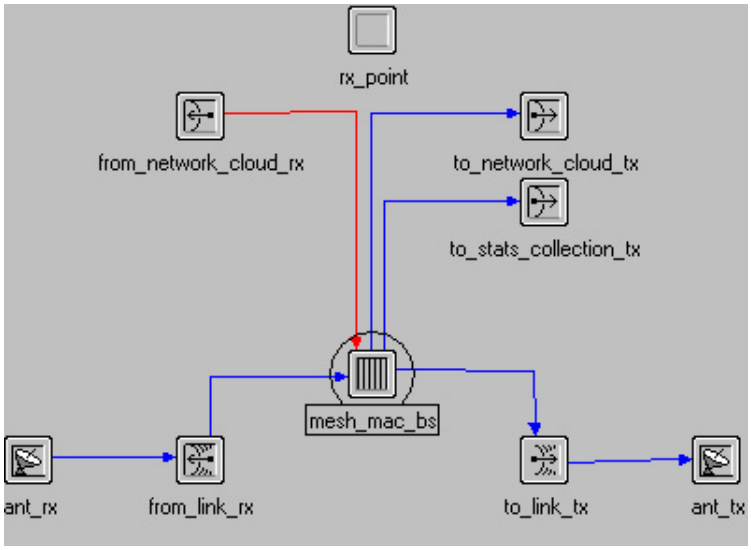


Figura 4.9: Modelo de Nodo de la Estación Base

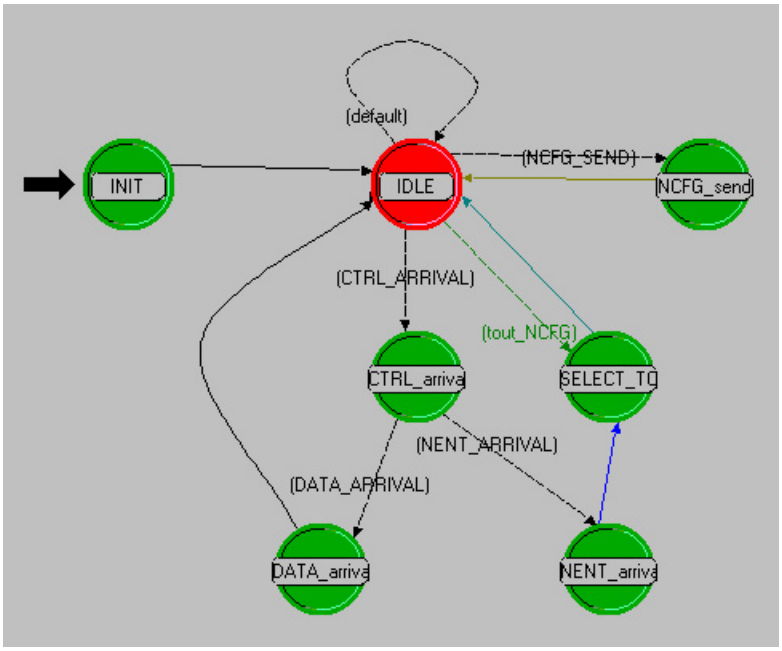


Figura 4.10: Modelo de Procesos de la Estación Base

NENT_Arrival

1. Obtiene el estado actual del nodo origen del paquete y verifica que el tipo del mensaje entrante sea superior al estado para evitar redundancias.

2. De cumplirse la condición 1. se encola el mensaje en la estructura para procesar paquetes.

3. Se incrementa el contador de mensajes MSH-NENT

Select_TO

1. Calcula el tiempo de transmisión desde el tiempo actual de simulación.
2. Calcula el tiempo NCFG de la TO dada, para enviar este tiempo como el inmediatamente próximo a competir.
3. Verifica que el número de mensajes en la cola sea mayor que 0
4. Si se cumple 3 y no hay mensaje en espera de ser enviado. Realiza los pasos del 4. al 10.
5. Ejecuta el algoritmo de Selección de Tiempo para mensajes MSH-NCFG, indicando si el nodo ganó o no en este tiempo.
6. Si la BS ganó en este tiempo y la bandera de espera de mensajes esta apagada, remueve el paquete de la cabeza de la cola y obtiene los parámetros de la dirección MAC y del tipo de mensaje MSH-NENT
7. De acuerdo al tipo de mensaje obtenido en 5. Se realizan diferentes acciones: Si MSH-NENT es tipo Net Entry Request se obtiene el MAC Address, se agrega a la lista de control de nodos y se actualiza el estado del nodo, se prende la bandera para el envío de la respuesta.
   Si MSH-NENT es tipo Net Entry ACK, solo actualiza el estado del nodo pero no envía mensaje de respuesta, queda en espera del mensaje Net Entry Close.
   Si MSH-NENT es tipo Net Entry Close, se agrega a la lista de nodos que ya están dentro de la red, se actualiza el estado del nodo y se prende la bandera para el envío de la respuesta.
8. Si la bandera de envío de mensaje esta prendida se envía una interrupción para que el mensaje MSH-NCFG sea preparado y enviado, en este punto se prende una bandera de envío de mensaje hasta que el paquete ha sido enviado, para evitar que 2 nodos diferentes intenten enviar al mismo tiempo. Si la bandera de envío esta apagada se envía una interrupción para volver a entrar al estado.
9. Se decrementa el número de mensajes en la cola.
10. Si no se cumplió 3. Se envía una interrupción en la siguiente oportunidad de transmisión MSH-NCFG al mismo estado, para empezar de nuevo el proceso en 1.

NCFG_Send Se prepara el mensaje para enviar la respuesta por medio de un paquete MSH-NCFG al MSH-NENT entrante.

1. Identifica el tipo de mensaje MSH-NENT y de acuerdo al tipo, obtiene el tipo de mensaje MSH-NCFG que corresponde.

2. De acuerdo al tipo de mensaje MSH-NCFG que corresponde construye el paquete para que sea enviado.

3. Envía el paquete MSH-NCFG.

4. Si el tipo programado de MSH-NCFG es Network Descriptor se envía una interrupción para que sea enviado el siguiente broadcast.

4.4.3. Capa MAC para estación suscriptora

En la figura 4.11 se muestra el diagrama de nodos que se utilizó en el modelo de la red mesh, en donde se utiliza el modelo de propagación que explicado más adelante.

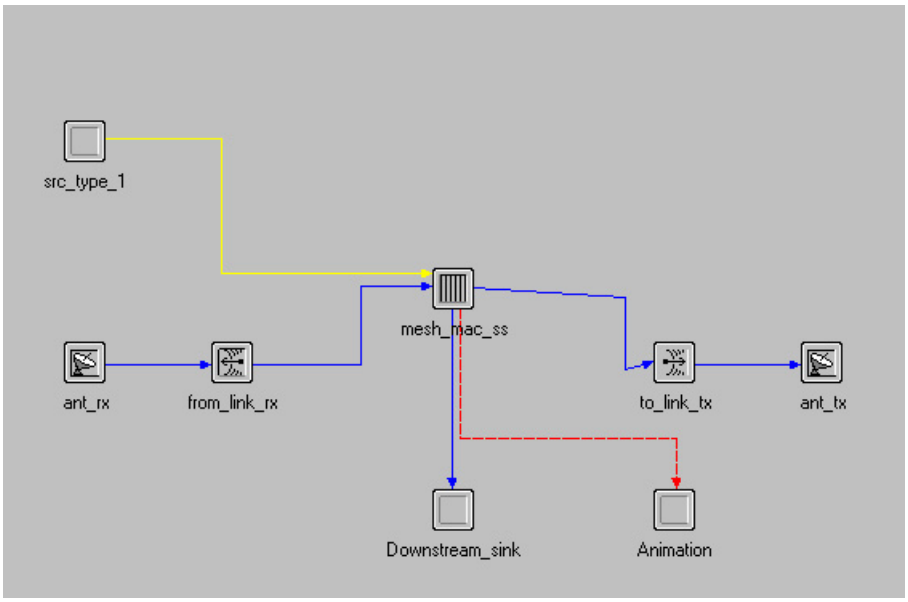


Figura 4.11: Modelo de Nodo de la Estación Suscriptora

Básicamente se trabajo en mesh_ss_mac la funcionalidad de la fase de inicialización fue implementada como se muestra en la figura 4.12. Se puede observar el flujo de los diferentes estados, dependiendo del tipo de paquete que llegué, el paquete a ser enviado o de ser necesario programar un re-envío. El manejo de tiempos se realiza dentro de estos estados. Al programar el evento de envío o re-envío.

mesh_ss_mac

A continuación se describe brevemente el comportamiento que se realiza en cada uno de los estados del diagrama de la imagen 4.12:

Init

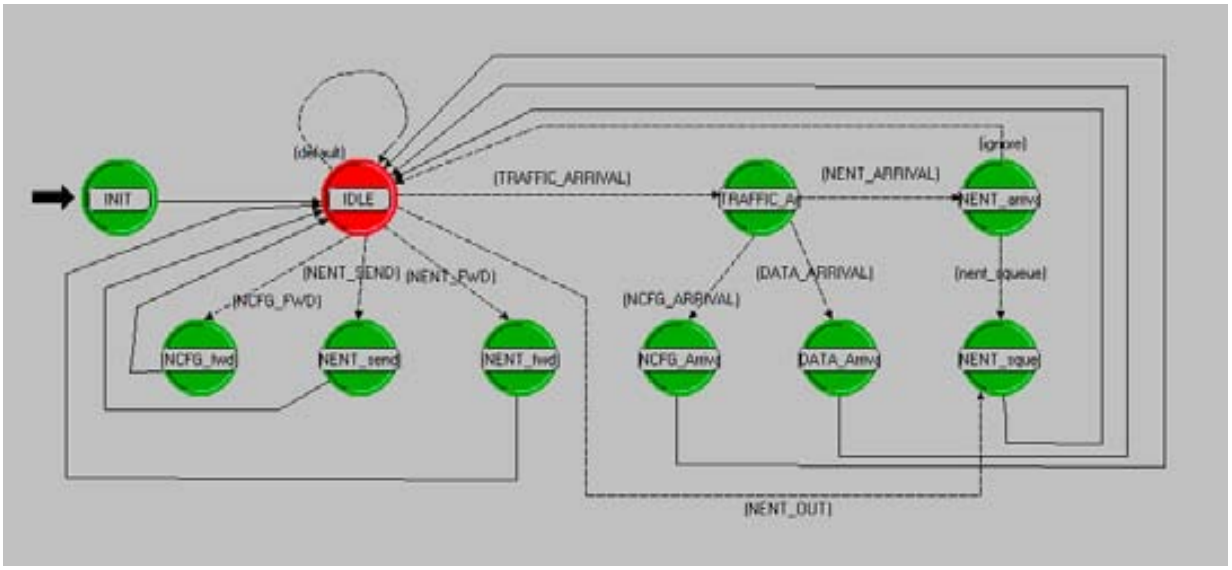


Figura 4.12: Modelo de Procesos de la Estación Suscriptora

1. Verifica que se cumpla la interrupción inicial, que este caso es cuando llega una interrupción tipo *OPC_INTRPT_STRM* que indica que ha llegado un nuevo paquete.

2. Guarda el paquete que ha llegado.

3. Envía la interrupción al estado que lleva el control de recepción de paquetes.

Idle

1. Identifica el tipo de interrupción que se presenta y de acuerdo al código con que fue lanzada prende o apaga las banderas que determinan los estados a los que se cederá el control.

Traffic_Arrived

1. Identifica que tipo de tráfico ha llegado y por medio de una bandera indica cuál fue.

Data_Arrival

1. Si el paquete que ha llegado contiene datos se queda en este estado para ser procesado.

2. Se obtiene la distancia del nodo actual, al origen del paquete para calcular su retardo.

3. Se calcula su retardo

NCFG_Arrival Si el paquete recibido es de tipo MSH-NCFG se procesa de la siguiente forma.

1. Verifica que la MAC Address del mensaje sea igual al nodo actual o que la MAC destino es 0, es decir esta recibiendo un mensaje broadcast de la BS (también se verifica en este caso que el nodo no haya comenzado su registro en la red o que de lo contrario ya sea un nodo registrado y en este caso solamente va a servir de puente para realizar un *forward*).

2. En caso de cumplirse 1, se consideran los siguientes casos del 3. al 9.

3. Se obtiene el subtipo de mensaje MSH-NENT entrante, se emprenden diferentes acciones dependiendo del tipo obtenido, cada acción se describe del 4 al 9.

4. Si el tipo es *Network Descriptor*, se verifica que también cumpla con la condición que el MAC Address del mensaje sea igual a cero y que el estado del registro del nodo sea igual a 0, es decir que aún no haya iniciado la entrada a la red.

   a) Se obtienen los números de saltos del mensaje entrante y se incrementa, para posteriormente agregar este parámetro a los paquetes del nodo presente, se obtienen parámetros para el conocimiento de la red.

   b) Ya que tenemos que el nodo apenas va a comenzar su entrada a la red, la variable que representa al nodo padre se obtiene del mensaje entrante, con el campo *MAC Source*.

   c) Se actualiza el registro del nodo como 1, es decir indica que ya cumplió el primer paso del registro.

   d) Se obtiene el tiempo de envío para mensajes MSH-NENT, se resta el retardo de propagación y con este tiempo la interrupción es enviada.

   e) Finalmente se destruye el paquete entrante.

5. Si el tipo es *Network Entry Open*, se verifica que el *MAC Address* del mensaje sea igual a la MAC local y que el registro de nodo este actualizado a 1.

   a) Se obtiene el tiempo de envío para mensajes MSH-NENT, se resta el retardo de propagación y con este tiempo la interrupción es enviada.

   b) Se actualiza el registro del nodo como 1, es decir indica que ya cumplió el primer paso del registro.

   c) Finalmente se destruye el paquete entrante.

6. Si el tipo es *Network Entry ACK*, se verifica que el *MAC Address* del mensaje sea igual a la MAC local y que el registro de nodo este actualizado a 2.

   a) En este paso termina el proceso de entrada del nodo a la red y se inserta en la lista general donde se encuentran los nodos que ya pueden transmitir datos o buscar nuevos nodos para que por medio de él , el nuevo nodo ingrese a la red.

Los anteriores son los casos base para cuando un nodo esta en su proceso de entrada a la red y llega un mensaje MSH-NENT, ahora se analiza como se ha resuelto el problema de cuando un nodo ya esta registrado en la red y llega un nuevo mensaje con MAC = 0 *(Network Descriptor)*

7. Se verifica que el nivel de registro sea 3, esto indica que el nodo ya está completamente dentro de la red.
8. Por otro lado se verifica que el mensaje provenga del padre, para respetar la jerarquía.
9. Se empaqueta el mensaje que se re-enviará conservando la MAC Address original y actualizando las direcciones de origen y destino.
10. Convierte el tiempo actual a oportunidad de transmisión y posteriormente esto a tiempo real, en el cual será enviada la siguiente interrupción para re-enviar el MSH-NCFG.
11. Se actualiza la bandera para activar el estado de re-envío.
12. El paquete entrante es eliminado.
13. En caso de no cumplirse 1. Significa que la MAC Address no coincide con la del nodo actual, ni tampoco es igual a cero. Por tanto solo hay otra opción de aceptar el mensaje y es cuando la MAC destino y la actual coinciden.
14. Valida que efectivamente el mensaje proviene de un nodo hijo.
15. Se reconstruye el paquete.
16. Convierte el tiempo actual a oportunidad de transmisión y posteriormente esto a tiempo real, en el cual será enviada la siguiente interrupción para re-enviar el MSH-NCFG.

NENT_Arrival Estado que sólo se activa cuando el segundo nivel de nodos se esta dando de alta dentro de la red.

1. Obtiene el estado actual del nodo origen del paquete y verifica que el tipo del mensaje entrante sea superior al estado para evitar redundancias.
2. De cumplirse la condición 1. se encola el mensaje en la estructura para procesar paquetes.
3. Se incrementa el contador de mensajes MSH-NENT.

NENT_Squeue Maneja la entrada y salida de la cola de mensajes MSH-NENT

1. Verifica que el contador de mensaje se mayor que cero.
2. Si la bandera de envío de mensaje NENT no esta bloqueada:
   a) Se obtiene el tiempo aleatorio para el envío de mensajes MSH-NENT
   b) Se remueve el mensaje de la cola y se guarda en un paquete local
   c) Se obtienen los parámetros del mensaje entrante

d) Se emprenden diferente acciones dependiendo del subtipo de paquete entrante:

1) Request. Obtiene y guarda el MAC Address entrante como hijo y/o su Sponsor ID junto con la MAC Address
2) ACK. Obtiene MAC Address
3) Close. Obtiene MAC Address

e) Se actualizan las MAC Origen y Destino

3. Se reconstruye el paquete.

4. Convierte el tiempo actual a oportunidad de transmisión y posteriormente esto a tiempo real, en el cual será enviada la siguiente interrupción para re-envía el MSH-NENT.

NCFG_Fwd

1. Ejecuta el algoritmo de Selección de tiempo e indica si ganó o no en el tiempo presente.

2. Si ganó activa la bandera de envío de paquete y envía el paquete.

3. De lo contrario envía una interrupción de tiempo a él mismo pero en la siguiente oportunidad de transmisión para volver a contender

NENT_Send

1. Genera un paquete tipo MSH-NENT

2. DE acuerdo al subtipo de MSH-NCFG recibido, elige el subtipo de MSH-NENT a enviar con los siguientes criterios:

a) NCFG Network Descriptor.. NENT Request

b) NCFG Network Entry Open ... NENT ACK

c) ... NENT Close

d) NCFG Network Entry ACK ...

3. Válida el estado actual con el estado cuando llega la interrupción para ver si es necesario enviar un time out

4. De resultar positivo el punto anterior, prepara el paquete MSH-NENT para ser enviado, encapsulando el tipo correspondiente.

5. Si se trata se un mensaje MSH-NENT subtipo *ACK*, se programa tiempo después un mensaje *Close*

6. Ahora envía una interrupción timeout que se activará en el segundo 1 y siendo relativo al número de saltos que el nodo se encuentre ya que la diferencia de estar 3 saltos a estar 2 puede crecer en forma considerable.

NENT_Fwd

1. Invoca al algoritmo de elección para mensajes MSH-NENT y obtiene un tiempo aleatorio para el envío del mensaje.

2. una interrupción para que el mensaje MSH-NENT sea enviado.

4.5. Tiempos de sincronización

Como se mencionó en el capítulo 3, el protocolo IEEE 802.16 utiliza Acceso Múltiple por división de tiempo (TDMA). En la figura 4.13 se muestra la división de tiempos para el envío-recepción de mensajes MSH-NENT y MSH-NCFG que se utilizó en el modelo. En el estándar se propone una oportunidad de transmisión (TO), como se muestra en [10], en donde también se realiza una comparación entre diferentes planificadores, con la configuración dada, y para mensajes tipo MSH-NENT por frame y 7 o 15 oportunidades de transmisión para mensajes de tipo MSH-NCFG.

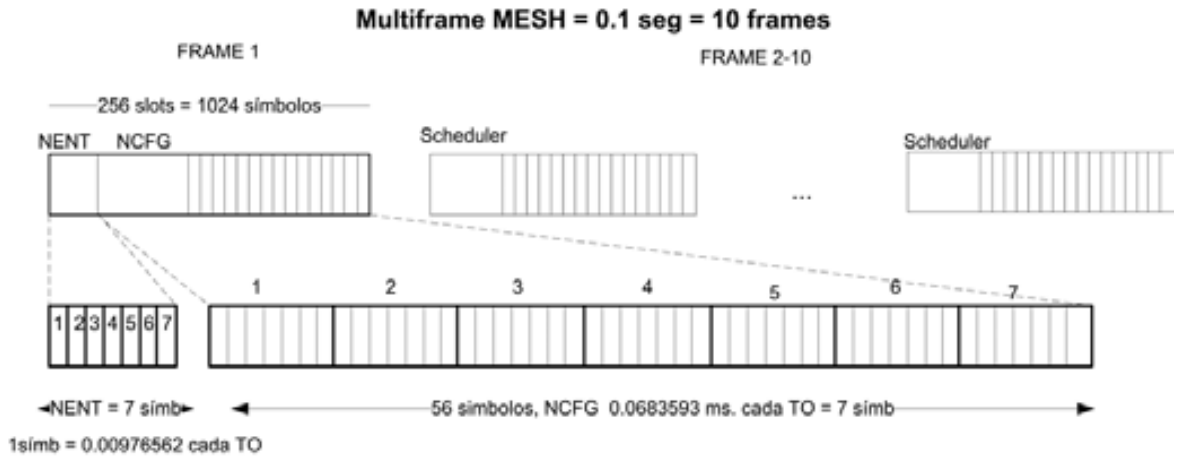


Figura 4.13: División de tiempos en los mensajes de control MSH-NCFG y MSH-NENT utilizados para el modelo

En el presente trabajo se está proponiendo que sean **7** oportunidades de transmisión para el envío de mensajes MSH-NENT, para hacer más eficiente el tiempo de envío. En el caso de las oportunidades de transmisión de los mensajes tipo MSH-NCFG se utilizaron las 7 sugeridas en el estándar, debido a que este mensaje puede ser muy grande cuando contiene la información de varios vecinos (ver tablas de 3.8.2). Cabe mencionar que esta división de tiempos solo tiene sentido para una tasa de transferencia de datos mayor a 59Mbps, por:

$$\frac{No.Simbolos * No.Bytes * 8}{Frame\_duration/No.\_slots} = \frac{4*72*8}{0{,}01s/256} \simeq 59Mbps$$

El primer frame de cada multiframe contiene mensajes de control (cada 10 frames), ya que tomando en cuenta el tamaño del paquete MSH-NENT (ver tablas en 3.8.3).

El tamaño de los paquetes MSH-NENT varía, dependiendo del tipo de mensaje que se necesita transmitir:

MSH-NENT:Request 216 bits = 27 bytes, que es paquete de mayor bytes a transmitir.

MSH-NENT:Ack 40 bits = 8 bytes

MSH-NENT:Close 40 bits = 8 bytes

Por tanto los mensajes MSH-NENT pueden enviarse en un solo símbolo, obteniendo así, 7 oportunidades de transmisión en vez de una.

4.5.1. Algoritmo de selección

Cada vez que un nodo ha terminado su proceso de inicialización, quiere decir obviamente que ya es parte de la red, por tanto su ID debe ser agregado dentro de la lista de nodos que van a competir por transmitir mensajes de tipo MSH-NCFG, esta lista por tanto es dinámica.

El algoritmo que elimina la posible existencia de colisiones el algoritmo de selección *Mesh Election* para el funcionamiento de este algoritmo es necesario el uso de tres parámetros importantes que influyen directamente en el cálculo del tiempo en que un nodo puede transmitir un mensaje MSH-NCFG, estos parámetros son:

1. **Xmt Holdoff Exponent**. Exponente de tiempo transmisión bloqueado.
2. **NextXmtMx**. Siguiente tiempo de transmisión.
3. **EarliestSubsequentXmtTime**. Tiempo de transmisión más cercano después del tiempo de espera.

Las cuáles están definidas en el mensaje *MSH-NCFG:Network Descriptor*. Estas variables ya han sido descritas brevemente en el capítulo 3, a continuación se enfatiza su impacto dentro del algoritmo de selección.

Xmt Holdoff Exponent

El tiempo de espera para transmitir mensajes MSH-NCFG (*Xmt Holdoff Time*) dependen del valor de este parámetro, el cuál como se ha revisado en el capítulo 3, esta definido en el paquete *MSH-NCFG:Network Descriptor* y su tamaño es de 5 bits, es decir puede variar entre 0 y 31. Además de utilizarse para calcular el intervalo en que un nodo puede competir dentro del Algoritmo de selección de tiempos de transmisión para mensajes MSH-NENT.

La ecuación que se utilizó para calcular el valor *Xmt Holdoff Time*, definida en el estándar [2], es la siguiente:

$$XmtHoldoffTime = 2^{XmtHoldoffExponent+4}$$

Por ejemplo,

- $XmtHoldoffExponent = 0, XmtHoldoffTime = 2^4 = 16$
- $XmtHoldoffExponent = 1, XmtHoldoffTime = 2^{1+4} = 2^5 = 32$
- $XmtHoldoffExponent = 2, XmtHoldoffTime = 2^{2+4} = 2^6 = 64$

Se puede observar que el *XmtHoldoffTime* crece exponencialmente a razón del valor de *XmtHoldoffExponent*, mientras este exponente tenga valores pequeños el tiempo de espera es menor, pero esto no determina cual valor de *XmtHoldoffExponent* es mejor por lo que se debe de encontrar un buen desempeño conjuntamente con las demás variables del modelo.

Next Xmt Mx

Otro de los parámetros conocidos en el mismo mensaje de configuración *MSH-NCFG:Network Descriptor* es *Next Xmt Mx* que tiene una longitud de 3 bits, por lo que los valores oscilan entre 0 y 7. El valor de *Next Xmt Time*, calculado en el algoritmo de elección y que determina el tiempo de la siguiente transmisión del nodo para un mensaje MSH-NCFG, depende de las variables *Next Xmt Mx* y *Xmt Holdoff Exponent*. Para determinar el rango de tiempo en que el nodo es elegible para competir para el cálculo del la siguiente oportunidad de transmisión, esta dado por:

$$2^{XmtHoldoffExp} * NextXmtMx < NextXmtTime \leq 2^{XmtHoldoffExp} * (NextXmtMx + 1)$$

Veamos algunos ejemplos de como se comporta *Next Xmt Time* en su intervalo de tiempo cuando los valores de los que depende directamente, *Next Xmt Mx* y *Xmt Holdoff Exponent*, oscilan entre sus diferentes rangos permitidos:

- $XmtHoldoffExponent = 0, NextXmtMx = 4 \Rightarrow 4 < NextXmtTime \leq 5$
- $XmtHoldoffExponent = 0, NextXmtMx = 7 \Rightarrow 7 < NextXmtTime \leq 8$
- $XmtHoldoffExponent = 7, NextXmtMx = 4 \Rightarrow 2^7 * 4 = 512 < NextXmtTime \leq 640$
- $XmtHoldoffExponent = 15, NextXmtMx = 1 \Rightarrow 2^{15} * 1 = 32768 < NextXmtTime \leq 65536$

En los ejemplos anteriores se puede observar que mientras el *Xmt Holdoff Exponent* y *Next Xmt Mx* sean mayores, el rango de tiempo en que el nodo es elegible para transmitir los mensajes MSH-NCFG es mayor, así como también se observa que el valor de $2^{XmtHoldoffExponent}$ se toma otro intervalo del mismo tamaño para que el tiempo este dentro de este rango. Por ejemplo:

Si $XmtHoldoffExponent = 7 y NextXmtMx = 4$, el rango es:

$$2^7 * 4 = 128 * 4 = 512 < NextXmtTime \leq 640$$

El bloque de tiempo del ejemplo anterior se muestra en la figura 4.14 encerrado en un circulo, y se muestran los posibles rangos para determinar el siguiente tiempo de transmisión *Next Xmt Time*.

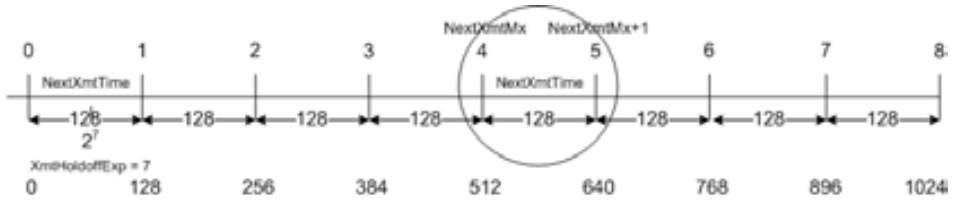


Figura 4.14: Tiempos posibles para *NextXmtTime* con un valor de *Xmt Holdoff Exponent* = 7

Entonces, *Xmt Holdoff Exponent* determina el tamaño del bloque en que el nodo es elegible para competir por las oportunidades de transmisión y el y *Next Xmt Mx* determina la posición en que se encontrará dicho bloque.

Earliest Subsequent Xmt Time

Esta variable también depende directamente del valor del parámetro *Xmt Holdoff Exponent*, y es el que determina el rango de transmisión del nodo ya que éste ha transmitido por primera vez dentro del tiempo *Next Xmt Time*. La ecuación para determinar este valor, es similar a la que se utiliza para calcular el tiempo *Xmt Holdoff Time*, y se muestra en continuación:

$$EarliestSubsequentXmtTime = 2^{XmtHoldoffExponent+4} * (NextXmtTime)$$

En este caso se toma *Next Xmt Time* como el valor del tiempo en que el nodo transmitió por última vez. Como se puede observar esta variable es el complemento de la variable *Xmt Holdoff Time* como se muestra en la figura 4.15

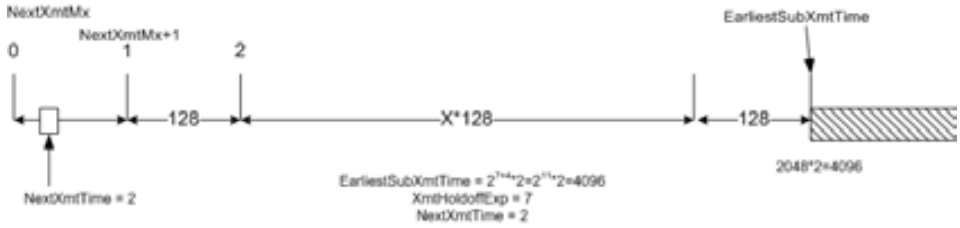


Figura 4.15: Cálculo del nuevo tiempo de transmisión: *EarliestSubsequentXmtTime* con *Xmt Holdoff Exponent* = 7

En el apéndice C se muestran las partes más importantes del código del modelo.

Capítulo 5

Resultados del análisis y estudio del comportamiento del estándar IEEE 802.16 con topología Mesh

5.1. Introducción

En este capítulo se presentan los resultados del modelo realizado con las características de inicialización del protocolo IEEE 802.16 con arquitectura Mesh. Se realizó un programa en el lenguaje C++ y posteriormente el modelo en la herramienta OPNET, este modelo fue descrito en el capítulo 3.

Se compara la simulación ideal que esta representada por el programa, con el modelo realizado en OPNET, y los diferentes resultados obtenidos al cambiar los tiempos de re-envío de mensajes MSH-NENT. Así como el tiempo de espera para el envío de un nuevo mensaje de tipo broadcast *MSH-NCFG:NetworkDescriptor*. Este mensaje es el que marca la pauta para iniciar la comunicación con nodos que aún no ingresan a la red. Sea directamente desde la MBS o a través de los re-envíos realizados desde los nodos que ya están dentro de la red y son parte de la trayectoria a la MBS. Estos tiempos mencionados se optimizan de tal forma que el uso de oportunidades de trasmisión resulte más eficiente. Durante este proceso de inicialización el comportamiento de tiempos para la expansión de la red se ve afectado por dos parámetros: *XmtHoldoffExponent* y *XmtTimeMx*, los cuáles también son analizados.

Por otro lado, se realiza una propuesta de valores para los tiempos de espera de los mensajes de control, esto es en caso de que los paquetes sufran colisiones y se realice un nuevo envío oportuno del mismo, además de realizar un encolamiento eficaz de paquetes.

Se observará el comportamiento de una red Mesh enfocadose principalmente a los tiempos de inicialización, que es una primera aproximación al estudio de una comunicación administrada por planeación centralizada.

5.2. Escenario de simulación

La simulación, tuvo como objetivo principal generar una red con arquitectura Mesh que pudiera mostrar el comportamiento de esta topología, realizando varios saltos. Entendiendo saltos como la métrica entre los nodos que se deben de utilizar para llegar a la MBS contándose al nodo actual. Para este objetivo se utilizó una red que está formada de una MBS y 100 nodos, distribuidos de la siguiente manera: La MBS tiene 10 nodos en su cobertura, es decir a un salto. En el segundo salto se agregaron 30 nodos que fungen como hijos de los nodos que se encuentran a un salto de distancia de la MBS. En el tercer y último salto se encuentran 60 nodos que están dentro de la cobertura de los 30 nodos que se encuentran a dos saltos de distancia de la MBS.

Se consideró la estación base centralizada, pero en caso de que exista un nodo que se encuentre en el área periférica de la red, que esté en contacto con el exterior, se puede considerar un subcaso del modelo generado.

En la figura 5.1 se muestra gráficamente ésta jerarquía y como se determinan la métrica de saltos.

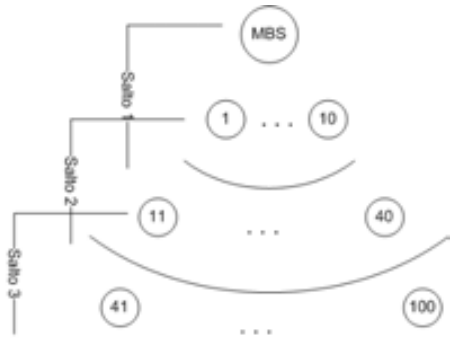


Figura 5.1: Jerarquía del Modelo utilizado

Esta jerarquía se utilizó tanto para la simulación en el programa en C++ como para el modelo en OPNET.

5.3. Comportamiento dinámico en redes IEEE 802.16 con arquitectura Mesh

Se generó un programa en el lenguaje de programación C++, donde se implementaron los algoritmos para calcular los tiempos de envío tanto el algoritmo de selección para mensajes MSH-NCFG como el tiempo aleatorio para elegir tiempos MSH-NENT. Y a partir de comprobar la funcionalidad correcta de la interacción entre los dos algoritmos de tiempo se adapto al modelo (descrito en el capítulo 4) para el análisis del comportamiento dinámico en la transmisión de los mensaje de inicialización en una red con topología Mesh.

5.3.1. Análisis de tiempos en la simulación del programa en C++

Para tener una referencia de tiempos de la inicialización de la red, como se mencionó anteriormente. Se generó un programa que simula el comportamiento del algoritmo *Mesh Time Selection* para que los nodos puedan competir por una misma oportunidad de transmisión de los mensajes MSH-NCFG. Este algoritmo determina al nodo ganador, es decir, el nodo que puede transmitir en esa oportunidad de transmisión. En este modelo también se consideró la generación aleatoria para el cálculo de la siguiente oportunidad de transmisión de los mensajes MSH-NENT. Cabe mencionar que este algoritmo, calcula un tiempo aleatorio dentro de las 7 oportunidades de transmisión que contiene el siguiente frame o bien si aún existen oportunidades de transmisión en el mismo frame pueden ser utilizadas por el mismo nodo.

Esta representación es suponiendo un mundo ideal para el envío/recepción de mensajes, es decir, sin existencia de colisiones, y por tanto sin tiempos de espera definidos, y con el envío de los mensajes *MSH-NCFG:Network Descriptor* en el periodo de tiempo exacto. Cuando los nodos que se encuentran a una cantidad de saltos terminaron de darse alta, se continúa con los nodos que tienen un salto más de distancia, además de que tampoco se consideró el cálculo de retardo de propagación.

Otra característica de este programa es que la distribución de los nodos es equitativa. La MBS tiene 10 nodos hijos, cada uno de estos nodos tiene 3 nodos hijos a su vez, y finalmente estos nodos tienen 2 nodos hijos cada uno. Como se muestra en la figura 5.2. Los nodos se consideraron de forma estática y enumerados secuencialmente: como se muestra el nodo 1 tiene como hijos los primeros 3 nodos de la siguiente capa, es decir los nodos 11, 12, 13 y así sucesivamente.

Figura 5.2: Árbol considerado en el programa para el cálculo de tiempos para el programa en C++

En la figura 5.3 se muestra el intercambio de mensajes que realizan los nodos que están a un salto de distancia de la MBS, El primer nodo (1) en darse de alta ocupa un tiempo de alrededor de 0.7 segs mientras el último nodo (3) 3.7 segs. También se observa que el intercambio de paquetes se encuentra dentro del tiempo definido para las oportunidades de transmisión permitidas para los mensajes de configuración y entrada a la red. Es decir, el propósito de esta simulación fue validar los algoritmos de selección de tiempo.

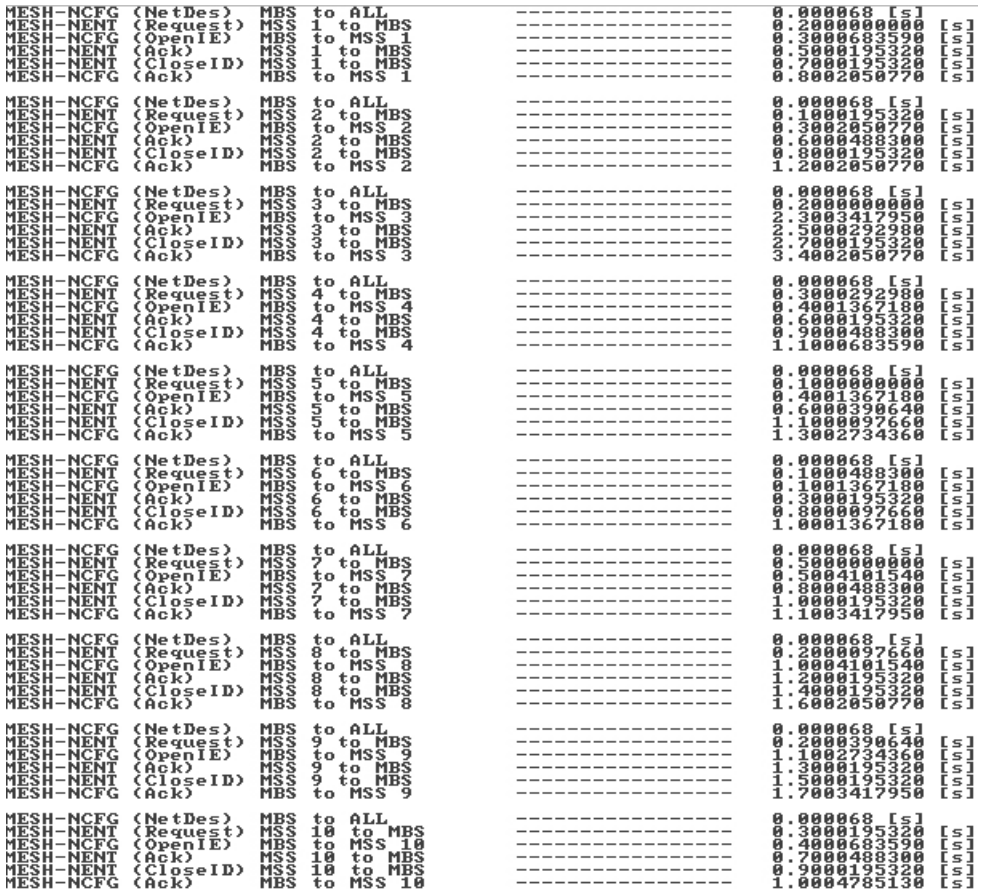


Figura 5.3: Ejemplo de inicialización de nodos a un salto de distancia MBS

A continuación se presentan dos figuras como ejemplo del intercambio de mensajes que se realiza entre los nodos que se encuentran a dos saltos de distancia de la MBS. En la figura 5.4 se muestra el mejor caso, en donde el nodo 39 tarda 14.4 segs en ingresar a la red. El peor caso se muestra en la figura 5.5 en donde el nodo 22 termina su ingreso a la red a los 73.7 segs, en donde hay un retardo de 70 segs aproximadamente para entrar a la red. El nodo 22 tiene como nodo padre al nodo 4, que realiza el re-envío de los mensajes MSH-NENT a la MBS y los mensajes MSH-NCFG de la MBS al nodo 22. En cada una de las figuras se muestra el tiempo en que ocurre cada uno de estos eventos.

Análogamente en la figura 5.6 se presentan un ejemplo de los mensajes que intercambia un nodo que se encuentra a tres saltos de distancia de la MBS, se toma el peor caso, es decir el último nodo en entrar a la red. El último nodo en ingresar a la red fue el nodo 98, que tiene de padre al nodo 39. El nodo 39 a su vez tiene como padre al nodo 10 que esta a un salto de distancia de la MBS. El retardo desde que el nodo 98 inició el proceso de entrada a la red es próximamente de 176 segs, este tiempo es más del doble que el peor caso de dos

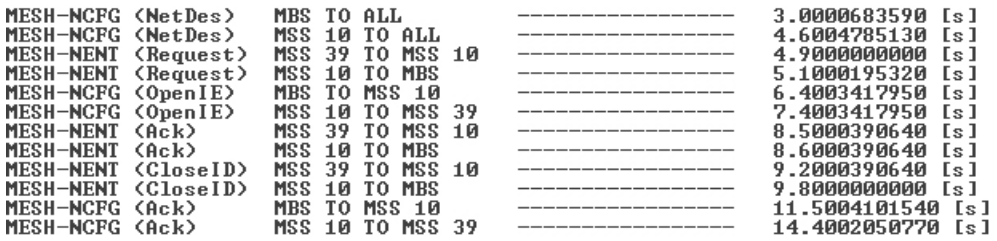


Figura 5.4: Ejemplo de inicialización del mejor caso a dos saltos de distancia de la MBS

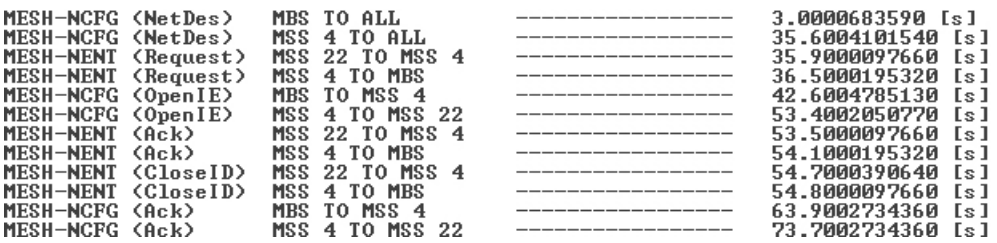


Figura 5.5: Ejemplo de inicialización del peor caso a dos saltos de distancia de la MBS

salto. Este retardo se debe a que el nodo tiene que competir contra 60 nodos (3er salto), en lugar de 30 (2do salto), además de que tiene que re-enviar otros 6 mensajes para llegar a la MBS y por tanto competir por esas oportunidades de transmisión adicionales. En la figura 5.6 se puede observar el envío de mensajes y el doble reenvío a través de los nodos 10 y 39 respectivamente, los cuáles tienen la función de puentes.

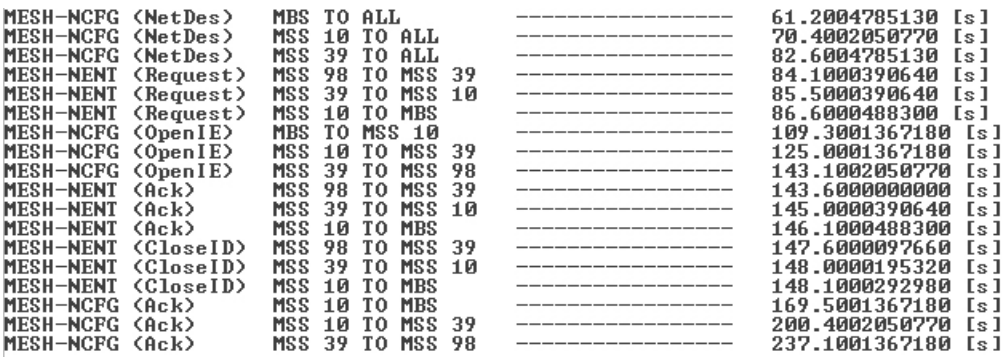


Figura 5.6: Ejemplo de inicialización de nodo a 3 saltos de la MBS

Esta simulación se realizó con el fin de verificar el correcto funcionamiento y la posible división de los tiempos. Cabe mencionar que esta simulación se obtuvo utilizando como parámetros los siguientes valores: número de oportunidades de transmisión NENT = 7, XmtHoldoffExponent = 0, y XmtTimeMx oscilando entre 0 y 7. La importancia del valor de estos parámetros serán mencionados en la última parte de este capítulo.

En la figura 5.7 se muestra el tiempo de la simulación realizada en el programa, en la que se refleja el crecimiento de tiempo por cada nodo que se utiliza como puente para llegar a la MBS. En la primera parte de la gráfica de nodo 1 al nodo 10 el tiempo de ingreso a la red es muy bajo, para darse de alta 10 nodos el tiempo es de 3.4 segs. Se puede ver claramente el efecto de utilizar el algoritmo de elección puesto que en cada oportunidad de transmisión contienden los 10 nodos y algunas oportunidades no son utilizadas, pero gracias a esto no hay colisiones, asegurando que el mensaje llega a su destino. El tiempo se incrementa considerablemente cuando los nodos se encuentran a dos saltos de distancia, debido a que cada nodo necesita 12 intercambios de paquetes, además de que en este caso se tienen 30 nodos compitiendo por las oportunidades de transmisión para envío de mensajes MSH-NCFG. Para los nodos que están a 3 saltos de distancia se observa un incremento considerable en el tiempo de ingreso a la red de los nodos de este nivel. Lo anterior se debe a que los nodos necesitan ahora enviar/re-enviar 18 mensajes en total, seis por cada salto. También se tiene que considerar que esta vez compiten 60 nodos

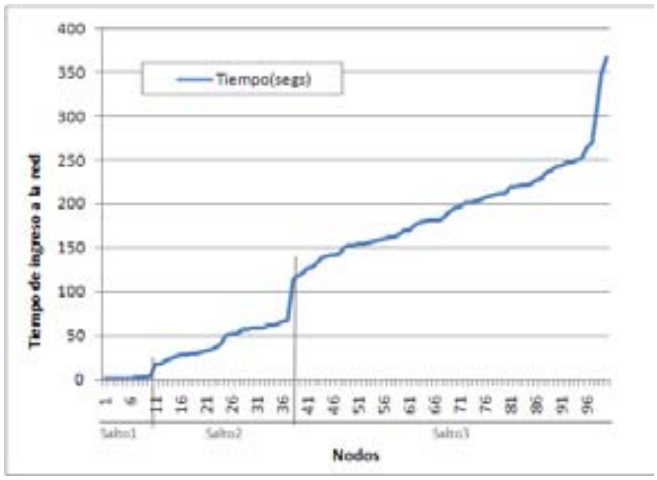


Figura 5.7: Tiempo en ingresar a la red de 100 nodos

Este programa obtuvo en su mejor tiempo alrededor de los 237 segs, para formar el árbol mostrado en la figura 5.2 para que los 100 nodos ingresaran en la red.

5.3.2. Análisis del modelo en OPNET

Una vez que se obtuvo el comportamiento deseado en la selección de las oportunidades de transmisión a partir de los algoritmos de *Mesh Time Selection* para los mensajes MSH-NCFG y la obtención de tiempos aleatorios para los mensajes MSH-NENT. Se utilizó en el modelo de simulación en OPNET (como clases externas), el cual fue descrito en el capítulo 4.

Ahora se verá el comportamiento dinámico cuando hay colisiones, que tanto afectan y cómo se relacionan con el número de usuarios. Además de los parámetros que determinan el funcionamiento dinámico y la cantidad de tiempo en ingresar todos los nodos a la red.

En la figura 5.8 se muestra la red Mesh de 100 nodos con las que se realizaron las pruebas. Los nodos están distribuidos con la jerarquía que se mencionó en la sección 5.2, los círculos indican la cobertura del grupo de nodos pertenecientes a cada salto.

Figura 5.8: Ejemplo del modelo con 100 nodos

En la figura 5.9 se muestra como se generó el árbol de ruteo a través de los nodos. Este árbol se generó conforme los nodos iban ganando la oportunidad de retransmitir el *NCFG-MSH:Network Descriptor*, dicho en otras palabras los primeros nodos en darse de alta son los que obtuvieron la comunicación con los nodos del siguiente salto, ya que una vez que un nodo recibe este mensaje de otro nodo, lo considera su padre, y con él termina el ingreso a la red. Una vez que todos los nodos están dentro de la red, se pueden optimizar las rutas por medio de algoritmos de optimización de rutas tales como OSPF (Open Short Path First) y DSR (Dynamic Source Routing).

En la figura 5.10 se observan los tiempos en darse de alta hasta llegar a los 100 nodos, cabe mencionar que en esta simulación los nodos no se dan de alta secuencialmente, ni considerando la distancia en saltos. Por ejemplo, un nodo que esta a 3 saltos de distancia de la MBS puede terminar su ingreso a la red, antes de uno que este a dos nodos de distancia de la MBS, esto también depende del tiempo en que se envíe el siguiente mensaje broadcast *MSH-NCFG:Network Descriptor.*

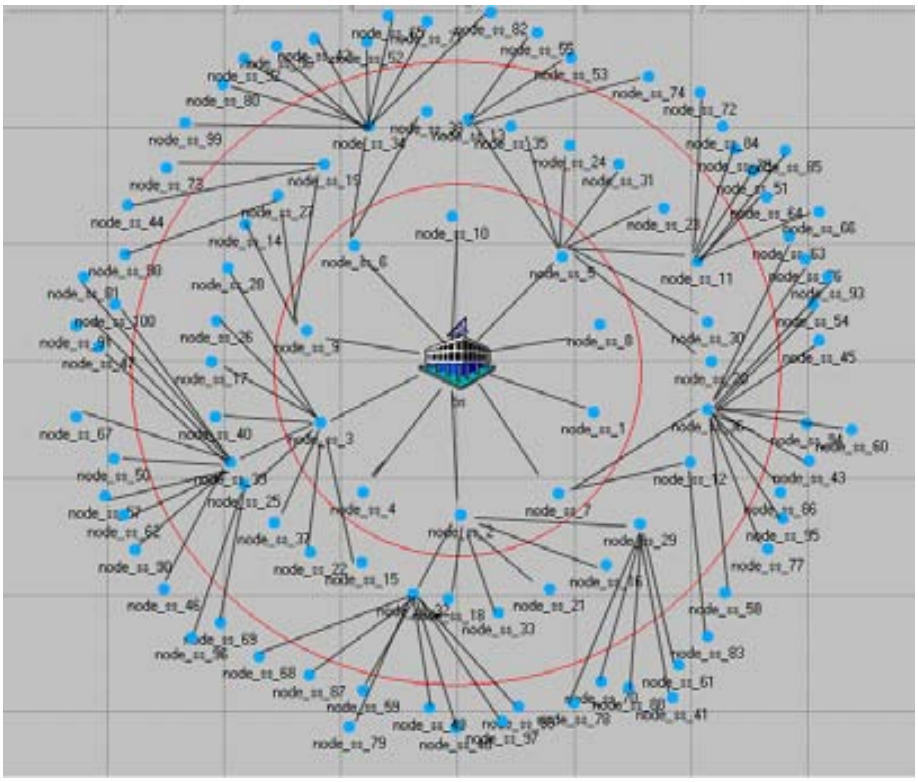


Figura 5.9: Árbol de ruteo generado

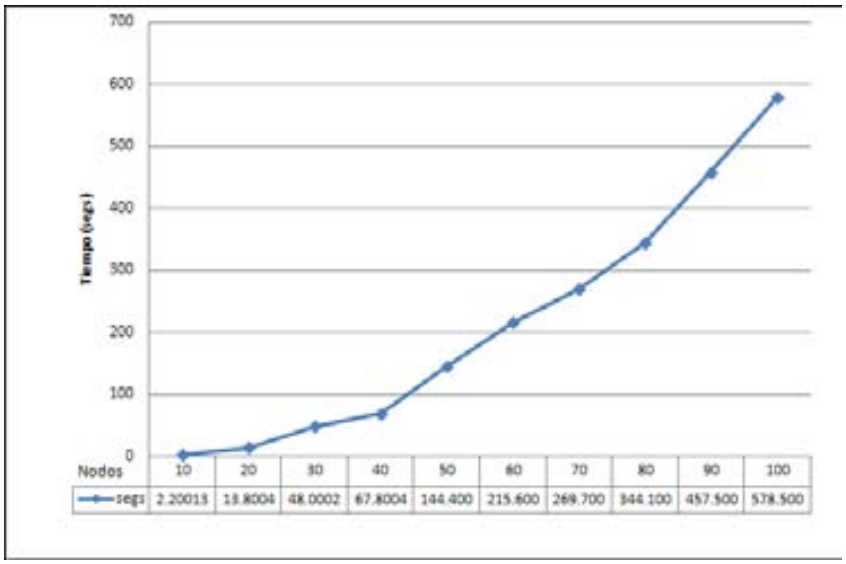


Figura 5.10: Tiempo en ingresar a la red de 100 usuarios

Tiempos MSH-NENT

Como se mencionó en el capítulo anterior el estándar propone una oportunidad de transmisión para mensajes MSH-NENT para cada frame que contiene mensajes de control (es decir 1 de cada 10 frames contiene mensajes de control MSH-NENT). En el presente trabajo se propusieron 7 oportunidades de transmisión, ya que tomando en cuenta el tamaño del paquete, como se describió en la tabla 3.10, es posible enviar hasta 7 mensajes, en un oportunidad de transmisión.

En la figura 5.11 se muestra la diferencia de ejecutar la simulación con una sola oportunidad de transmisión, con 3 y con 7 que es la variante que se esta proponiendo en esta trabajo.

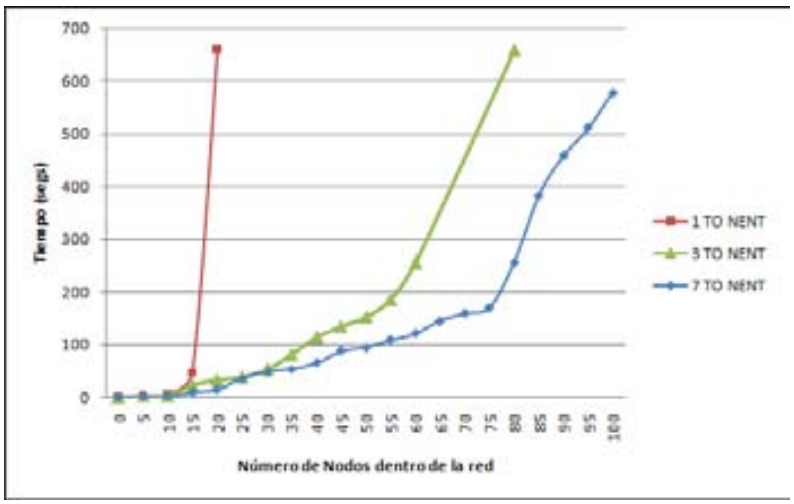


Figura 5.11: Tiempos ingresar a la red 100 usuarios, con 1,3 y 7 oportunidades de transmisión para mensajes MSH-NENT

En el caso de una sola oportunidad se puede observar que cuando hay más de dos saltos, el tiempo crece exponencialmente y que cuando el número de nodos crece, se generan cada vez más colisiones. Por consiguiente, el mecanismo propuesto por el estándar es viable para no más de 15 nodos, y con una distancia máxima de 2 saltos.

Con tres oportunidades de transmisión el tiempo crece a mayor velocidad que con 7, pero existe la oportunidad de agregar nodos en una cantidad mayor a 3 saltos de distancia. Finalmente con 7 oportunidades de transmisión el tiempo tiene un crecimiento constante hasta 75 nodos, después tiene incremento muy considerable hasta los 100 nodos, debido al número de colisiones registradas.

A continuación se hará un análisis sobre una de las variables que influyen directamente en el tiempo de ingreso a la red que es el tiempo de espera (*timeout*) manejado para el mensaje MSH-NENT. El tiempo de espera indica cuanto tardará el nodo en re-enviar el mismo mensaje porque no ha recibido un mensaje respuesta, por tanto se asume que el mensaje colisionó.

Para mensajes de tipo MSH-NENT, el tiempo de espera no puede ser muy pequeño cuando tenemos varios nodos transmitiendo a un padre, como puede ser cuando los nodos envían peticiones a la MBS, como se describió en el capítulo 4 los paquetes se agregan a una cola en espera para ser procesados. Estos paquetes van saliendo de la cola, cada vez que el nodo gana la oportunidad para transmitir. Entonces, si el tiempo de espera es muy pequeño, se puede dar frecuentemente el siguiente caso: Un nodo ha enviado un paquete MSH-NENT y éste permanece un tiempo considerable en la cola de paquetes de la MBS o de la MSS padre, y se cumple el tiempo de espera, el nodo re-enviará nuevamente el paquete MSH-NENT, asumiendo que el paquete ha colisionado. Este re-envío puede ser redundante si el primer paquete enviado no colisionó, pero sigue en espera de ser procesado. Por tanto se crean redundancias que solamente saturan la red y las colas de paquetes de los nodos destinos.

El tiempo de espera tampoco puede ser muy grande porque se desperdician una cantidad considerable de oportunidades de transmisión, además de que podrían coincidir con la siguiente etapa de envío de mensajes MSH-NENT (después de un mensaje *MSH-NCFG:Network Descriptor)* causando más colisiones. Por otro lado, el tiempo no puede ser constante ya que debe ir relacionado con el número de saltos que debe de dar el mensaje para llegar a la MBS.

Se observó que con valores fijos, las colas de los diferentes nodos se llenan y pueden dar como resultado, ya sea que el simulador se detenga demandando recursos insuficientes o colas del tamaño de 800 o más mensajes (por los re-envíos). Ahora bien, si se toma solo el valor fijo multiplicado por el número de saltos, esto da una mejor aproximación. Pero conforme se aumenta el número de saltos, las colas comienzan a saturarse, ya que tiene un crecimiento exponencial. En el sistema se verifica el estado actual del nodo (la fase de ingreso a la red en que se encuentra en el momento actual) antes de procesar un mensaje, para evitar enviar una respuesta innecesaria, si el estado actual del nodo es inferior al que corresponde la contestación envía la respuesta, de lo contrario deshecha el paquete.

Para calcular el tiempo de espera utilizado para los resultados mostrados en la figura 5.10, se originó un tiempo que dependiera del identificador del nodo para crear desfasamiento dentro de los nuevos tiempos (ya que como se mencionó, solamente se calcula el tiempo NENT en el siguiente frame del tiempo que es dado). Esto podría causar nuevamente colisiones, de este valor obtenido se relacionó con el número de saltos, ya que depende de esta variable el número de veces que se tiene que esperar a que los diferentes nodos que forman la trayectoria a la MBS tengan o ganen sus oportunidades de transmisión. A continuación se muestra los criterios utilizados para el cálculo de este tiempo, con esto se evitó un número significativo de colisiones:

$if(node\_serial\_id <= 1100)$
$temp = (node\_serial\_id - 1000) * 0{,}4;$
$if(node\_id <= 1040)$
$temp = (node\_serial\_id - 1000) * 0{,}4;$
$if(node\_serial\_id <= 1010)$
$temp = 0{,}4;$
$time\_tx\_NENT = selected\_time\_NENT(node\_serial\_id, op\_sim\_time() + temp * hops, prop\_distance);$

Se puede ver que después de calcular el factor de acuerdo a la dirección identificador del nodo (node_serial_id) se multiplica por 0.4 ya que en cuatro frames se puede dar tiempo a que los nodos que no colisionaron terminen su entrada a la red (este valor se originó del tiempo en que tarda un solo nodo en ingresar a la red). Con este valor multiplicado por el número de saltos (el número de veces que el paquete tiene que esperar otra oportunidad de transmisión para que el mensaje sea enviado desde el nodo padre).

Tiempos MSH-NCFG

El tiempo de espera para el envío de mensajes MSH-NCFG, generalmente están relacionados al momento en que se recibe un mensaje MSH-NENT, en la MBS. Cuando se recibe el mensaje MSH-NENT dentro del modelo se envía una interrupción en el tiempo de la oportunidad de transmisión más próxima para mensajes MSH-NCFG. El tiempo en que se envía la interrupción es determinado por el algoritmo de selección.

Este algoritmo determina si el nodo compite y de ser así, si el nodo resulta ganador en la oportunidad de transmisión actual. De no resultar ganador tiene que esperar a la siguiente oportunidad de transmisión para intentar enviar la respuesta.

Un caso especial es el mensaje *MSH-NCFG:Network Descriptor*, que si bien el nodo a enviarlo compite, se asigna un tiempo para enviarlo periódicamente. Este paquete es enviado en un mensaje tipo broadcast, con dirección MAC destino = 0. Al analizar este tiempo al igual que el tiempo de espera para mensajes MSH-NENT, va relacionado con el número de saltos, para dar oportunidad de terminar el ingreso a la red de la mayoría de nodos que se encuentran en medio de este proceso.

Para obtener este tiempo de envío, se utilizó una variable que es un contador de los mensajes MSH-NCFG:*Network Descriptor* enviados en broadcast. El contador de mensajes es el punto de referencia para que cada vez sea más distanciado el envío de mensajes. El tiempo que se utilizó para la simulación obtenida fue:

$$cont\_net\_descrip * cont\_net\_descrip + cont\_net\_descrip * 5{,}000$$

En el Apéndice B se muestran el comportamiento en los primeros 20 segundos de la simulación, tanto para los mensajes MSH-NENT y MSH-NCFG, con esto se verifica el buen funcionamiento del modelo de simulación.

En la figura 5.12 se muestra la gráfica de la comparación entre el crecimiento de los tiempos de ingreso a la red, separado por saltos de distancia, con el programa realizado en C++ (sin colisiones) y los resultados obtenidos en el modelo OPNET (con colisiones). Se puede apreciar que entre mayor es el número de nodos y de saltos se incrementa la separación entre unos valores y otros, esto se debe a las colisiones que se incrementan directamente proporcional al número de nodos. También se observa que donde se separan estas dos líneas es cuando

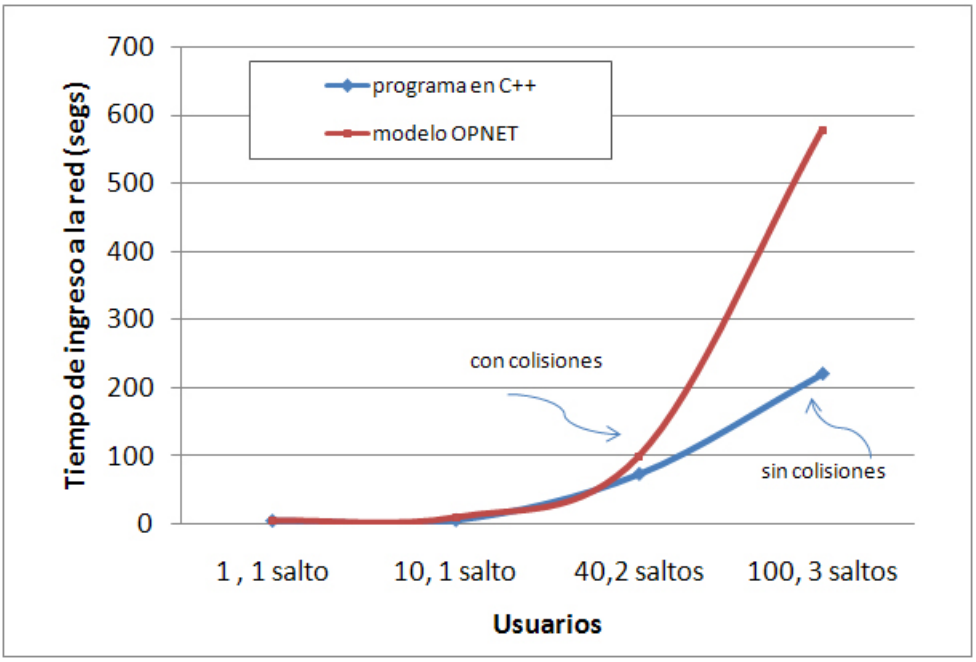


Figura 5.12: Comparación de tiempos del programa y del modelo

comienza el ingreso a la red de los nodos que se encuentran a 3 saltos de distancia, del nodo 41 al 100.

Para tener un mejor entendimiento del funcionamiento del modelo en OPNET, a continuación se muestran los fragmentos de salida del procedimiento para el ingreso del nodo 1040 a la red:

```
...
BS NEW NCFG type 1 with dest: 0 at 2.000068 MSS 1003, Rcv
MSH-NCFG type:  from mac_source 1 with mac_address 0
...
**MSS 1003: NCFG fwd tipo: 1, time: 2.000205
MSS 1040: Rcv MSH-NCFG type:  from mac_source 1 with mac_address 0
...
MSS 1040: ENV MSH-NENT type 1, status_node 1, mac_parent 1003,
at time 2.200017
    tout temp 16.000000, mac_address 1040 at time MSH-NENT: 34.300029
MSS1003: Rcv pkt NENT de: 1040
    Receives Registration Request for Child-SID 1040
MSS 1003: Rcv NENT children: 1040 at time 2.200023 MSS 1003:Prg NENT
at 2.500010
...
**MSS 1003: NENT fwd time: 2.500010
...
BS NEW NCFG type 2 with dest: 1040 at 2.800068
...
**MSS 1003: NCFG fwd tipo: 2, time: 3.400273
```

```
...
MSS 1040: Rcv MSH-NCFG type:  from mac_source 2 with mac_address
1040 MSS: 1040 NODO: 40, tiempo para NENT: 3.699997
...
MSS 1040: ENV MSH-NENT type 1, status_node 1, mac_parent 1003, at
time 3.699997
...
MSS1003: Rcv NENT de: 1040, MSS 1003:Rcv NENT de hijo: 1040, fwd to
1000 at time 3.700004 MSS 1003: Prg NENT type: 2 at 3.900059
...
**MSS 1003: NENT fwd time: 3.900059
BS, frame: 40, slot:5, receive NENT at time: 3.900063
...
MSS 1040, Prg MSH-NENT type CLOSE, status_node 2:
 tout temp 13.600000, mac_address 1034 at time MSH-NENT: 31.300020
...
**MSS 1003: ,NENT fwd time: 4.700000
BS, frame: 47, slot:5, receive NENT at time: 4.700004 BS, Rcv
MSH-NENT type:  from mac_address to squeue 3
...
BS NEW NCFG type 4 with dest: 1040 at : 5.300410
...
**MSS 1003: NCFG fwd tipo: 4, time: 5.500410
...

MSS 1040: Rcv MSH-NCFG type:  from mac_source 4 with mac_address
MSS 1040: Finished!!!!!!!!!!!!

********************** Gral List with 12 users ***************
SID 1000                SID 1008                SID 1006
SID 1005                SID 1007                SID 1001
SID 1010                SID 1003                SID 1002
SID 1009                SID 1004                SID 1040
```

Haciendo referencia a la salida mostrada anteriormente, en la figura 5.13 se observan los tiempos en que ingresa el nodo con identificador 1040 a la red utilizando el nodo con identificador 1003 como puente para llegar a la MBS, y un ejemplo de los tiempos en que se realiza el ingreso a la red. Hay que tener en cuenta que el nodo 1040 es de los primeros nodos en darse de alta dentro de la red, y que se encuentra a 2 saltos de la MBS. Debido a que es uno de los primeros nodos en ingresar a la red no sufre colisiones y por tanto su retardo es muy bajo.

5.3.3. XmtHoldoffExponent, XmtTimeMx

Como se explicó en el capítulo anterior hay dos parámetros que son necesarios para determinar el tiempo en que se transmitirá un mensaje MSH-NCFG de acuerdo al algoritmo *Mesh Election*, a continuación se muestran algunos resultados que se obtuvieron al cambiar estos valores.

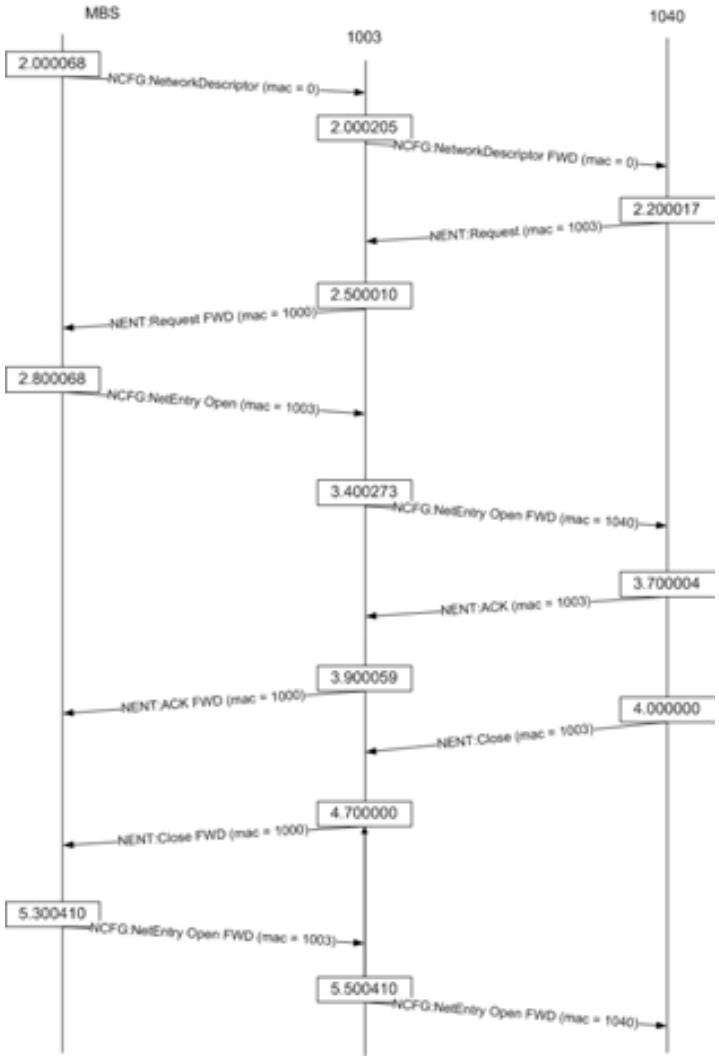


Figura 5.13: Ejemplo de ingreso a la red del nodo con identificador 1040 que se encuentra a 2 saltos de la MBS

En la figura 5.14 se muestra el resultado de variar el valor de *XmtHoldoffExponent* para el modelo teórico. Cuando se tiene un valor de 11 en este parámetro el tiempo crece abruptamente, debido a que después de transmitir una vez, el nodo tiene que esperarse el tiempo *XmtHoldoffTime* que como se explicó en el capítulo 4, depende directamente de este parámetro con la ecuación: $XmtHoldoffTime = 2^{XmtHoldoffExponent+4}$. Por la razón anterior se observa también un crecimiento elevado para los valores de 5 y 7. Para los tiempos de 0 a 4 se observa un crecimiento similar a las gráficas, esto se debe a que aunque haya diferencia de intervalos, el tiempo en que los nodos ganan para transmitir sus mensajes MSH-NCFG coincide aproximadamente con el tiempo que ya pasaron su *XmtHoldoffTime* De lo anterior podemos concluir que para nodos que trasmitan poco se puede asignar un *XmtHoldoffExponent* grande y viceversa para los nodos que transmiten gran cantidad de datos, un *XmtHoldoffExponent*

entre 1 y 4. Hay que tener en cuenta que lo ideal sería tener valores diferentes para cada nodo dependiendo de la frecuencia de uso que haga de la red.

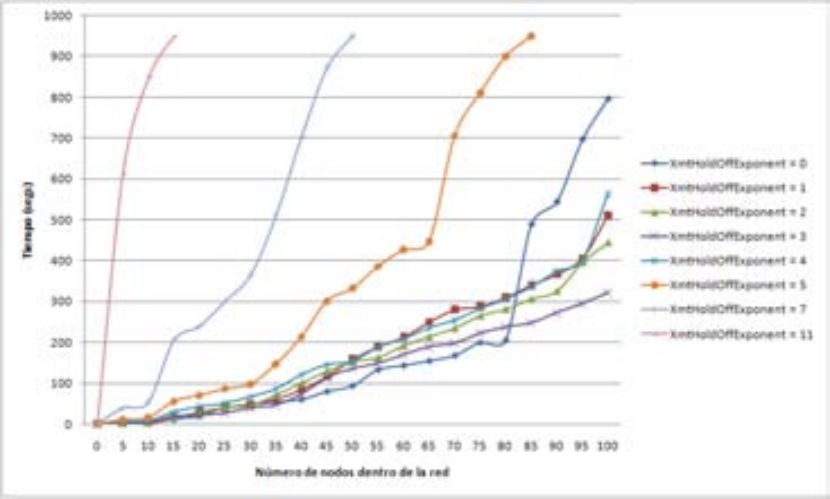


Figura 5.14: Tiempos de ingreso a la red con diferentes *XmtHoldoffExponent*, sin colisiones (programa C++)

Ahora en la figura 5.15 se muestra el resultado dinámico de variar el valor de *XmtHoldoffExponent*. Se puede observar que al tener en su mayoría de las veces un paquete MSH-NENT para ser contestado se pueden aprovechar mejor las oportunidades de transmisión en cada frame. Por otro lado se observa, similar al comportamiento de la simulación sin colisiones, que para los valores pequeños los tiempos son los mismos ya que caen dentro de los mismos rangos de competencia y envío. Como las colisiones retrasan el tiempo de envío, también se aprecia que para el caso del *XmtHoldoffExponent* = 7, no afecta tanto en el desempeño como en el modelo anterior, la razón es porque como se mencionó anteriormente generalmente se tiene un paquete NENT en espera cuando el nodo compite, entonces no desaprovecha el momento en que él resulta ganador. La tendencia de crecimiento es mayor para valores de exponentes más grandes, lo que representa tiempos más separados para transmitir, aunque a su vez un tiempo mayor en los que es permitido que el nodo compita.

Estos resultados se obtuvieron con *XmtTimeMx* oscilando entre los valores de 0 y 7 conforme los nodos iban ingresando a la red y que ya tenían condición para competir para el envío de mensajes MSH-NCFG.

Finalmente, en la figura 5.16 se muestran los resultados de variar los tiempos *XmtTimeMx*, con un valor de la variable *XmtHoldOffExponent* de cada nodo variando entre 1 y 3. Estos valores son los mejores exponentes para reducir el tiempo en el envío de los mensajes. Como se explicó en el capítulo 4 este parámetro indica la posición dentro del rango de tiempo determinado por el *XmtHoldoffExponent*, únicamente el tiempo se mueve dentro de este rango. Por tanto, los tiempos con diferentes valores de *XmtTimeMx* tienen la misma tendencia de crecimiento y sus periodos de transmisión oscilan dependiendo del tiempo *XmtTimeMx* que fue dado.

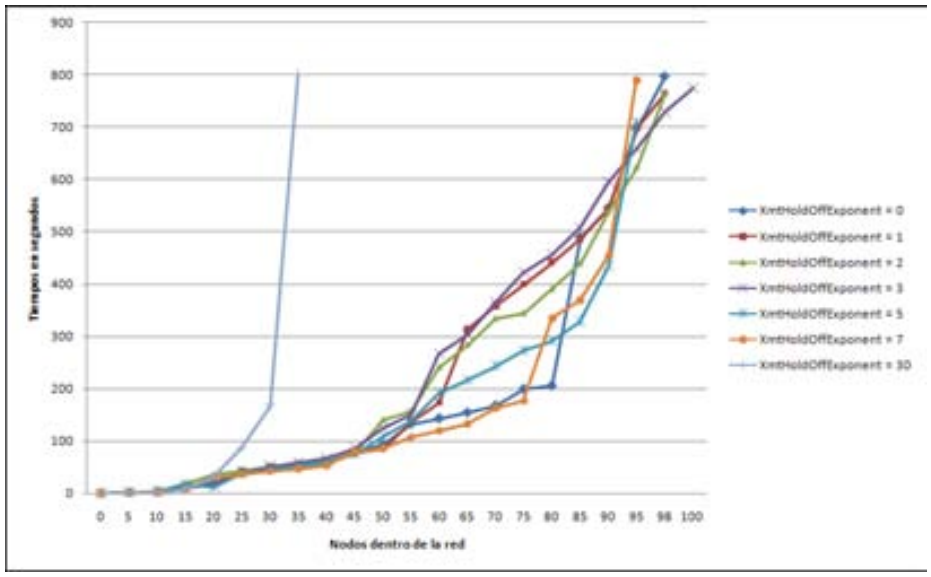


Figura 5.15: Tiempos ingresar a la red de 90 usuarios, variando *XmtHoldoffExponent*, en el modelo OPNET

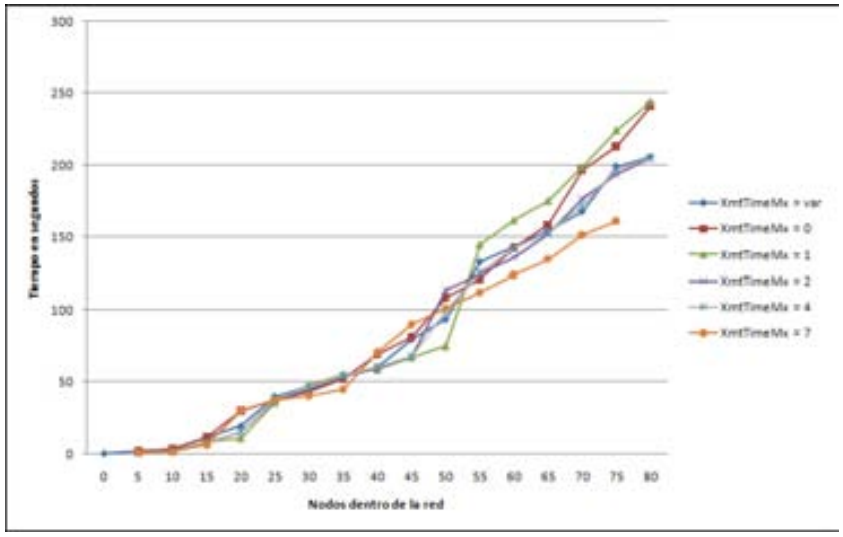


Figura 5.16: Tiempos ingresar a la red de 80 usuarios, variando *XmtTimeMx*, en el modelo OPNET

En el Apéndice D se muestra el código implementado para calcular el tiempo utilizando los algoritmos de selección de tiempos.

5.4. Conclusiones

En este capítulo se mostraron los principales resultados que se obtuvieron a partir del programa en C++ para la selección de oportunidades de transmisión y del modelo de simulación en OPNET. Estos resultados reflejan el comportamiento en la inicialización de una red Mesh y el impacto de los diferentes parámetros. Estas variables son tiempos de espera, y los parámetros para el algoritmo de selección de tiempo como son*XmtHoldoffExponent* y *XmtTimeMx*. Una de las principales contribuciones de esta tesis es el hecho de utilizar 7 oportunidades de transmisión para los mensajes MSH-NENT, con lo que se obtiene la posibilidad del control de un mayor número de nodos para su inicialización. Cabe mencionar que en los artículos referenciados en este trabajo generalmente manejan 10 nodos.

Capítulo 6

Conclusiones

Se ha creado un modelo que permite el estudio de la inicialización de nodos en una red basada en el protocolo IEEE 802.16 con arquitectura Mesh. Se ha implementado un algoritmo general para que el modelo funcione según lo descrito en el estándar IEEE 802.16-2004[2].

Parte importante del estudio dinámico de esta topología de red, es la relación del número de usuarios con la cantidad de tiempo que tardan en ingresar todos los nodos a la red.

Se analizaron los parámetros para dar a la inicialización de los nodos una mayor eficiencia y mejor uso de las oportunidades de transmisión. Se propusieron alternativas para optimizar el reparto de oportunidades de transmisión. Entre ellas, se ha visto que los tiempos de espera para los mensajes MSH-NENT y MSH-NCFG son útiles para el aprovechamiento de estas oportunidades y la implementación del algoritmo de selección.

Se detectaron valores para los parámetros como *XmtHoldOffExponent* para establecer un funcionamiento más eficiente con un valor de 3 en todos los nodos. Sin embargo, el valor de este parámetro da la noción de los tiempos de transmisión para que cuando se implemente la parte del envío de datos, se puede utilizar como un valor para la prioridad de transmisión de cada nodo.

Se observó el incremento en el tiempo de ingreso a la red de un grupo de 100 nodos, distribuidos en 1,2 y 3 saltos de distancia de la MBS.

Pero principalmente, se planteó y analizó la propuesta del uso efectivo del ancho de banda utilizando hasta 7 oportunidades de transmisión para los mensajes MSH-NENT y así obtener con tiempos más reducidos, mayor número de nodos dentro de la red.

6.1. Trabajos Futuros

Ya que este modelo es una base para la funcionalidad de lo que compone redes Mesh, hay una variedad extensa que puede ser realizada a partir de esta tesis, desde la funcionalidad descrita en el estándar hasta algunos temas ya propuestos de manera teórica en algunos

artículos, e incluso la mejora del modelo, utilizando transmisión paralela de mensajes en coberturas que no interfieran entre sí.

Los siguientes puntos podrán ser implementados:

1. Optimizar rutas por medio de algún algoritmo como OSPF y DSR.
2. Implementar la planeación para los tiempos de transmisión de datos, que puede ser centralizada o distribuida, e incluso las dos y por medio de una variable determinar cuál de las dos se utilizará en un momento dado.
3. Adaptar los procesos de *ranging* y de *grant* al uso de planificadores y de datos que funcionen análogamente como los sistemas PMP.
4. Incluso integrar seguridad a este tipo de topología.

Apéndice A

Glosario de acrónimos

ADSL Asymmetric Digital Subscriber Line

AUC Authentication Center

AP Access Point

BER Bit Error Rate

BRAN Broadband Radio Access Networks

BSC Base Station Controller

BTS Base Transceiver Station

BWA Broadband Wireless Access

CDMA Code Division Multiple Access

CRC Cyclic Redundancy Check

CSD Circuit-Switched Data

DSL Digital Subscriber Line

DOCSIS Data Over Cable Service Interface Specification

EDGE Enhanced Data Rates for GSM Evolution

EIR Equipment Identity Register

ETSI European Telecommunications Standards Institute

FCC Federal Communications Commission

FDD Frequency Division Duplex

FDMA Frequency Division Multiple Access

FFT Fast Fourier Transform

GPRS General Packet Radio Service

GPS Global Positioning Systems

GSM Global System for Mobile Communications

HIPERACCESS HIgh PErformance Radio Access

HIPERLAN HIgh PErformance Radio Local Area Network

HIPERMAN HIgh PErformance Radio Metropolitan Area Network

HLR Home Location Register

HSCSD High-speed Circuit-Switched Data

HSDPA High Speed Downlink Packet Access

HUMAN High-Speed Unlicensed Metropolitan Area Networks

IEEE Institute of Electrical and Electronics Engineers

IDU Indoor Unit

ITU International Telecommunications Union

IMT2000 International Mobile Telecommunication 2000

LMSD Local Multipoint Distribution Service

LOS Line Of Sight.

MAC Medium Access Control.

MAC PDU MAC Protocol Data Unit

MAC SDU MAC Service Data Unit

MBS Mesh Base Station

MBWA Mobile Broadband Wireless Access

MIH Media Independent Handover

MMDS Multichannel Multipoint Distribution Service

MS Mobile Station

MSC Mobile Services Switching Center

MSH-NCFG Mesh Network Configuration

MSH-NENT Mesh Network Entry

MSH-CSCH Mesh Centralized Scheduler

MSH-DSCH Mesh Distributed Scheduler

MSS Mesh Subscriber Station

NLOS Non-Line-of-Sight

NMT Nordic Mobile Telephone

NTT Nippon Telegraph and Telephone

OFDM Orthogonal Frequency-Division Multiplexing

PCMCIA Personal Computer Memory Card International Association

PCS Personal Communications Services

PDA Personal Digital Assistant

PLMN Public Land Mobile Network

PMP Point-to-Multipoint

PSTN Public Switched Telephone Network

QAM Quadrature Amplitude Modulation

QoS Quality of service

QPSK Quadrature Phase Shift Keying

RF Radio Frecuency

SMS Short Message Service

SOFDMA Scalable OFDMA

SOHO Small Office/Home Office

SS Subscriber Stations

TIA Telecommunications Industry Association

TDD Time Division Duplex

TDMA Time Division Multiple Access

TDM Time Division Multiplexing

TO Transmission Opportunity

TCP Transmission Control Protocol

UMTS Universal Mobile Telecommunications Service

USB Universal Serial Bus

VLR Visitor Location Register

WCDMA Wideband Code Division Multiple Access

WiFi Wireless Fidelity

WiMAX Worldwide Interoperability for Microwave Access

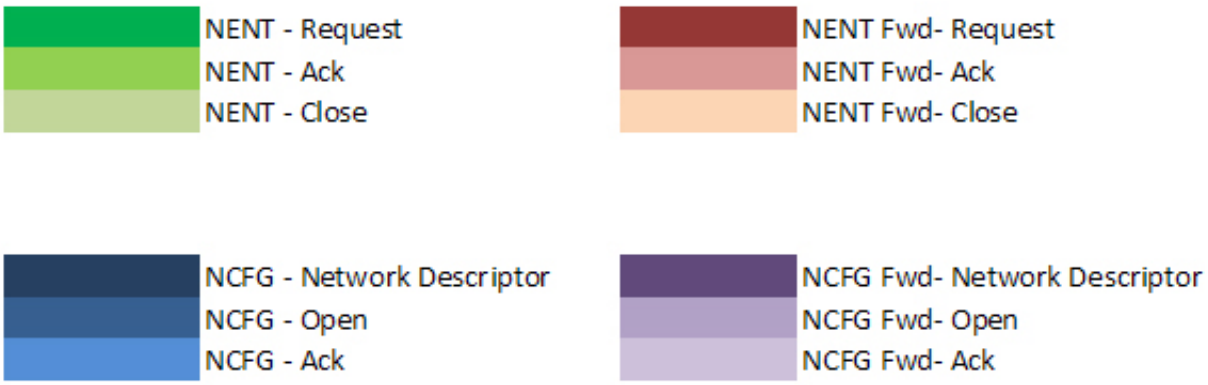
WLAN Wireless Local Area Network

WMAN Wireless Metropolitan Area Network

Apéndice B

Diagrama de frames con oportunidades de transmisión

En este diagrama de tiempo se muestra como se reparten los tiempos de oportunidades de transmisión para MSH-NCFG y MSH-NENT en sus respectivos pasos de ingreso a la red con $XmtHoldoffExponent = 0$ y un $XmtTime = 2$. Y los tiempos utilizados se muestran la existencia de colisiones y como se realiza los tiempos de espera. A continuación se muestran los colores que se han utilizado en los diagrama y su equivalencia ya sea en mensajes MSH-NCFG ó MSH-NENT.



Time	Oportunidades de transmisión NENT							Oportunidades de transmisión NCFG						
s(µs)	0(0)	1(09)	2(19)	3(29)	4(39)	5(48)	6(58)	0(68)	1(137)	2(205)	3(273)	4(341)	5(410)	6(478)
0.1														
0.2	7|10	8	3|1	5	4|2|9	6			8	5	6			
0.3														
0.4	8		6			5								
0.5	8	6			5				8		6			
0.6								5						
0.7	7	1	10	9		2	3							
0.8			4							7				1
0.9														10
1	7		1									9	2	
1.1	7	1					10		3			4		
1.2	2	10	9		3			7			1	10		
1.3	9		3	2		4								
1.4	4													
1.5											9			
1.6														
1.7					2						2			
1.8			3										3	
1.9			9											
2		4						1000		3			2	
2.1								9		7			5	
2.2			40|33	26|37	29	15|22|18	17|32				4		1	
2.3	28|16|25	21|39	31|	19|	35|27|12	14	36|23|24				6			
2.4	11	20|13|30											10	
2.5	3,40	2,33	34		38	9,14								
2.6						7,12		8						
2.7	5,11													
2.8	6,38													
2.9														
3				6,34										
3.1														
3.2		2,29			9,14	3,39								
3.3			7,36											
3.4														
3.5	6,38													
3.6											9->14			
3.7														
3.8														
3.9					9,27	2,32								
4														
4.1														
4.2														3->40
4.3														
4.4								2->33						
4.5							2,21	3->40	6->34	2->33				
4.6														6->38
4.7					40		33	9->19						
4.8				40		33		9->19						
4.9														
5					3,39	19								
5.1			2,33		19			7->12			5->11			
5.2												5->11		
5.3							9,19							
5.4														
5.5	11							7->12						
5.6	11													2->29
5.7					3,40	12	5,11		9->27		9->27			
5.8					12	5,11		3->39		2->29		3->39		
5.9						2,33								

Time	Oportunidades de transmisión NENT							Oportunidades de transmisión NCFG						
s(µs)	0(0)	1(09)	2(19)	3(29)	4(39)	5(48)	6(58)	0(68)	1(137)	2(205)	3(273)	4(341)	5(410)	6(478)
6			27											
6.1		39	27	29	7,12						7->36		6->34	
6.2		29												
6.3				34										
6.4			34			2,29	3,39	6->38		2->32			2->32	
6.5				36										
6.6	5,11	36					6,34							
6.7	32				38					2->21			2->21	
6.8	32			38										
6.9	7,12													
7	9,27	21	3,39											
7.1	21													
7.2	7,36					2,32								
7.3														
7.4	6,38		2,32								5->11			
7.5						9,27								3->40
7.6						2,21					7->12	3->40		
7.7			7,36											
7.8												2->33		2->33
7.9								9->19						9->19
8		6,38												
8.1					2,21					2->29				
8.2														
8.3												3->39		
8.4														
8.5										6->34				5->11
8.6														
8.7										5->11				
8.8										7->36				
8.9														
9								9->32						
9.1														
9.2														
9.3														
9.4														
9.5														
9.6														
9.7														
9.8														
9.9														
10														
10.1												9->32		
10.2														
10.3														
10.4												6->38		7->12
10.5														
10.6														
10.7														
10.8												2->29		
10.9														2->29
11														
11.1														
11.2												3->39		
11.3										2->21				3->39
11.4														
11.5														
11.6														
11.7														
11.8								6->34						

Time	Oportunidades de transmisión NENT							Oportunidades de transmisión NCFG						
s(µs)	0(0)	1(09)	2(19)	3(29)	4(39)	5(48)	6(58)	0(68)	1(137)	2(205)	3(273)	4(341)	5(410)	6(478)
11.9														
12														
12.1														
12.2														
12.3												6->34		
12.4														
12.5														
12.6														
12.7														
12.8							13							
12.9														
13														
13.1					5,13			2->32						
13.2														
13.3														
13.4												2->32		
13.5														
13.6														
13.7	14											9->27		
13.8														9->27
13.9														
14														
14.1														
14.2														
14.3	15													
14.4					3,15									
14.5														
14.6														
14.7														
14.8														
14.9														
15														
15.1			16											
15.2					2,16							7->36		
15.3														
15.4								7->36						
15.5								6->38				2->21		
15.6														
15.7														
15.8														
15.9		17												
16				3,17				1000						
16.1								6	2			2->16		6->34
16.2														
16.3			80	82|65	56	71	52	2->32		1		3		4
16.4	99	92|42	68|79		87	34,80	49|59			5->13				
16.5	89	97|48					32,87	5->13		7				
16.6						6,34,80								7->36
16.7	13	18	76	94	58|60|45	93|86	63|57|74|43							8
16.8	13	95		5,13			2,32,87	10		7->12				
16.9						83	36,76			9				
17														
17.1			36,95					9->19		2->29				
17.2				73	88	78|41|61	32,89	9->27				3->15		
17.3	44		70											
17.4	98	19,73						3->40		2->33				
17.5		29,70		47	46|100	90|57|62	50|57|62|91	3->39						
17.6	81|67					69	34,52							
17.7			96	2,32,89	40,47									

Time	Oportunidades de transmisión NENT							Oportunidades de transmisión NCFG						
s(µs)	0(0)	1(09)	2(19)	3(29)	4(39)	5(48)	6(58)	0(68)	1(137)	2(205)	3(273)	4(341)	5(410)	6(478)
17.8						39,69								
17.9							15	2->16		2->16				
18						15								
18.1	19,41	2,29,70	16											
18.2	16	3,39,69					2,16							
18.3														
18.4				34,71							3->17			
18.5			20	39,96										
18.6			6,34,71			3,15								
18.7	5,20							3->17						
18.8														
18.9			17											
19	3,17	17	34,99											
19.1	3,15													
19.2	21		6,34,99											
19.3													2->32->87	
19.4			3,39,96									6->34->80		
19.5														
19.6										6->34->80				
19.7														
19.8														
19.9	3,17					22								

Apéndice C

Porciones de código del modelo en OPNET

En esta sección se muestran los segmentos más significativos de los códigos programados para la estación base y la estación suscriptora. (Sólo se conservó una mínima parte del código generado por la aplicación OPNET.

C.1. Estación base Mesh

```
/* Process model C form file: mesh_bs.pr.c */

/* Header Block */

/***** Includes *****/

 #include <opnet.h>
 #include "mesh_support.h"
 #include "mesh_gral.h"
 #include  <stdlib.h>
 #include  <stdio.h>

/***** State transition macros *****/

 #define NO_INTRPT_CODE          -1
 #define NCFG_CODE               10
 #define NCFG_CODE_BRD           15
 #define NENT_CODE               20
 #define SQUEUE_NENT_CODE        25
 #define WAIT_TX_CODE            28
 #define NO_STREAM               -1


/* End of Header Block */

/* Function Block */

enum { _block_origin = __LINE__ };

 static void get_network_parameters(){


    FIN (get_network_paratmeters ());


     if (op_prg_odb_ltrace_active("func_track"))
    {
        printf("\\t\\t\\t FUNCTION - cm_get_sim_attributes \\n");
    }
    op_ima_sim_attr_get(OPC_IMA_INTEGER, "Xmt_Holdoff_exp", &holdoff_exp);
    op_ima_sim_attr_get(OPC_IMA_INTEGER, "Symbols_per_Slot", &symbol_x_slot);
    op_ima_sim_attr_get(OPC_IMA_INTEGER, "Bytes_per_Symbol", &bytes_per_slot);
    op_ima_sim_attr_get(OPC_IMA_INTEGER, "Xmt_Power", &xmt_power);
    op_ima_sim_attr_get(OPC_IMA_INTEGER, "Xmt_Antenna", &xmt_ant);
    op_ima_sim_attr_get(OPC_IMA_INTEGER, "Frame_Duration_code", &frame_duration_code);
```

```
    op_ima_sim_attr_get(OPC_IMA_INTEGER, "MSH_CTRL_LEN", &ctrl_len);
    op_ima_sim_attr_get(OPC_IMA_INTEGER, "Network_base_channel", &net_base_ch);

    FOUT;
}


static void init_gral_list(){

    NEIGHBOR_INIT_STRUCT* neighInit;

    int listSize = 0;
    FIN (init_parameter_reg());

        printf("\\t\\t\\t FUNCTION - init_parameter_reg   \\n");
        neighInit = (NEIGHBOR_INIT_STRUCT*) op_prg_mem_alloc (sizeof (NEIGHBOR_INIT_STRUCT));
        neighInit->xmt_holdoff_exp = 0;
        neighInit->next_tx_exp = 2;
        neighInit->nodeID = 1000;
        op_prg_list_insert(nodeGralList,neighInit,OPC_LISTPOS_TAIL);
        op_prg_list_insert(competitiveList,neighInit,OPC_LISTPOS_TAIL);

    listSize = op_prg_list_size(init_neigh);
    FOUT;
}

static Packet* encap_pkt (int type, int dest, int address, Packet*
emb_data_pkt, int no_nbr, int mac_entry) {

    int total_size;
    int msg_size;
    int hdr_size;

    Packet* msh_msg;
    Packet* msh_bs_pkt;
    Packet* mac_frame;
    Packet* hdr_ptr;
    int indice = 0;

    NEIGHBOR_DESCRIP_STRUCT** neighs_desc;
    NEIGHBOR_DESCRIP_STRUCT* neigh_desc;


    FIN (encap_pkt ());

    neigh_data = (NEIGHBOR_DATA_STRUCT*) op_prg_mem_alloc (no_nbr*sizeof (NEIGHBOR_DATA_STRUCT));


    if (op_prg_odb_ltrace_active("debug_info"))
        printf("\\t\\t\\t FUNCTION - encap_pkt \\n");

    msh_bs_pkt = op_pk_create_fmt (MSH_NCFG_BS_DESCRIP_PK_FMT);

    if(type == NCFG_NETWORK_DESCRIPTOR){
        op_pk_nfd_set (msh_bs_pkt, "BS Node", 1);
        op_pk_nfd_set (msh_bs_pkt, "Number of hops", 1);
        op_pk_nfd_set (msh_bs_pkt, "Xmt energy/bit", 1);

        neigh_data->num_nodes = no_nbr;

    /* Create MAC management packet. */
    msh_msg = op_pk_create_fmt (MSH_NCFG_PK_FMT);

   /* Set the destination address of the packet. */
    op_pk_nfd_set (msh_msg, "Management Message Type", type);
    op_pk_nfd_set (msh_msg, "Nbr Entries", ncfg_ptr->no_nbr_ent);
    op_pk_nfd_set (msh_msg, "BS Entries",  ncfg_ptr->no_bs_ent);
    op_pk_nfd_set (msh_msg, "Embedded Packet Flag", ncfg_ptr->emb_pkt_flag);
    op_pk_nfd_set (msh_msg, "Xmt Power", ncfg_ptr->xmt_pwr);
    op_pk_nfd_set (msh_msg, "Xmt Antenna", ncfg_ptr->xmt_antena);
    op_pk_nfd_set (msh_msg, "NetEntry MAC Address Flag", 1);
    op_pk_nfd_set (msh_msg, "NetConfig Count", ncfg_ptr->net_ent_mac_addr);
    op_pk_nfd_set (msh_msg, "Timestamp Frame Number", 0);
    op_pk_nfd_set (msh_msg, "Timestamp Net CSlot", 0);
    op_pk_nfd_set (msh_msg, "Timestamp Sync Hop Count", ncfg_ptr->net_cfg_cnt);
    op_pk_nfd_set (msh_msg, "Xmt Holdoff Exponent", 3);
    op_pk_nfd_set (msh_msg, "Next Xmt Mx", 2);

    if(type == NCFG_NETWORK_DESCRIPTOR){
        op_pk_nfd_set (msh_msg, "NetEntry MAC Address Flag", 0);
    }else{
        op_pk_nfd_set (msh_msg, "NetEntry MAC Address Flag", 1);
        op_pk_nfd_set (msh_msg, "NetEntry MAC Address", mac_address);
    }
```

```
    op_pk_nfd_set (msh_msg, "BS Descriptor", msh_bs_pkt);
    op_pk_nfd_set (msh_msg, "Node Nbr Descriptor",neigh_data,op_prg_mem_copy_create, op_prg_mem_free, sizeof (NEIGHBOR_DATA_STRUCT));
    op_pk_nfd_set (msh_msg, "Embedded Data", emb_data_pkt);

    /* Create MAC management packet. */
    mac_frame = op_pk_create_fmt (MSH_PK_FMT);

    /* Get the total size of the payload in bits. */
    msg_size = op_pk_total_size_get (msh_msg);


    if (op_prg_odb_ltrace_active("debug_info"))
    {
    printf("\\t Mesh sending packet  of type %s, size %d at %lf\\n",
           MSH_PK_FMT, (int)op_pk_total_size_get(mac_frame), time_now);
    }

    /* Create MAC frame and encapsulate header and management packet. */
    hdr_ptr = op_pk_create_fmt (MSH_HD_PK_FMT);
    /* Set the destination address of the packet. */
    op_pk_nfd_set (hdr_ptr, "len", 10);
    op_pk_nfd_set (hdr_ptr, "MAC Source", mac_bs);
    op_pk_nfd_set (hdr_ptr, "MAC Destination", dest);

    /* Get the header size is bits. */
    hdr_size = op_pk_total_size_get (hdr_ptr);


    op_pk_nfd_set(mac_frame, "Header", hdr_ptr);
    op_pk_nfd_set(mac_frame, "Payload", msh_msg);

    total_size = msg_size + MGMT_HEADER_SIZE + hdr_size;
    FRET (mac_frame);
    }



static Packet* encap_emb_data(int type,int dest, int address, int
mac_entry) {
        Packet* emb_data_pkt;
        Packet* send_pkt;
        Packet* emb_data_IE_pkt;
        Packet* emb_channel_IE_pkt;
        Packet* emb_burst_profile_pkt;

        FIN (encap_emb_data ());

        switch (type)
            {
            case NCFG_NETWORK_DESCRIPTOR:
                mac_address = 0;

                emb_channel_IE_pkt = op_pk_create_fmt (MSH_NCFG_CHANNEL_PK_FMT);
                op_pk_nfd_set (emb_channel_IE_pkt, "Physical Channel center frecuency", 0);
                op_pk_nfd_set (emb_channel_IE_pkt, "Physical Channel width", 0);
                emb_burst_profile_pkt = op_pk_create_fmt (MSH_NCFG_BURST_PROFILES_PK_FMT);
                op_pk_nfd_set (emb_burst_profile_pkt, "FEC Code Type", 0);
                op_pk_nfd_set (emb_burst_profile_pkt, "Mandatory Exit Threshold", 0);
                op_pk_nfd_set (emb_burst_profile_pkt, "Mandatory Entry Threshold", 0);
                emb_data_IE_pkt = op_pk_create_fmt (MSH_NCFG_NET_DESCRIP_PK_FMT);
                op_pk_nfd_set (emb_data_IE_pkt, "Frame Length Code", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "MSH-CTRL-LEN", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "MSH_DSCH_NUM", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "MSH-CSCH-DATA-FRACTION", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "Scheduling Frames", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "Num_Burst_Profiles", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "Operator_ID", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "XmtEnergyUnitsExponent", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "Channels", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "MinCSFForwardibgDelay", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "ExtendedNeighborhoodType", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "Channel_IE", emb_channel_IE_pkt);
                op_pk_nfd_set (emb_data_IE_pkt, "Burst_Profiles", emb_burst_profile_pkt);
                break;

            case NCFG_NETWORK_ENT_OPEN:
                emb_data_IE_pkt = op_pk_create_fmt (MSH_NCFG_NET_ENTRY_OPEN_PK_FMT);
                op_pk_nfd_set (emb_data_IE_pkt, "Minislot Start", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "Minislot Range", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "Frame number", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "Channel", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "Schedule validity", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "Channel rcv", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "Estimated Propagation Delay", 0);
                break;
```

```
            case NCFG_NETWORK_ENT_REJECT:
                emb_data_IE_pkt = op_pk_create_fmt (MSH_NCFG_NET_ENTRY_REJECT_PK_FMT);
                op_pk_nfd_set (emb_data_IE_pkt, "Rejection Code", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "Rejection Reason", 0);
                printf("NCFG_NETWORK_ENT_REJECT\\n");
                break;

            case NCFG_NETWORK_ENT_ACK:
                emb_data_IE_pkt = op_pk_create_fmt (MSH_NCFG_NET_ENTRY_ACK_PK_FMT);
                update_neigh = 1;
                break;

            case NCFG_NEIGHBOR_LINK_EST:
                emb_data_IE_pkt = op_pk_create_fmt (MSH_NCFG_NEIGH_LINK_PRO_PK_FMT);
                op_pk_nfd_set (emb_data_IE_pkt, "Action Code", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "Nbr Authentication value", 0);
                op_pk_nfd_set (emb_data_IE_pkt, "Link ID", 0);
                break;

            default:
                break;
            }

        emb_data_pkt = op_pk_create_fmt (MSH_NCFG_EMB_DATA_PK_FMT);
        op_pk_nfd_set (emb_data_pkt, "Type", type);
        op_pk_nfd_set (emb_data_pkt, "Length", 10);
        op_pk_nfd_set (emb_data_pkt, "Embedded_data_IE", emb_data_IE_pkt);

        send_pkt = encap_pkt(type, dest, address,emb_data_pkt,15, mac_entry);

        FRET (send_pkt);


    }


static void store_sids(Packet *pkPtr)
    {
    int mac_addr;

    sidEntry *masterEntry;


    FIN(store_sids());
    if (op_prg_odb_ltrace_active("func_track"))
        printf("\\t\\t\\t FUNCTION - store_sids   \\n");

    op_pk_nfd_get(pkPtr,"MAC Address",&mac_addr);



    printf(" Receives Registration Request for SID %d\\n",mac_addr);

    /* Make an entry in the master list. */
    masterEntry = (sidEntry *)op_prg_mem_alloc(sizeof(sidEntry));
    masterEntry->SID = mac_addr;
    masterEntry->macID = 0;
    masterEntry->sponsorID = 0;
    masterEntry->link = 1;
    masterEntry->xmt_holdoff_exp = 0;
    masterEntry->next_tx_exp = 0;

    op_prg_list_insert(masterList,masterEntry,OPC_LISTPOS_TAIL);
    FOUT;
}

static int get_state_node(int mac){

    sidEntry *masterEntry;

    int state_node = 0;
    int i = 0;
    int listSize;

    FIN(get_state_node());

        listSize = op_prg_list_size(masterList);

        printf("\\n\\t\\t ******************** get state Master List with %d users\\n",  listSize);
        for(i = 0;i < listSize;i++)
            {
                masterEntry = (sidEntry *) op_prg_list_access(masterList,i);
                if(masterEntry->SID == mac){
                     state_node = masterEntry->link;
                    break;
                }
```

```
        }

    FRET(state_node);

}

static void update_state_node(int mac, int type){

    sidEntry *masterEntry;

    int state_node = 0;
    int i = 0;
    int listSize;

    FIN(get_state_node());

        listSize = op_prg_list_size(masterList);

        printf("\\n\\t\\t ********************** update state Master List with %d users\\n",  listSize);
        for(i = 0;i < listSize;i++)
            {
                masterEntry = (sidEntry *) op_prg_list_access(masterList,i);
                if(masterEntry->SID == mac){
                    masterEntry->link=type;
                    break;
                }
        }
    FOUT;

}

//DA de alta el en la lista de vecinos al nodo entrante
static void store_neigh(Packet *pkPtr)
    {
    int mac_addr;

    NEIGHBOR_DESCRIP_STRUCT *neighEntry;

    FIN(store_sids());
    //SE incrementa el número de vecinos
     neigh_data->num_nodes++;
    if (op_prg_odb_ltrace_active("func_track"))
        printf("\\t\\t\\t FUNCTION - store_sids   \\n");

    op_pk_nfd_get(pkPtr,"MAC Address",&mac_addr);

    printf(" Receives Registration Request for SID %d\\n",mac_addr);

    /* Make an entry in the master list. */
    neighEntry = (NEIGHBOR_DESCRIP_STRUCT *)op_prg_mem_alloc(sizeof(NEIGHBOR_DESCRIP_STRUCT));
    neighEntry->macID = 0;
    neighEntry->sponsorID = 0;
    neighEntry->link = 0;
    neighEntry->xmt_holdoff_exp = 0;
    neighEntry->next_tx_exp = 0;

    op_prg_list_insert(neighborList,neighEntry,OPC_LISTPOS_TAIL);
    FOUT;
}


static void print_gral_list(void) {
    int i, listSize;
    NEIGHBOR_INIT_STRUCT *listEntry;

    FIN(print_master_list());

        if (op_prg_odb_ltrace_active("func_track"))
            printf("\\t\\t\\t FUNCTION - print_master_list   \\n");
        listSize = op_prg_list_size(nodeGralList);
        printf("\\n\\t\\t **********************General List with % users %d\\n",  listSize);
        for(i = 0;i < listSize;i++){
            listEntry = (sidEntry *) op_prg_list_access(nodeGralList,i);
            printf("\\t\\tSID %d", listEntry->nodeID);
            }
    printf("\\n");
    FOUT;

    }


static int in_network(int mac){

    sidEntry *childEntry;

    int is_in_net = 0;
```

```
    int i = 0;
    int listSize;
    FIN(in_network());

        listSize = op_prg_list_size(masterList);

        for(i = 0;i < listSize;i++)
            {
                childEntry = (sidEntry *) op_prg_list_access(masterList,i);
                if(childEntry->SID == mac){
                    is_in_net = 1;
                    break;
                }

        }
    printf("\\n");

    FRET(is_in_net);
}

static Packet* encap_nent_queue(Packet *msg_ptr, Packet* hdr_ptr,
Packet* nent_req,int mac, int type){

    Packet* pkt_send;

    FIN(encap_nent_queue());

        op_pk_nfd_set(nent_req, "MAC Address", mac);
        op_pk_nfd_set(msg_ptr, "Type", type);
        op_pk_nfd_set(msg_ptr, "NENT_Request", nent_req);
        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", msg_ptr);

    FRET(pkt_send);
}

/* End of Function Block */


/* Process model interrupt handling procedure */

void mesh_BS (void)
    {
    int _block_origin = 0;
    FIN (mesh_BS ());
    if (1)
        {
        int intrpt_type;
        int intrpt_stream;
        int intrpt_code;
        int nent_arrival;
        int ctrl_arrival;
        int data_arrival;
        int wait_tx;
        int nent_squeue;
        int type_ncfg;
        int subq_index;
        int win_NCFG;

        int mac_source;
        int mac_entry;

        int type;
        int mac;
        int state_node;
        int temp;

        Packet* msg_ptr;
        Packet* hdr_ptr;
        Packet* nent_arrived;
        Packet* nent_request_ptr;

        char pk_format[32];
        char hd_format[32];

        double          time_tx_NCFG;

        unsigned long       time_mx;

        int         listSize;

        NEIGHBOR_INIT_STRUCT* dataNeighNCFG;
        FSM_ENTER (mesh_BS)

        FSM_BLOCK_SWITCH
```

```
{
/*---------------------------------------------------------*/
/** state (INIT) enter executives **/
FSM_STATE_ENTER_FORCED_NOLABEL (0, "INIT", "mesh_BS () [INIT enter execs]")
    {
    intrpt_type = op_intrpt_type();

    if (op_prg_odb_ltrace_active("state"))
        printf("BS: Iniciando...");

    masterList = op_prg_list_create();
    init_neigh = op_prg_list_create();
    nodeGralList = op_prg_list_create();
    competitiveList = op_prg_list_create();

    //Tablas de inicializacion para calcular tiempo NCFG

    get_station_attributes();
    init_parameter_reg(1);
    init_gral_list();

    if (intrpt_type != OPC_INTRPT_BEGSIM)
        {
        printf("\\n\\t FATAL ERROR -\\n"
                "  \\t Initial interrupt was type %d \\n"
                "  \\t Should have been BEGIN SIM (%d) \\n",
                intrpt_type, OPC_INTRPT_BEGSIM);

        op_sim_end("","","","");
        }

    if (op_prg_odb_ltrace_active("debug_info"))
        printf("BS: Obteniendo parametros de red..\\n");
    get_network_parameters();

    if (op_prg_odb_ltrace_active("debug_info"))
        printf("BS: Network frame duration: %d\\n",network_general_ptr->frame_duration);

    ncfg_ptr = (NCFG_STRUCT*) op_prg_mem_alloc (sizeof (NCFG_STRUCT));
    ncfg_ptr = mesh_support_get_ncfg();

    if (op_prg_odb_ltrace_active("state")){
        printf("BS: CONTENIDO / INIT de la estructura %d\\n",ncfg_ptr->msg_type);
        printf("BS: Tiempo actual: %f ",op_sim_time());
        }

    time_now = op_sim_time();
    mac_address = 0;
    update_neigh = 0;
    count_nent = 0;
    ncfg_net_descrip = 1;

    if (op_prg_odb_ltrace_active("time_dg"))
    {
       printf("+++++++++Inicia multiframe++++++++++++++");
        }
    op_intrpt_schedule_self (0.00006836, NCFG_CODE);
    }

/** state (INIT) transition processing **/
FSM_TRANSIT_FORCE (1, state1_enter_exec, ;, "default", "", "INIT", "IDLE")
    /*---------------------------------------------------------*/
/** state (IDLE) enter executives **/

/** state (IDLE) exit executives **/
FSM_STATE_EXIT_UNFORCED (1, "IDLE", "mesh_BS () [IDLE exit execs]")
    {
    if (op_prg_odb_ltrace_active("state"))
        printf("BS: IDLE EXIT\\n");


    nent_arrival = 0;
    ctrl_arrival = 0;

    /*************************************************************
    * Make sure interrupt was STRM
    *************************************************************/

    intrpt_type = op_intrpt_type();

    switch (intrpt_type)
        {
        case OPC_INTRPT_STRM:
            /*******************************************************
            * Get frame from stream
            *******************************************************/
```

```
            intrpt_stream = op_intrpt_strm();
            intrpt_code = op_intrpt_code();

            nent_arrived = op_pk_get (intrpt_stream);
            op_pk_format(nent_arrived,pk_format);
            if (op_prg_odb_ltrace_active("pkt_msg")) {
                printf("BS: Imprimiendo pakt q llega:\\n");
                op_pk_print(nent_arrived);
            }
            ctrl_arrival = 1;
            if (op_prg_odb_ltrace_active("debug_info"))
            {
                printf("\\n\\t Base Station - STREAM -\\n"
                "  \\t Idle state interrupt was type. %s \\n",
                pk_format);
                }
            break;

        case OPC_INTRPT_SELF:
            intrpt_code = op_intrpt_code();
            if(intrpt_code == NCFG_CODE ){
                ncfg_send = 1;
                wait_ncfg = 0;
            }else if (intrpt_code == NCFG_CODE_BRD){
                /* Para reiniciar variables y enviar un NCFG network descriptor */
                ncfg_send = 1;
                intrpt_code = NCFG_CODE;
                type_nent = 0;

            }else if (intrpt_code == WAIT_TX_CODE){
                wait_tx = 1;

            }
            if (op_prg_odb_ltrace_active("debug_info")) {
                printf("\\n\\t Base Station - SELF -\\n"
                "  \\t Idle state interrupt was type. %d \\n", intrpt_code);
                }
            break;

        case OPC_INTRPT_STAT:
            intrpt_code = op_intrpt_code();
            if (op_prg_odb_ltrace_active("debug_info"))
                {
                printf("\\n\\t Base Station - STAT -\\n"
                "  \\t Idle state interrupt was type. %d \\n",
                intrpt_code);
                }
            break;

        case OPC_INTRPT_REMOTE:
            intrpt_code = op_intrpt_code();
            if (op_prg_odb_ltrace_active("debug_info"))
            {
                printf("\\n\\t Base Station - REMOTE -\\n"
                "  \\t Idle state interrupt was type ignored. %d \\n",
                intrpt_code);
                }
            break;
        default:

            printf("\\n\\t Base Station - FATAL ERROR -\\n"
                "  \\t Idle state interrupt was type. %d , %d, %d , %d , %d, %d , %d\\n",
                intrpt_type, OPC_INTRPT_STAT,OPC_INTRPT_BEGSIM ,OPC_INTRPT_REMOTE, OPC_INTRPT_ENDSIM,OPC_INTRPT_PROCESS ,OPC_INTRPT_MCAST);
            break;

        }

    if (op_prg_odb_ltrace_active("pkt_msg")){
        printf("NCFG_SEND :%d\\n", NCFG_SEND);
        printf("CTRL_ARRIVAL :%d\\n", CTRL_ARRIVAL);
        printf("tout_NCFG :%d\\n", tout_NCFG);
    }
   }

/** state (IDLE) transition processing **/
FSM_INIT_COND (NCFG_SEND)
FSM_TEST_COND (CTRL_ARRIVAL)
FSM_TEST_COND (tout_NCFG)
FSM_DFLT_COND
FSM_TEST_LOGIC ("IDLE")

FSM_TRANSIT_SWITCH
    {
    FSM_CASE_TRANSIT (0, 2, state2_enter_exec, ;, "NCFG_SEND", "", "IDLE", "NCFG_send")
    FSM_CASE_TRANSIT (1, 4, state4_enter_exec, ;, "CTRL_ARRIVAL", "", "IDLE", "CTRL_arrival")
    FSM_CASE_TRANSIT (2, 6, state6_enter_exec, ;, "tout_NCFG", "", "IDLE", "SELECT_TO")
```

```
    FSM_CASE_TRANSIT (3, 1, state1_enter_exec, ;, "default", "", "IDLE", "IDLE")
    }
    /*---------------------------------------------------------*/

/** state (NCFG_send) enter executives **/
FSM_STATE_ENTER_FORCED (2, state2_enter_exec, "NCFG_send", "mesh_BS () [NCFG_send enter execs]")
    {
    if (op_prg_odb_ltrace_active("state"))
        printf("BS at %lf: NCFG/Enter\\n",op_sim_time());

    ncfg_send = 1;
    print_gral_list();

    switch (type_nent)
        {
           case NET_ENTRY_REQ:
                type_ncfg = NCFG_NETWORK_ENT_OPEN;
                if (op_prg_odb_ltrace_active("pkt_msg"))
                    printf("\\n----BS: Recibi NET_ENTRY_REQ, envío NCFG_NETWORK_ENT_OPEN \\n");
                break;

           case NET_ENTRY_ACK: //y  NET_ENTRY_CLOSE:
                //type_ncfg = NCFG_NETWORK_ENT_ACK;
                //neigh_data->num_nodes=no_nbr+1;
                 //ingresa_neighbor();

                if (op_prg_odb_ltrace_active("pkt_msg"))
                    printf("\\n----BS: Recibi NET_ENTRY_ACK espero NENT_ENTRY_CLOSE \\n");
                break;
           case NET_ENTRY_CLOSE: //y  NET_ENTRY_CLOSE:
                type_ncfg = NCFG_NETWORK_ENT_ACK;
                insert_node(mac_address);

                if (op_prg_odb_ltrace_active("pkt_msg"))
                    printf("\\n----BS: Recibi NET_ENTRY_CLOSE envío NCFG_NETWORK_ENT_ACK \\n");
                break;

           default:

                type_ncfg = NCFG_NETWORK_DESCRIPTOR;
                mac_address = 0;
                mac_dest = 0;
                if (op_prg_odb_ltrace_active("pkt_msg")){
                    printf("\\n----BS:No recibi nada envio NCFG INIcial, envío NCFG_NETWORK_DESCRIPTOR\\n");
                    printf("----Valor type_nent: %d\\n",type_nent);
                    }
                break;

        }
    if(type_ncfg == NCFG_NETWORK_ENT_ACK)
         update_neigh = 1;
    if(type_nent != NET_ENTRY_ACK){
        Msh_NCFG_ptr = encap_emb_data(type_ncfg, mac_dest, mac_address, mac_address);
        op_pk_send (Msh_NCFG_ptr,0);
    }

    if (op_prg_odb_ltrace_active("time_dg"))
            {
        printf("BS NEW NCFG type %d with dest: %d at %f \\n" , type_ncfg, mac_address, op_sim_time());
            }

    if (type_ncfg == NCFG_NETWORK_DESCRIPTOR){
        if(ncfg_net_descrip < 12){
            temp = ncfg_net_descrip*ncfg_net_descrip + (ncfg_net_descrip*5.000);
            if(ncfg_net_descrip == 1) temp = 2.00;
            op_intrpt_schedule_self (op_sim_time()   + temp , NCFG_CODE_BRD);
            ncfg_net_descrip++;
      }

    }
    }

/** state (NCFG_send) transition processing **/
FSM_TRANSIT_FORCE (1, state1_enter_exec, ;, "default", "", "NCFG_send", "IDLE")
    /*---------------------------------------------------------*/

/** state (NENT_arrival) enter executives **/
FSM_STATE_ENTER_FORCED (3, state3_enter_exec, "NENT_arrival", "mesh_BS () [NENT_arrival enter execs]")
    {
    if (op_prg_odb_ltrace_active("state"))
        printf ("\\nBS: NENT/Enter:\\n ");

    op_pk_nfd_get(msg_ptr, "NENT_Request", &nent_request_ptr);
    op_pk_nfd_get(nent_request_ptr, "MAC Address",&mac );

    op_pk_nfd_get(msg_ptr, "Type",&type );
```

```
    state_node = get_state_node(mac);

    if(state_node < type){
        if (op_prg_odb_ltrace_active("time_dg"))
                printf("BS, Rcv MSH-NENT type:  from mac_address to squeue %d  \\n", type_nent,  mac);
        nent_arrived = encap_nent_queue(msg_ptr, hdr_ptr, nent_request_ptr, mac, type);
        op_subq_pk_insert (0, nent_arrived, OPC_QPOS_TAIL);
        count_nent ++;
        nent_squeue = 1;
    }
    }

/** state (NENT_arrival) exit executives **/
FSM_STATE_EXIT_FORCED (3, "NENT_arrival", "mesh_BS () [NENT_arrival exit execs]")
    {
    if (op_prg_odb_ltrace_active("state"))
        printf ("\\nBS: NENT/Exit:\\n ");
    }

/** state (NENT_arrival) transition processing **/
FSM_TRANSIT_FORCE (6, state6_enter_exec, ;, "default", "", "NENT_arrival", "SELECT_TO")
    /*---------------------------------------------------------*/


/** state (CTRL_arrival) enter executives **/
FSM_STATE_ENTER_FORCED (4, state4_enter_exec, "CTRL_arrival", "mesh_BS () [CTRL_arrival enter execs]")
    {
    if (op_prg_odb_ltrace_active("state"))
        printf("BS: Entrando Arrival_CTRL....\\n");

    ctrl_arrival = 0;
    nent_arrival = 0;
    data_arrival = 0;

    if (!strcmp(pk_format, MSH_PK_FMT)){

        op_pk_nfd_get(nent_arrived, "Header", &hdr_ptr);

        /*********************************************************
         * Figure out which type of message is in the management packet
         *********************************************************/
        op_pk_nfd_get(nent_arrived, "Payload", &msg_ptr);
        op_pk_format(msg_ptr, pk_format);

        if (op_prg_odb_ltrace_active("pkt_msg")){
        printf("--------------------> El  formato es  %s. \\n"
           "  \\t %s\\n",
           pk_format,MSH_NENT_PK_FMT);
        }
        if (!strcmp(pk_format, MSH_NENT_PK_FMT))
        {
            nent_arrival = 1;
            time_now = op_sim_time();

            if (op_prg_odb_ltrace_active("time_dg"))
                printf("BS, frame: %d, slot:%d, receive NENT at time: %f\\n",
                    (int) (ConvTO_NENT (time_now)/7),((int) (ConvTO_NENT (time_now)/7))%7 ,
                     time_now);

        }else{

            data_arrival = 1;
            if (op_prg_odb_ltrace_active("pkt_msg"))
                printf("\\n\\tBase Station - Data Arrival - type %s. \\n"
                    "  \\t Should have been %s or %s\\n",
                    pk_format, MSH_NCFG_PK_FMT, MSH_NENT_PK_FMT);
        }
    }else{

        printf("\\n\\t Base Station - FATAL ERROR -\\n"
           "  \\t Idle state decapsulated packet of type %s. \\n"
           "  \\t Should have been %s or % s\\n",
           pk_format, MSH_NCFG_PK_FMT, MSH_NENT_PK_FMT);
           op_sim_end("","","","");
    }
    }

/** state (CTRL_arrival) transition processing **/
FSM_INIT_COND (NENT_ARRIVAL)
FSM_TEST_COND (DATA_ARRIVAL)
FSM_TEST_LOGIC ("CTRL_arrival")

FSM_TRANSIT_SWITCH
    {
    FSM_CASE_TRANSIT (0, 3, state3_enter_exec, ;, "NENT_ARRIVAL", "", "CTRL_arrival", "NENT_arrival")
```

```
    FSM_CASE_TRANSIT (1, 5, state5_enter_exec, ;, "DATA_ARRIVAL", "", "CTRL_arrival", "DATA_arrival")
    }
    /*---------------------------------------------------------*/

/** state (DATA_arrival) enter executives **/
FSM_STATE_ENTER_FORCED (5, state5_enter_exec, "DATA_arrival", "mesh_BS () [DATA_arrival enter execs]")
    {
    if (op_prg_odb_ltrace_active("state"))
        printf("BS at %lf: DATA_Arrival/Enter\\n",op_sim_time());
    }

/** state (DATA_arrival) transition processing **/
FSM_TRANSIT_FORCE (1, state1_enter_exec, ;, "default", "", "DATA_arrival", "IDLE")
    /*---------------------------------------------------------*/

/** state (SELECT_TO) enter executives **/
FSM_STATE_ENTER_FORCED (6, state6_enter_exec, "SELECT_TO", "mesh_BS () [SELECT_TO enter execs]")
    {
    if (op_prg_odb_ltrace_active("state"))
        printf("BS at %lf: SELECT_TO/Enter\\n", op_sim_time());

    wait_tx = 0;

    listSize = op_prg_list_size(competitiveList);

    time_mx = ConvTO_NCFG(op_sim_time());
    time_tx_NCFG = ConvTime_NCFG(time_mx);

    if (count_nent > 0){

        win_NCFG = selected_time_NCFG(listSize, node_serial_id, time_tx_NCFG);
        if (win_NCFG ==1 && wait_ncfg == 0){

            nent_arrived = op_subq_pk_remove (0, OPC_QPOS_HEAD);

            op_pk_nfd_get(nent_arrived, "Header", &hdr_ptr);
            op_pk_nfd_get(nent_arrived, "Payload", &msg_ptr);
            op_pk_nfd_get(hdr_ptr, "MAC Source", &mac_source);

            mac_dest = mac_source;

            /* schedule self interrupt based on packets insert time */
            op_pk_nfd_get(msg_ptr, "Type", &type_nent);
            if (op_prg_odb_ltrace_active("pkt_msg"))
                printf("\\n\\tBS*********EL paquete NENT recibido es de tipo: %d,
                *** recibido de %d \\n",type_nent, mac_source);
            if (op_prg_odb_ltrace_active("pkt_msg"))
                printf("\\n\\tBS*********EL paquete NENT recibido es de tipo: %d ,
                recibido en tiempo: %10.8f\\n",type_nent,op_sim_time());
            if (type_nent == NET_ENTRY_REQ)
                {
                    op_pk_nfd_get(msg_ptr, "NENT_Request", &nent_request_ptr);
                    op_pk_nfd_get(nent_request_ptr, "MAC Address", &mac_address);
                    // se dan de alta los nodos en caso de que no este dado previamente de alta,
                    // ya que existen redundancia por los timeouts
                    if(!in_network(mac_address)){
                        store_sids(nent_request_ptr);
                        print_master_list();
                        ncfg_send = 1;
                        }else{
                           ncfg_send = 0;
                           }

                    }else{
                        if (type_nent == NET_ENTRY_CLOSE){
                            op_pk_nfd_get(msg_ptr, "NENT_Request", &nent_request_ptr);
                            op_pk_nfd_get(nent_request_ptr, "MAC Address", &mac_address);
                            if(!in_network(mac_address)){
                               store_sids(nent_request_ptr);
                               print_master_list();
                               ncfg_send = 1;
                               }else{
                                   ncfg_send = 0;
                                   }
                           }
                        if(mac_address == mac_source)
                            mac_address = mac_source;
                        }

                        if (type_nent == NET_ENTRY_ACK){
                            ncfg_send = 0;
                       }else{
                            ncfg_send = 1;
                            listSize = op_prg_list_size(init_neigh);
                            }
```

```
                    wait_ncfg = 1;
                    if (op_prg_odb_ltrace_active("time_dg"))
                        printf("BS, out squeue MSH-NENT type: %d from mac_source %d at time %f \\n\\n", type_nent,  mac_source, op_sim_time());

                    if(ncfg_send==0){
                        wait_ncfg = 0;
                        op_intrpt_schedule_self(op_sim_time(), WAIT_TX_CODE);

                    }else{
                    op_intrpt_schedule_self (time_tx_NCFG, NCFG_CODE);

                   }
                    update_state_node(mac_address, type_nent);

                    count_nent --;
                }else{
                    time_mx = ConvTO_NCFG(op_sim_time());
                    time_tx_NCFG = ConvTime_NCFG(time_mx);
                    op_intrpt_schedule_self(time_tx_NCFG, WAIT_TX_CODE);
                    }
            }
            }
          /** state (SELECT_TO) transition processing **/
          FSM_TRANSIT_FORCE (1, state1_enter_exec, ;, "default", "", "SELECT_TO", "IDLE")
              /*---------------------------------------------------------*/
          }
      FSM_EXIT (0,mesh_BS)
      }
  }

Compcode mesh_BS_init (void ** gen_state_pptr)
  {
  int _block_origin = 0;
  static VosT_Address obtype = OPC_NIL;

  FIN (mesh_BS_init (gen_state_pptr))

  if (obtype == OPC_NIL)
      {
      /* Initialize memory management */
      if (Vos_Catmem_Register ("proc state vars (mesh_BS)",
          sizeof (mesh_BS_state), Vos_Vnop, &obtype) == VOSC_FAILURE)
          {
          FRET (OPC_COMPCODE_FAILURE)
          }
      }

  *gen_state_pptr = Vos_Catmem_Alloc (obtype, 1);
  if (*gen_state_pptr == OPC_NIL)
      {
      FRET (OPC_COMPCODE_FAILURE)
      }
  else
      {
      /* Initialize FSM handling */
      ((mesh_BS_state *)(*gen_state_pptr))->current_block = 0;
      FRET (OPC_COMPCODE_SUCCESS)
      }
  }

void mesh_BS_diag (void)
  {
  /* No Diagnostic Block */
  }

void mesh_BS_terminate (void)
  {
  int _block_origin = __LINE__;

  FIN (mesh_BS_terminate (void))
      Vos_Catmem_Dealloc (pr_state_ptr);
  FOUT;
  }
```

C.2. Estación suscriptora Mesh

```
/* Process model C form file: mesh_ss.pr.c */

/* Header Block */

#include "mesh_support.h"
#include "mesh_gral.h"
```

```
#define NCFG_CODE       10
#define NCFG_FWD_CODE   15
#define NENT_CODE       20
#define NENT_TOUT_CODE  21
#define NENT_FWD_CODE   25
#define NENT_SQUEUE_CODE    27

/* Function Block */

static void insert_node(int address){
    NEIGHBOR_INIT_STRUCT* neighInit;

    int i=0;
    int listSize = 0;
    FIN (init_parameter_reg());
        neighInit = (NEIGHBOR_INIT_STRUCT*) op_prg_mem_alloc (sizeof (NEIGHBOR_INIT_STRUCT));
        neighInit->xmt_holdoff_exp = 11;
        neighInit->next_tx_exp = 0;
        neighInit->nodeID = address;
        op_prg_list_insert(nodeGralList,neighInit,OPC_LISTPOS_TAIL);
    if(address < 1041){
        op_prg_list_insert(competitiveList,neighInit,OPC_LISTPOS_TAIL);

    }
    FOUT;
}

static void encap_nent_req( Packet* request, int mac_dest, int
mac_mes){
    Packet* hdr_ptr;

    FIN(encap_nent_req());

        pkt_send = op_pk_create_fmt(MSH_PK_FMT);
        hdr_ptr = op_pk_create_fmt(MSH_HD_PK_FMT);
        msh_nent_ptr = op_pk_create_fmt(MSH_NENT_PK_FMT);

        op_pk_nfd_set(msh_nent_ptr, "Management Message Type", 40);
        op_pk_nfd_set(msh_nent_ptr, "Type", type_nent);
        op_pk_nfd_set(msh_nent_ptr, "Sponsor Node ID", mac_parent);
        op_pk_nfd_set(msh_nent_ptr, "Xmt Power", xmt_power);
        op_pk_nfd_set(msh_nent_ptr, "Xmt Antenna", xmt_power);
        op_pk_nfd_set(msh_nent_ptr, "NENT_Request", request);

        op_pk_nfd_set (hdr_ptr, "len", 10);
        op_pk_nfd_set (hdr_ptr, "MAC Source", mac_address);
        op_pk_nfd_set (hdr_ptr, "MAC Destination", mac_dest);

        op_pk_nfd_set(pkt_send, "Header", hdr_ptr);
        op_pk_nfd_set(pkt_send, "Payload", msh_nent_ptr);

        op_pk_total_size_set (pkt_send, 208);


    FOUT;
}


static Packet* encap_ncfg_rply (Packet* hdr_ptr,Packet* msg_ptr,
Packet* emb_data_ptr, int type, int mac_dest, int mac_mes)
    {


    int type_msg, no_nbr_ent, no_bs_ent, emb_pkt_flag,  net_ent_flag, net_ent_mac_addr, time_frame_no, time_net_slot, net_cfg_cnt; //, address;
    int len;
    double xmt_pwr, xmt_ant;
    double holdoff, next_xmt;
    Packet* msh_msg;
    Packet* mac_frame;
    Packet* emb_data_pkt;
    Packet* emb_data_IE_pkt;
    Packet* msh_bs_pkt;

    FIN (encap_ncfg_rply ());
        op_pk_nfd_get (emb_data_ptr, "Length", &len);
        op_pk_nfd_get (emb_data_ptr, "Embedded_data_IE", &emb_data_IE_pkt);

        emb_data_pkt = op_pk_create_fmt (MSH_NCFG_EMB_DATA_PK_FMT);
        op_pk_nfd_set (emb_data_ptr, "Type", type);
        op_pk_nfd_set (emb_data_ptr, "Length", 10);
        op_pk_nfd_set (emb_data_ptr, "Embedded_data_IE", emb_data_IE_pkt);


    op_pk_nfd_get (msg_ptr, "Management Message Type", &type_msg);
    op_pk_nfd_get (msg_ptr, "Nbr Entries", &no_nbr_ent);
    op_pk_nfd_get (msg_ptr, "BS Entries",  &no_bs_ent);
```

```
    op_pk_nfd_get (msg_ptr, "Embedded Packet Flag", &emb_pkt_flag);
    op_pk_nfd_get (msg_ptr, "Xmt Power", &xmt_pwr);
    op_pk_nfd_get (msg_ptr, "Xmt Antenna", &xmt_ant);
    op_pk_nfd_get (msg_ptr, "NetEntry MAC Address Flag", &net_ent_flag);
    op_pk_nfd_get (msg_ptr, "NetConfig Count", &net_ent_mac_addr);
    op_pk_nfd_get (msg_ptr, "Timestamp Frame Number", &time_frame_no);
    op_pk_nfd_get (msg_ptr, "Timestamp Net CSlot", &time_net_slot);
    op_pk_nfd_get (msg_ptr, "Timestamp Sync Hop Count", &net_cfg_cnt);
    op_pk_nfd_get (msg_ptr, "Xmt Holdoff Exponent", &holdoff);
    op_pk_nfd_get (msg_ptr, "Next Xmt Mx", &next_xmt);
    op_pk_nfd_get (msg_ptr, "BS Descriptor", &msh_bs_pkt);
    op_pk_nfd_set (msh_bs_pkt, "Number of hops", num_hops);
    op_pk_nfd_get (msg_ptr, "Node Nbr Descriptor", &neigh_data);


    msh_msg = op_pk_create_fmt (MSH_NCFG_PK_FMT);
    /* Set the destination address of the packet. */
    op_pk_nfd_set (msh_msg, "Management Message Type", type_msg);
    op_pk_nfd_set (msh_msg, "Nbr Entries", no_nbr_ent);
    op_pk_nfd_set (msh_msg, "BS Entries",  no_bs_ent);
    op_pk_nfd_set (msh_msg, "Embedded Packet Flag", emb_pkt_flag);
    op_pk_nfd_set (msh_msg, "Xmt Power", xmt_pwr);
    op_pk_nfd_set (msh_msg, "Xmt Antenna", xmt_ant);
    op_pk_nfd_set (msh_msg, "NetEntry MAC Address Flag", net_ent_flag);
    op_pk_nfd_set (msh_msg, "NetConfig Count", net_ent_mac_addr);
    op_pk_nfd_set (msh_msg, "Timestamp Frame Number", time_frame_no);
    op_pk_nfd_set (msh_msg, "Timestamp Net CSlot", time_net_slot);
    op_pk_nfd_set (msh_msg, "Timestamp Sync Hop Count", net_cfg_cnt);
    op_pk_nfd_set (msh_msg, "Xmt Holdoff Exponent", holdoff);
    op_pk_nfd_set (msh_msg, "Next Xmt Mx", next_xmt);
    op_pk_nfd_set (msh_msg, "NetEntry MAC Address", mac_mes);

    op_pk_nfd_set (msh_msg, "BS Descriptor", msh_bs_pkt);
    op_pk_nfd_set (msh_msg, "Node Nbr Descriptor",neigh_data,op_prg_mem_copy_create, op_prg_mem_free, sizeof (NEIGHBOR_DATA_STRUCT));
    op_pk_nfd_set (msh_msg, "Embedded Data", emb_data_ptr);


    /* Create MAC management packet. */
    mac_frame = op_pk_create_fmt (MSH_PK_FMT);

    hdr_ptr = op_pk_create_fmt (MSH_HD_PK_FMT);
    /* Set the destination address of the packet. */
    op_pk_nfd_set (hdr_ptr, "len", 10);

    op_pk_nfd_set (hdr_ptr, "MAC Source", mac_address);
    op_pk_nfd_set (hdr_ptr, "MAC Destination", mac_dest);

    op_pk_nfd_set(mac_frame, "Header", hdr_ptr);
    op_pk_nfd_set(mac_frame, "Payload", msh_msg);


    FRET (mac_frame);
    }

static Packet* encap_nent_fwd(Packet *msg_ptr,int mac_sponsor,Packet
*hdr_ptr,int mac_child, Packet* nent_req, int mac_mes){

    Packet* nent_req_ptr;
    int mng_type = 0;
    double power,ant;
    int opConInfo, opAuth, nodeSerial;

    FIN(encap_nent_fwd());

        nent_req_ptr = op_pk_create_fmt(MSH_NENT_REQ_PK_FMT);

        if (type_nent == NET_ENTRY_REQ)
            {
            op_pk_nfd_get(nent_req, "OpConfInfo", &opConInfo);
            op_pk_nfd_get(nent_req, "Operator Authentication Value", &opAuth);
            op_pk_nfd_get(nent_req, "Node serial Number", &nodeSerial);

            op_pk_nfd_set(nent_req_ptr, "MAC Address", mac_child);
            op_pk_nfd_set(nent_req_ptr, "OpConfInfo", opConInfo);
            op_pk_nfd_set(nent_req_ptr, "Operator Authentication Value", opAuth);
            op_pk_nfd_set(nent_req_ptr, "Node serial Number", nodeSerial);
            }else{
                op_pk_nfd_set(nent_req_ptr, "MAC Address", mac_child);
            }

        pkt_send = op_pk_create_fmt(MSH_PK_FMT);

        msh_nent_ptr = op_pk_create_fmt(MSH_NENT_PK_FMT);

        op_pk_nfd_get(msg_ptr, "Management Message Type", &mng_type);
        op_pk_nfd_get(msg_ptr, "Xmt Power", &power);
```

```
        op_pk_nfd_get(msg_ptr, "Xmt Antenna", &ant);

        op_pk_nfd_set(msh_nent_ptr, "Management Message Type", mng_type);
        op_pk_nfd_set(msh_nent_ptr, "Type", type_nent);
        op_pk_nfd_set(msh_nent_ptr, "Sponsor Node ID", mac_sponsor);
        op_pk_nfd_set(msh_nent_ptr, "Xmt Power", power);
        op_pk_nfd_set(msh_nent_ptr, "Xmt Antenna", ant);
        op_pk_nfd_set(msh_nent_ptr, "NENT_Request", nent_req_ptr);
        op_pk_nfd_set(hdr_ptr, "MAC Source", mac_address);
        op_pk_nfd_set(hdr_ptr, "MAC Destination", mac_parent);

        op_pk_destroy(msg_ptr);
        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", msh_nent_ptr);


    FRET(pkt_send);



}

static int isChild(int mac_ch){

    sidEntry *childEntry;

    int sponsorID = 0;
    int i = 0;
    int listSize;
    FIN(isChild());

        listSize = op_prg_list_size(childList);

        printf("\n\t\t ************************Child List with %d users\n",  listSize);
        for(i = 0;i < listSize;i++)
            {
                childEntry = (sidEntry *) op_prg_list_access(childList,i);
                if(childEntry->SID == mac_ch){
                    sponsorID = childEntry-> sponsorID;
                    break;
                }

        }
    printf("\n");

    FRET(sponsorID);
}

static int get_state_node(int mac){

    sidEntry *masterEntry;

    int state_node = 0;
    int i = 0;
    int listSize;

    FIN(get_state_node());

        listSize = op_prg_list_size(masterList);

        printf("\n\t\t ******************** get state Master List with %d users\n",  listSize);
        for(i = 0;i < listSize;i++)
            {
                masterEntry = (sidEntry *) op_prg_list_access(masterList,i);
                if(masterEntry->SID == mac){
                     state_node = masterEntry->link;
                    break;
                }

        }

    printf("\n");
    FRET(state_node);
}

static Packet* encap_nent_queue(Packet *msg_ptr, Packet* hdr_ptr,
    int mac_dest, Packet* nent_req, int mac, int type){

    Packet* pkt_nent;
    FIN(encap_nent_queue());

        op_pk_nfd_set(nent_req, "MAC Address", mac);
        op_pk_nfd_set(msg_ptr, "Type", type);
        op_pk_nfd_set(msg_ptr, "NENT_Request", nent_req);
        pkt_nent = op_pk_create_fmt(MSH_PK_FMT);
```

```
        op_pk_nfd_set(hdr_ptr, "MAC Destination", mac_dest);
        op_pk_nfd_set(pkt_nent, "Header", hdr_ptr);
        op_pk_nfd_set(pkt_nent, "Payload", msg_ptr);

    FRET(pkt_nent);
}

static void store_sids_child(int mac, int mac_sponsor){

    /*FIN(store_sids_child());
        op_prg_list_insert(childList,mac,OPC_LISTPOS_TAIL);
    FOUT;*/

    sidEntry *childEntry;


    FIN(store_sids_child());
        if (op_prg_odb_ltrace_active("func_track"))
            printf("\t\t\t FUNCTION - store_sids   \n");

        printf(" Receives Registration Request for Child-SID %d\n",mac);

    /* Make an entry in the master list. */
        childEntry = (sidEntry *)op_prg_mem_alloc(sizeof(sidEntry));
        childEntry->SID = mac;
        childEntry->macID = 0;
        childEntry->sponsorID = mac_sponsor;
        childEntry->link = 0;
        childEntry->xmt_holdoff_exp = 0;
        childEntry->next_tx_exp = 0;
        //childEntry->state = 1;
        op_prg_list_insert(childList,childEntry,OPC_LISTPOS_TAIL);


    FOUT;

}

static void print_gral_list(void) {
    int i, listSize;
    NEIGHBOR_INIT_STRUCT *listEntry;

    FIN(print_master_list());

        if (op_prg_odb_ltrace_active("func_track"))
            printf("\t\t\t FUNCTION - print_master_list   \n");
        listSize = op_prg_list_size(nodeGralList);
        printf("\n\t\t ************************SS: Gral List with % users %d\n",  listSize);
        for(i = 0;i < listSize;i++){
            listEntry = (sidEntry *) op_prg_list_access(nodeGralList,i);
            printf("\t\tSID %d", listEntry->nodeID);
            }
    printf("\n");
    FOUT;

    }

static void print_child_list() {
    int i, listSize;
    sidEntry *childEntry;

    FIN(print_child_list());
        listSize = op_prg_list_size(childList);
        printf("\n\t\t ************************Child List with %d users\n",  listSize);
        for(i = 0;i < listSize;i++)
            {
            childEntry = (sidEntry *) op_prg_list_access(childList,i);
            printf("\t\tSID %d, ", childEntry->SID);
            printf("\t\tSponsorID %d \n", childEntry->sponsorID);
            }
    printf("\n");
    FOUT;
 }


static long double delay_calc(double distance){
    double delay = 0;
    delay = distance / 300000000 ;

    return delay;
}


/* Process model interrupt handling procedure */
```

```
void mesh_ss (void)
    {

        FSM_BLOCK_SWITCH
            {
            /*---------------------------------------------------------*/
            /** state (INIT) enter executives **/
            FSM_STATE_ENTER_FORCED_NOLABEL (0, "INIT", "mesh_ss () [INIT enter execs]")
                {
                if (op_prg_odb_ltrace_active("state"))
                    printf ("******************SS: INIT/Enter:\n ");

                /* Initialization of the process model.             */
                /* All the attributes are loaded in this routine    */
                mesh_ss_mac_sv_init ();

                //Tablas de inicializacion para calculo de tiempo NENT
                init_parameter_reg(10);

                childList = op_prg_list_create();

                op_ima_sim_attr_get (OPC_IMA_INTEGER, "data_nodes_active", &nodes_active);

                intrpt_type = op_intrpt_type ();


                if(intrpt_type == OPC_INTRPT_STRM)
                {
                        input_strm = op_intrpt_strm();
                        intrpt_code = op_intrpt_code();
                        pkt_arrive = op_pk_get (input_strm);
                        op_pk_format(pkt_arrive,pk_format);
                        op_intrpt_schedule_self (op_sim_time(), NCFG_CODE);
                }
                }

            /** state (IDLE) exit executives **/
            FSM_STATE_EXIT_UNFORCED (1, "IDLE", "mesh_ss () [IDLE exit execs]")
                {
                if (op_prg_odb_ltrace_active("state"))
                    printf ("\nMSS %d: IDLE/Exit:at %lf\n " , mac_address,op_sim_time());


                intrpt_type = op_intrpt_type ();

                switch (intrpt_type)
                {
                    case OPC_INTRPT_STRM:
                        if (op_prg_odb_ltrace_active("pkt_msg"))
                            printf("MSS %d: Interruption strm q llega: %d, Codigo: %d, Tiempo: %f\n",
                                mac_address,intrpt_type, intrpt_code,op_sim_time());

                        input_strm = op_intrpt_strm();
                        intrpt_code = op_intrpt_code();
                        pkt_arrive = op_pk_get (input_strm);
                        op_pk_format(pkt_arrive,pk_format);
                        if (op_prg_odb_ltrace_active("pkt_msg")){
                            printf("MSS %d: Recibiendo pkt con formato: %s\n", mac_address,pk_format);
                            printf("MSS %d: Interrupcion Stream\n", mac_address);
                            }


                case OPC_INTRPT_SELF:
                    intrpt_code = op_intrpt_code();
                    /*REcibiendo mensaje de inicializacion*/
                        switch(intrpt_code){
                            case NCFG_CODE:
                                intrpt_type = OPC_INTRPT_STRM;
                                break;
                            case NCFG_FWD_CODE:
                                if(fwd_send ==0){
                                    ncfg_fwd = 1;

                                    }else
                                    ncfg_fwd = 0;
                                break;

                            case NENT_FWD_CODE:
                                    nent_fwd = 1;
                                    wait_nent = 0;
                                break;
                            case NENT_SQUEUE_CODE:
                                    nent_out = 1;
                                break;
                            case NENT_CODE:
                                   tout_nent = 0;
```

```
                break;
            default:
                break;
            }
            break;
     default:
        printf("MSS %d: default interrupcion%d\n", mac_address, intrpt_type);
        break;
    }

    }


/** state (IDLE) transition processing **/
FSM_INIT_COND (TRAFFIC_ARRIVAL)
FSM_TEST_COND (NENT_SEND)
FSM_TEST_COND (NCFG_FWD)
FSM_TEST_COND (NENT_FWD)
FSM_TEST_COND (NENT_OUT)
FSM_DFLT_COND
FSM_TEST_LOGIC ("IDLE")

FSM_TRANSIT_SWITCH
    {
    FSM_CASE_TRANSIT (0, 4, state4_enter_exec, ;, "TRAFFIC_ARRIVAL", "", "IDLE", "TRAFFIC_Arrived")
    FSM_CASE_TRANSIT (1, 3, state3_enter_exec, ;, "NENT_SEND", "", "IDLE", "NENT_send")
    FSM_CASE_TRANSIT (2, 7, state7_enter_exec, ;, "NCFG_FWD", "", "IDLE", "NCFG_fwd")
    FSM_CASE_TRANSIT (3, 8, state8_enter_exec, ;, "NENT_FWD", "", "IDLE", "NENT_fwd")
    FSM_CASE_TRANSIT (4, 6, state6_enter_exec, ;, "NENT_OUT", "", "IDLE", "NENT_squeue")
    FSM_CASE_TRANSIT (5, 1, state1_enter_exec, ;, "default", "", "IDLE", "IDLE")
    }
    /*---------------------------------------------------------*/



/** state (NCFG_Arrival) enter executives **/
FSM_STATE_ENTER_FORCED (2, state2_enter_exec, "NCFG_Arrival", "mesh_ss () [NCFG_Arrival enter execs]")
    {
    if (op_prg_odb_ltrace_active("state"))
        printf ("\n******************SS: NCFG_Arrival/Enter:\n ");

    tout_nent = 0;

    op_pk_nfd_get(pkt_arrive, "Header", &hdr_ptr);
    op_pk_format(hdr_ptr, hd_format);
    op_pk_nfd_get(hdr_ptr, "MAC Source", &mac_source);
    op_pk_nfd_get(hdr_ptr, "MAC Destination", &mac_dest);
    op_pk_nfd_get(msg_ptr, "Node Nbr Descriptor", &neigh_data);
    op_pk_nfd_get(msg_ptr, "NetEntry MAC Address", &msg_mac);
    op_pk_nfd_get(msg_ptr, "Embedded Data", &emb_data_ptr);

    if( msg_mac == mac_address || ( (mac_dest == 0) && (register_node == 0 || register_node == 3) ) ){

    if(mac_address == mac_dest)
        op_pk_nfd_get(emb_data_ptr, "Type", &type_ncfg);
    else
        if (msg_mac == 0)
            op_pk_nfd_get(emb_data_ptr, "Type", &type_ncfg);

            if (op_prg_odb_ltrace_active("time_dg"))
                printf("MSS %d, Rcv MSH-NCFG type:  from mac_source %d with mac_address %d\n ",
                    mac_address, type_ncfg, msg_mac, mac_source);

        /* Si es un type_ncfg = 0, el nodo aun no ha sido registrado y NCFG con mac = 0 */
        if(type_ncfg == NCFG_NETWORK_DESCRIPTOR && register_node == 0 && msg_mac == 0){
            op_pk_nfd_get (msg_ptr, "BS Descriptor", &msh_bs_pkt);
            op_pk_nfd_get (msh_bs_pkt, "Number of hops", &num_hops);
            num_hops ++;

            if (op_prg_odb_ltrace_active("pkt_msg"))
                printf("MSS %d, Es de tipo Network Descriptor\n", mac_address);


            /* Inicializa el proceso de registro se guarda al nodo padre, aquel nodo de quién recibio el NCFG, mac = 0*/
            mac_parent = mac_source;

            /* calcular valores de NextXmtTime*/
            listSize = op_prg_list_size(init_neigh);

            time_tx_NENT = selected_time_NENT( node_serial_id, op_sim_time(), prop_distance);
            time_tx_NENT = time_tx_NENT - delay_value;

            register_node = 1;
            nent_send = 1;
            if (op_prg_odb_ltrace_active("pkt_msg")){
                printf ("MSS: %d NODO: %d, tiempo para NENT: %10.8lf\n", mac_address, node_serial_id, time_tx_NENT);
```

```
        }
    /*Envia el tiempo calculado para que sea enviado el mensaje NENT*/
    op_intrpt_schedule_self (time_tx_NENT, NENT_CODE);
    op_pk_destroy(pkt_arrive);
    op_pk_destroy(msg_ptr);

}else{

    if(type_ncfg == NCFG_NETWORK_ENT_OPEN && register_node == 1 && msg_mac == mac_address ){
        if (op_prg_odb_ltrace_active("pkt_msg"))
            printf("MSS %d: Es de tipo OPEN\n",mac_address);
        time_tx_NENT = selected_time_NENT( node_serial_id, op_sim_time(), prop_distance);
        time_tx_NENT = time_tx_NENT - delay_value;
        nent_send = 1;
        //Se calcula el tiempo de envio por medio del algoritmo
        printf("MSS: %d NODO: %d, tiempo para NENT: %f\n", mac_address, node_serial_id,  time_tx_NENT);
        op_intrpt_schedule_self (time_tx_NENT, NENT_CODE);
        register_node = 2;

        }else{

            if(register_node == 3 && type_ncfg == NCFG_NETWORK_DESCRIPTOR){
                printf("MSS %d EL NODO YA ESTA REGISTRADO!!!!!!!!!!!!\n", mac_address);
                if(mac_source == mac_parent){
                    printf("El nodo recibio un mensaje de su padre, tipo ncfg antes de encap:
                        %d con destino a:\n", type_ncfg, mac_source);
                    //empaqueta ncfg
                    sponsorID = isChild (msg_mac);
                    if(sponsorID != 0)
                    ncfg_rply_ptr = encap_ncfg_rply(hdr_ptr, msg_ptr,emb_data_ptr, type_ncfg, sponsorID, msg_mac);
                   else
                    ncfg_rply_ptr = encap_ncfg_rply(hdr_ptr, msg_ptr,emb_data_ptr, type_ncfg,  0, msg_mac);
                    /* Calcula el tiempo para la siguiente TO de acuerdo al tiempo actual */

                    printf("El nodo recibio un mensaje de su padre, tipo ncfg antes de encap: %d con destino a:
                        %d o mac_parent: %d\n", type_ncfg, sponsorID, mac_parent);
                    time_mx = ConvTO_NCFG(op_sim_time());
                    time_tx_NCFG = ConvTime_NCFG(time_mx);

                    printf(" tiempo para interrupcion time_tx: %lf\n",time_tx_NCFG);
                    fwd_send = 0;

                    op_pk_destroy(pkt_arrive);
                    op_intrpt_schedule_self(time_tx_NCFG, NCFG_FWD_CODE);

                    }else{

                        printf("MSS %d, El padre es: %d, mac_source es: %d",mac_address, mac_parent, mac_source);
                        }

                }else{

                if(register_node == 2 && type_ncfg == NCFG_NETWORK_ENT_ACK  && msg_mac == mac_address ){
                    printf("MSS %d Termina proceso de entrada del nodo!!!!!!!!!!!!\n\n",mac_address);
                    register_node = 3;
                    insert_node(mac_address);
                    print_gral_list();
                    }else{
                 /*Verifica si es su hijo*/
                    sponsorID = isChild (msg_mac);
                    if( sponsorID != 0 ){
                        if(sponsorID == mac_address){
                            sponsorID = msg_mac;

                        }
                        op_pk_nfd_get(emb_data_ptr, "Type", &type_ncfg);
                            printf("MSS %d El nodo recibio un mensaje de su padre, tipo ncfg antes de encap: %d,
                                 de: %d con destino a: %d\n",mac_address, type_ncfg, mac_source,sponsorID);
                            ncfg_rply_ptr = encap_ncfg_rply(hdr_ptr, msg_ptr,emb_data_ptr, type_ncfg, sponsorID, msg_mac);
                        /* Calcula el tiempo para la siguiente TO de acuerdo al tiempo actual */
                        fwd_send = 0;
                        time_mx = ConvTO_NCFG(op_sim_time());
                        time_tx_NCFG = ConvTime_NCFG(time_mx);
                        if(tout_fwd_ncfg > op_sim_time()){
                            tout_fwd_ncfg = time_tx_NCFG + 6.0;
                            op_intrpt_schedule_self(time_tx_NCFG, NCFG_FWD_CODE);
                        }else{

                        printf("MSS %d: Este nodo ya no reenvía NCFG", mac_address);
                        }
                        }

                        }
                    }
            }
    }
```

```
        if (op_prg_odb_ltrace_active("pkt_msg")){
            printf("MSS:Valor REgistrado: %d\n", register_node);
            printf("MSS: Valor tipo mensaje NCFG: %d\n",type_ncfg);
            printf("MSS: Valor MAC: %d\n",msg_mac);
        }
    }else{
        /*Verifica que mac destino del paquete entrante pertezca o no a un hijo*/
        if(mac_dest == mac_address ){
                sponsorID = isChild (msg_mac);
                if(sponsorID == mac_address){
                    sponsorID = msg_mac;
                    }
                op_pk_nfd_get(emb_data_ptr, "Type", &type_ncfg);
                printf("MSS %d El nodo recibio un mensaje de su padre, tipo ncfg antes de encap: %d, de: %d con destino a: %d\n",
                    mac_address,type_ncfg, mac_source,mac_dest);
                ncfg_rply_ptr = encap_ncfg_rply(hdr_ptr, msg_ptr,emb_data_ptr, type_ncfg, sponsorID, msg_mac);

                /* Calcula el tiempo para la siguiente TO de acuerdo al tiempo actual */
                fwd_send = 0;
                time_mx = ConvTO_NCFG(op_sim_time());

                time_tx_NCFG = ConvTime_NCFG(time_mx);
                            op_intrpt_schedule_self(time_tx_NCFG, NCFG_FWD_CODE);
            }else{
               if (op_prg_odb_ltrace_active("pkt_msg"))
                    printf("MSS %d el paquete NCFG ignorado por que era para la mac %d y register_node %d\n\n",
                        mac_address,mac_dest, register_node);
         }
     }
    }

/** state (NENT_send) enter executives **/
FSM_STATE_ENTER_FORCED (3, state3_enter_exec, "NENT_send", "mesh_ss () [NENT_send enter execs]")
    {
    if (op_prg_odb_ltrace_active("state"))
        printf("** MSS %d : NENT_SEND/enter\n", mac_address);
    nent_req_ptr = op_pk_create_fmt(MSH_NENT_REQ_PK_FMT);
    switch(type_ncfg)
    {
        case NCFG_NETWORK_DESCRIPTOR:
            type_nent = NET_ENTRY_REQ;
            if (op_prg_odb_ltrace_active("pkt_msg"))
                printf("SS %d:Recibi NCFG_NETWORK_DESCRIPTOR, preparo NET_ENTRY_REQ\n", mac_address);
            break;
         case NCFG_NETWORK_ENT_OPEN:
            type_nent = NET_ENTRY_ACK;
            if (op_prg_odb_ltrace_active("pkt_msg"))
                printf("SS %d: REcibi NCFG_NETWORK_ENT_OPEN, preparo NET_ENTRY_ACK\n", mac_address);
            break;
         case NCFG_NETWORK_ENT_OPEN_1:
            type_nent = NET_ENTRY_CLOSE;
            if (op_prg_odb_ltrace_active("pkt_msg"))
                printf("SS %d: REcibi NCFG_NETWORK_ENT_OPEN_1, preparo NET_ENTRY_ACK\n", mac_address);
            break;
         case NCFG_NETWORK_ENT_REJECT:
            //Falta decidir en caso de un reject
            type_nent = NET_ENTRY_REQ;
            if (op_prg_odb_ltrace_active("pkt_msg"))
                printf("NET_ENTRY_REJECT??\n");
            break;
         case NCFG_NETWORK_ENT_ACK:
            type_nent = NET_ENTRY_REQ;
            if (op_prg_odb_ltrace_active("pkt_msg")){
                    printf("SS %d: REcibi NCFG_NETWORK_ENT_ACK\n",mac_address);
                    }
            break;
         case NCFG_NEIGHBOR_LINK_EST:
            // se actualiza la lista de vecinos type_nent = NET_ENTRY_REQ;
            if (op_prg_odb_ltrace_active("pkt_msg"))
                printf("SS: REcibi NCFG_NEIGHBOR_LINK_ESTK\n");
            break;
          default:
            printf("FORMATO DE NCFG NO RECONOCIDO");
            break;

        }

    state_node = get_state_node(mac_address);
        if(type_nent >= state_node){
            tout_nent = 1;
        }else{
            tout_nent = 0;
            }

    if (tout_nent == 1){
```

```
        switch(type_nent){
            case NET_ENTRY_REQ:
                op_pk_nfd_set(nent_req_ptr, "MAC Address", mac_address);
                op_pk_nfd_set(nent_req_ptr, "OpConfInfo", 0);
                op_pk_nfd_set(nent_req_ptr, "Operator Authentication Value", 0);
                op_pk_nfd_set(nent_req_ptr, "Node serial Number", 0);
                if (op_prg_odb_ltrace_active("pkt_msg"))
                printf("SS %d:Recibi NCFG_NETWORK_DESCRIPTOR, Envio NET_ENTRY_REQ\n", mac_address);

                break;
            case NET_ENTRY_ACK:
                op_pk_nfd_set(nent_req_ptr, "MAC Address", mac_address);
                if (op_prg_odb_ltrace_active("pkt_msg"))
                printf("SS %d: REcibi NCFG_NETWORK_ENT_OPEN, envio NET_ENTRY_ACK\n", mac_address);
                break;
            case NET_ENTRY_CLOSE:
                op_pk_nfd_set(nent_req_ptr, "MAC Address", mac_address);
                if (op_prg_odb_ltrace_active("pkt_msg"))
                printf("SS %d: REcibi NCFG_NETWORK_ENT_OPEN, envio NET_ENTRY_CLOSE\n", mac_address);
                break;
            default:
                printf("FORMATO DE NENT NO RECONOCIDO");
                break;
        }

        if (op_prg_odb_ltrace_active("pkt_msg"))
            printf("SS: register_node %d\n",register_node);

        encap_nent_req(nent_req_ptr, mac_parent, msg_mac);
        if (op_prg_odb_ltrace_active("pkt_msg")){
            printf("************************************SS%d: Enviando pkt.  msh_ent.................**********\n",mac_address);
        }

        op_pk_send (pkt_send,0);

        if (op_prg_odb_ltrace_active("time_dg"))
            printf("\nMSS %d, ENV MSH-NENT type %d, status_node %d, mac_parent %d, at time %f \n",
                 mac_address, type_nent, register_node, mac_parent, op_sim_time());

        if(type_nent == NET_ENTRY_ACK){
                type_ncfg= NCFG_NETWORK_ENT_OPEN_1;
                time_mx = ConvTO_NENT(op_sim_time());

                time_tx_NENT = ConvTime_NENT(time_mx+7);
                time_tx_NENT = time_tx_NENT - delay_value;
                printf("MSS %d, Prg MSH-NENT type CLOSE, status_node %d:\n",tout_nent, type_nent, register_node);
                op_intrpt_schedule_self(time_tx_NENT,NENT_CODE);
        }

        //REcibiendo mensaje de inicializacion
            hops = num_hops - 1;
            if(mac_address <= 1100) temp = (mac_address - 1000) * 0.4;
            if(mac_address <= 1040) temp = (mac_address - 1000) * 0.4;
            if(mac_address <= 1010) temp =  0.4;

        time_tx_NENT = selected_time_NENT( node_serial_id, op_sim_time()+ temp*hops, prop_distance);
        if (op_prg_odb_ltrace_active("time_dg"))
            printf(" tout temp %f, mac_address %d at time MSH-NENT: %f\n",temp, mac_address,time_tx_NENT);
        op_intrpt_schedule_self(time_tx_NENT , NENT_CODE);
    }
    }

/** state (NENT_send) exit executives **/
FSM_STATE_EXIT_FORCED (3, "NENT_send", "mesh_ss () [NENT_send exit execs]")
    {
    if (op_prg_odb_ltrace_active("state"))
        printf("** MSS %d : NENT_SEND/exit\n", mac_address);
    }

/** state (TRAFFIC_Arrived) enter executives **/
FSM_STATE_ENTER_FORCED (4, state4_enter_exec, "TRAFFIC_Arrived", "mesh_ss () [TRAFFIC_Arrived enter execs]")
    {
    if (op_prg_odb_ltrace_active("state"))
        printf("MSS: Entrando a TRAFFIC_arrived");

    ncfg_arrival    = 0;
    nent_arrival    = 0;

    prop_distance = op_td_get_dbl (pkt_arrive, OPC_TDA_RA_START_DIST);
    delay_value = delay_calc(prop_distance);

    /*****************************************************************
    * Check frame format
    *****************************************************************/
    nodes_active = 100;
    if(nodes_active + 1000 < mac_address){
```

```
        data_arrival = 1;
    }else{
      if (strcmp(pk_format,MSH_PK_FMT))
        {
        printf("\n\t MSS %d - FATAL ERROR -"
           "  \t Traffic Arrived state received frame of type %s."
           "  \t Should have been %s\n", mac_address,
           pk_format,MSH_PK_FMT);
           data_arrival = 1;
        }else{

            if (op_prg_odb_ltrace_active("pkt_msg")){
                printf("\n\t SS - %d PKT NCFG RECEIVED -\n"
                       "  \t Format: %s\n", mac_address,
                       pk_format,MSH_PK_FMT);
                }

                op_pk_nfd_get(pkt_arrive, "Payload", &msg_ptr);
                op_pk_format(msg_ptr, pk_format);

                if (op_prg_odb_ltrace_active("pkt_msg")) {
                    printf(" MSS: El  formato recibido --Payload-- es  %s. \n"
                    "  \t %s\n",
                    pk_format,MSH_NCFG_PK_FMT);
                }

                if (!strcmp(pk_format, MSH_NCFG_PK_FMT))
                    {
                    ncfg_arrival = 1;
                    }
                else
                    {
                    if (!strcmp(pk_format, MSH_NENT_PK_FMT))
                        if (op_prg_odb_ltrace_active("pkt_msg")) {
                            printf("MSS %d: El  formato recibido es  %s. \n"
                                "  \t %s\n", mac_address, pk_format,MSH_PK_FMT);
                            }
                        nent_arrival = 1;

                    }

                }
    }
    }

/** state (TRAFFIC_Arrived) transition processing **/
FSM_INIT_COND (NCFG_ARRIVAL)
FSM_TEST_COND (DATA_ARRIVAL)
FSM_TEST_COND (NENT_ARRIVAL)
FSM_TEST_LOGIC ("TRAFFIC_Arrived")

FSM_TRANSIT_SWITCH
    {
    FSM_CASE_TRANSIT (0, 2, state2_enter_exec, ;, "NCFG_ARRIVAL", "", "TRAFFIC_Arrived", "NCFG_Arrival")
    FSM_CASE_TRANSIT (1, 5, state5_enter_exec, ;, "DATA_ARRIVAL", "", "TRAFFIC_Arrived", "DATA_Arrival")
    FSM_CASE_TRANSIT (2, 9, state9_enter_exec, ;, "NENT_ARRIVAL", "", "TRAFFIC_Arrived", "NENT_arrival")
    }
    /*---------------------------------------------------------*/

/** state (NENT_squeue) enter executives **/
FSM_STATE_ENTER_FORCED (6, state6_enter_exec, "NENT_squeue", "mesh_ss () [NENT_squeue enter execs]")
    {
    if (op_prg_odb_ltrace_active("state"))
        printf("SS %d: NENT_squeue/Enter\n", mac_address);

    nent_out = 0;
    ignore_fwd = 0;
    if (count_nent > 0){

        if ( wait_nent == 0){
            time_tx_NENT = selected_time_NENT( node_serial_id, op_sim_time()+.1, prop_distance);

            do{
            pkt_arrive = op_subq_pk_remove (0, OPC_QPOS_HEAD);
            count_nent--;
            }while(pkt_arrive == OPC_NIL && (count_nent > 0));

            op_pk_nfd_get(pkt_arrive, "Header", &hdr_ptr);
            op_pk_nfd_get(hdr_ptr, "MAC Source", &mac_source);
            op_pk_nfd_get(hdr_ptr, "MAC Destination", &mac_dest);

            op_pk_nfd_get(pkt_arrive, "Payload", &msg_ptr);

            printf("SS%d: REcibiendo paquete NENT de: %d, con DESTINO %d\n", mac_address, mac_source, mac_dest);
            type_nent = 0;
```

```
            op_pk_nfd_get(msg_ptr, "Type", &type_nent);
            op_pk_nfd_get(msg_ptr, "Sponsor Node ID", &mac_sponsor);
            if (op_prg_odb_ltrace_active("pkt_msg"))
                printf("\n\tSS%d*********EL paquete NENT recibido es de tipo: %d, el padre es:%d \n",mac_address,type_nent,mac_parent);

            if (type_nent == NET_ENTRY_REQ){
                op_pk_nfd_get(msg_ptr, "NENT_Request", &nent_req_ptr);
                op_pk_nfd_get(nent_req_ptr, "MAC Address", &mac_address_ch);
                if (op_prg_odb_ltrace_active("pkt_msg")){
                    printf("**************MsS %d: Llegando NENT_ENTRY de MAC. %d\n",mac_address,mac_address_ch);
                    }


        /* se dan de alta los nodos hijos */
                if(!isChild(mac_address_ch))
                    store_sids_child(mac_address_ch,mac_sponsor);
                    print_child_list();

                }else{

                op_pk_nfd_get(msg_ptr, "NENT_Request", &nent_req_ptr);
                    op_pk_nfd_get(nent_req_ptr, "MAC Address", &mac_address_ch);
                    }

                listSize = op_prg_list_size(init_neigh);
                printf("MSS %d:REcibi NENT de hijo: %d, preparando para reenviar a %d de: %d at time %lf\n",
                    mac_address,mac_address_ch,mac_parent, mac_dest, op_sim_time());
                state_node = get_state_node(mac_address_ch);
            if(state_node <= type_nent){

                nent_fwd_ptr = encap_nent_fwd(msg_ptr, mac_sponsor,hdr_ptr, mac_address_ch, nent_req_ptr, msg_mac);
                wait_nent = 1;
                printf("MSS %d:Enviando interruocion de tipo nent: %d en %lf\n",mac_address, type_nent, time_tx_NENT);
                    op_intrpt_schedule_self (time_tx_NENT, NENT_FWD_CODE);
                }else{
                printf("MSS %d state_node %d type_nent %d",mac_address, state_node, type_nent);
                }

                }else{

            hops = num_hops - 1;
            time_tx_NENT = selected_time_NENT( node_serial_id, op_sim_time()+((0.5)), prop_distance);
                printf("MSS %d:2222Enviando interrupcion en tipo_nent %d con wait_nent %d en %lf\n",
                    mac_address, type_nent, wait_nent, time_tx_NENT);

                op_intrpt_schedule_self (time_tx_NENT, NENT_SQUEUE_CODE);

                }
            }
        }

    /** state (NENT_squeue) transition processing **/
    FSM_TRANSIT_FORCE (1, state1_enter_exec, ;, "default", "", "NENT_squeue", "IDLE")
        /*---------------------------------------------------------*/

    /** state (NCFG_fwd) enter executives **/
    FSM_STATE_ENTER_FORCED (7, state7_enter_exec, "NCFG_fwd", "mesh_ss () [NCFG_fwd enter execs]")
        {
        if (op_prg_odb_ltrace_active("state"))
            printf("Entrando NCFG_fwd/Init: %d", mac_address);


        if (op_prg_odb_ltrace_active("pkt_msg")){
            printf("Imprimiendo paquete para enviar\n");
        }

        listSize = op_prg_list_size(competitiveList);

        time_tx_NCFG = op_sim_time();
        wtx = selected_time_NCFG(listSize, mac_address, op_sim_time());

        if (wtx ==1){
            fwd_send = 1;
            if (op_prg_odb_ltrace_active("time_dg")){
                printf("**MSS %d:, NCFG fwd tipo: %d, time: %f\n",mac_address, type_ncfg,op_sim_time());
            }
            op_pk_send (ncfg_rply_ptr,0);

        }else{
            /*Calcula la siguiente oportunidad de transmisión para el mismo frame u otro*/
            time_mx = ConvTO_NCFG(op_sim_time()); // + 0.000068359);
            time_tx_NCFG = ConvTime_NCFG(time_mx);
            op_intrpt_schedule_self(time_tx_NCFG, NCFG_FWD_CODE);
        }

        }
```

```
/** state (NCFG_fwd) transition processing **/
FSM_TRANSIT_FORCE (1, state1_enter_exec, ;, "default", "", "NCFG_fwd", "IDLE")
    /*---------------------------------------------------------*/

/** state (NENT_fwd) enter executives **/
FSM_STATE_ENTER_FORCED (8, state8_enter_exec, "NENT_fwd", "mesh_ss () [NENT_fwd enter execs]")
    {
    if (op_prg_odb_ltrace_active("state"))

        printf("SS %d: Entrando NENT_fwd/Init", mac_address);

        listSize = op_prg_list_size(init_neigh);
        tout_fwd_ncfg = op_sim_time() + 1.0;
        if (op_prg_odb_ltrace_active("time_dg")){
            printf("**MSS %d: ,NENT fwd time: %f\n",mac_address,op_sim_time());
        }
        op_pk_send (nent_fwd_ptr,0);
    }

/** state (NENT_fwd) transition processing **/
FSM_TRANSIT_FORCE (1, state1_enter_exec, ;, "default", "", "NENT_fwd", "IDLE")
    /*---------------------------------------------------------*/



/** state (NENT_arrival) enter executives **/
FSM_STATE_ENTER_FORCED (9, state9_enter_exec, "NENT_arrival", "mesh_ss () [NENT_arrival enter execs]")
    {
    if (op_prg_odb_ltrace_active("state"))
        printf("SS %d: NENT_arrival/Enter\n", mac_address);


    /*********************************************************
         * Figure out which type of message is in the management packet
         *********************************************************/

    op_pk_nfd_get(pkt_arrive, "Header", &hdr_ptr);
    op_pk_nfd_get(hdr_ptr, "MAC Destination", &mac_dest);


    if (mac_dest == mac_address){

        op_pk_nfd_get(msg_ptr, "NENT_Request", &nent_req_ptr);
        op_pk_nfd_get(nent_req_ptr, "MAC Address",&mac );
        op_pk_nfd_get(msg_ptr, "Type",&type );

    if (op_prg_odb_ltrace_active("time_dg"))
        printf("MSS %d: RCV pkt type: %d from %d",mac_address, &mac, &type);

    state_node = get_state_node(mac);

        if(state_node < type){

            pkt_arrive = encap_nent_queue(msg_ptr, hdr_ptr, mac_dest, nent_req_ptr, mac, type);
            op_subq_pk_insert (0, pkt_arrive, OPC_QPOS_TAIL);
            op_subq_print(0);
            count_nent ++;
            nent_queue = 1;

            }
    }else {
        nent_queue = 0;
        if (op_prg_odb_ltrace_active("pkt_msg"))
            printf("MSS %d el paquete NENT es ignorado por que era para la mac %d\n\n",mac_address,mac_dest);
    }
    }


/** state (NENT_arrival) exit executives **/
FSM_STATE_EXIT_FORCED (9, "NENT_arrival", "mesh_ss () [NENT_arrival exit execs]")
    {
    if (op_prg_odb_ltrace_active("state"))
        printf("SS %d:  NENT_Arrival/Exit", mac_address);
    }


/** state (NENT_arrival) transition processing **/
FSM_INIT_COND (nent_squeue)
FSM_TEST_COND (ignore)
FSM_TEST_LOGIC ("NENT_arrival")

FSM_TRANSIT_SWITCH
    {
    FSM_CASE_TRANSIT (0, 6, state6_enter_exec, ;, "nent_squeue", "", "NENT_arrival", "NENT_squeue")
    FSM_CASE_TRANSIT (1, 1, state1_enter_exec, ;, "ignore", "", "NENT_arrival", "IDLE")
```

```
            }
            /*--------------------------------------------------------*/


          }


        FSM_EXIT (0,mesh_ss)
        }
    }

Compcode mesh_ss_init (void ** gen_state_pptr)
    {
    int _block_origin = 0;
    static VosT_Address obtype = OPC_NIL;

    FIN (mesh_ss_init (gen_state_pptr))

    if (obtype == OPC_NIL)
        {
        /* Initialize memory management */
        if (Vos_Catmem_Register ("proc state vars (mesh_ss)",
            sizeof (mesh_ss_state), Vos_Vnop, &obtype) == VOSC_FAILURE)
            {
            FRET (OPC_COMPCODE_FAILURE)
            }
        }

    *gen_state_pptr = Vos_Catmem_Alloc (obtype, 1);
    if (*gen_state_pptr == OPC_NIL)
        {
        FRET (OPC_COMPCODE_FAILURE)
        }
    else
        {
        /* Initialize FSM handling */
        ((mesh_ss_state *)(*gen_state_pptr))->current_block = 0;

        FRET (OPC_COMPCODE_SUCCESS)
        }
    }

void mesh_ss_diag (void)
    {
    /* No Diagnostic Block */
    }


void mesh_ss_terminate (void)
    {
    int _block_origin = __LINE__;

    FIN (mesh_ss_terminate (void))

    Vos_Catmem_Dealloc (pr_state_ptr);

    FOUT;
    }
```

Apéndice D

Algoritmos de selección de tiempo

Código de los algoritmos para la selección de tiempo que se utilizaron en el modelo de OPNET para su comportamiento dinámico. Primero se enlista el código del Algoritmo Mesh Election para el cálculo de transmisión de los mensajes MSH-NCFG y en la segunda parte el algoritmo para la selección de oportunidad de selección para los mensajes MSH-NENT, los cuales fueron utilizados en modalidad de código externo en C++, aunque utilizando bibliotecas de opnet, para que fuera compatible.

D.1. Algoritmo *Mesh Election*

En el archivo mesh_support.h se declararon algunas listas y estructuras globales como es el caso de *NEIGHBOR_INIT_STRUCT* que es la estructura que contiene los parámetros de inicialización que los demás nodos deben conocer, un conjunto de estas estructuras son contenidas en la lista *CompetitiveList* que es la lista dinámica en la que ingresa un nodo, una vez que ya terminó su proceso de ingreso a la red y que ya es un competidor en potencia en espera de los mensajes MSH-NCFG que tenga que re-enviar.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "mesh_support.h"
#include "mesh_gral.h"
#include "opnet.h"
#define TIME_X_SYMB 0.000068359
#define NCFG_FRAME 7

#define NO_FRAMES 10
#define TIME_FRAME 0.01

NEIGHBOR_INIT_STRUCT* dataNeighNCFG;
unsigned long* XmtHoldoffTime;
unsigned long* NextXmtmx;
unsigned long* NextXmtTime1;
unsigned long* NextXmtTime2;
unsigned long* nodelistid;
unsigned long* num_trans_nodo;
unsigned long* TmpXmtTime;

int success;

int selected_time_NCFG ( int listSize, int node_id, double
        realtime){

    unsigned int xmttime=0;

    int i,j;
    unsigned long xmtHoldoffExponent;
```

```
int tam = listSize + 2;
NextXmtmx       = (unsigned long *) malloc(sizeof(unsigned long)*tam);
NextXmtTime1    = (unsigned long *) malloc(sizeof(unsigned long)*tam);
NextXmtTime2    = (unsigned long *) malloc(sizeof(unsigned long)*tam);
nodelistid      = (unsigned long *) malloc(sizeof(unsigned long)*tam);
num_trans_nodo  = (unsigned long *) malloc(sizeof(unsigned long)*tam);
XmtHoldoffTime  = (unsigned long *) malloc(sizeof(unsigned long)*tam);
TmpXmtTime      = (unsigned long *) malloc(sizeof(unsigned long)*tam);

success = 0;

xmttime = ConvTO_NCFG (realtime);

for( i=1; i<= listSize; i++){ ////////////////////// es de 5 bits nada mas
    nodelistid[i]=0;
    XmtHoldoffTime[i]= 0;
    NextXmtTime2[i] = 0;
    NextXmtTime1[i] = 0;

    if(i <=41) NextXmtmx[i] = 7;
    if(i < 35) NextXmtmx[i] = 6;
    if(i < 30) NextXmtmx[i] = 5;
    if(i < 25) NextXmtmx[i] = 4;
    if(i < 20) NextXmtmx[i] = 3;
    if(i < 15) NextXmtmx[i] = 2;
    if(i < 10) NextXmtmx[i] = 1;
    if(i <= 5) NextXmtmx[i] = 0;

    }

for (i = 1; i<= listSize; i++){

    dataNeighNCFG = (NEIGHBOR_INIT_STRUCT *) op_prg_list_access(competitiveList,i-1);
    xmtHoldoffExponent = dataNeighNCFG->xmt_holdoff_exp;
    tmpXmt = xmtHoldoffExponent +4;
    XmtHoldoffTime[i] = pow(2,tmpXmt);

    NextXmtTime1[i] = pow(2,xmtHoldoffExponent)*NextXmtmx[i] +(dataNeighNCFG->next_tx_exp);
    NextXmtTime2[i]= pow(2,xmtHoldoffExponent) *NextXmtmx[i]+(dataNeighNCFG->next_tx_exp)+ xmttime + pow(2,xmtHoldoffExponent);

    tmpXmt = NextXmtTime2[i]+ XmtHoldoffTime[i];
    dataNeighNCFG->earliest_time = tmpXmt;
}

j = 1;

List* init_neigh = competitiveList;

for (i = 1; i<= listSize; i++){

        dataNeighNCFG = (NEIGHBOR_INIT_STRUCT *) op_prg_list_access(init_neigh,i-1);
        xmtHoldoffExponent = dataNeighNCFG->xmt_holdoff_exp;
        tmpXmt = xmtHoldoffExponent +4;
        XmtHoldoffTime[i] = pow(2,tmpXmt);
        if (dataNeighNCFG->earliest_time < xmttime){

            NextXmtTime1[i]= pow(2,xmtHoldoffExponent)*NextXmtmx[i] +xmttime;
            NextXmtTime2[i]= pow(2,xmtHoldoffExponent)*NextXmtmx[i] +xmttime+ + pow(2,xmtHoldoffExponent);
            dataNeighNCFG->earliest_time = NextXmtTime2[i] + XmtHoldoffTime[i];
            dataNeighNCFG->next_tx_exp = xmttime;

        }
            /***** Determina los nodos que contenderan  *******/
        if((xmttime >= NextXmtTime1[i] && xmttime <= NextXmtTime2[i]) || (xmttime == dataNeighNCFG->earliest_time)){
            nodelistid[j] = dataNeighNCFG->nodeID;
            j++;

        }


    }
    //inicia la competencia
    success = 0;
    if (j == 1) {//no hay nodos que compitan
        }
    else if (j == 2){ //solo hay un nodo
        if(node_id == nodelistid[1]){
           success = 1;

            return success;
        }
         //solo hay un nodo en el primer renglon de la lista
        }else{
             if (!elected_NCFG(xmttime, node_id, j, nodelistid)){
                dataNeighNCFG->next_tx_exp =  xmttime ;
```

```
            }else{
                success = 1;
                }


        }
        //hay que quitar el nodo hasta su EarliestSubsequentXmtTime mas cercano
        free(NextXmtmx);
        free(NextXmtTime1);//   = NULL;
        free(NextXmtTime2); //  = NULL;
        free(nodelistid); //        = NULL;
        free(num_trans_nodo); //    = NULL;
        free(XmtHoldoffTime); //    = NULL;
        free(TmpXmtTime); //        = NULL;

    return success;
}

unsigned long inline_smear_NCFG(unsigned long val) {
    val += (val << 12);
    val ^= (val >> 22);
    val += (val << 4);
    val ^= (val >> 9);
    val += (val << 10);
    val ^= (val >> 2);
    val += (val << 7);
    val ^= (val >> 12);
    return val;
}

/* Mesh Election Algorithm */

int elected_NCFG(unsigned long xmttime, unsigned long mynodeid, int
nbc, unsigned long nodelistid[]) {
    int i;
    unsigned long nbr_smear_val, smear_val1, smear_val2;

    smear_val1 = inline_smear_NCFG(mynodeid ^ xmttime);
    smear_val2 = inline_smear_NCFG(mynodeid + xmttime);

    for (i = 1; i <= nbc; i++)
    {
        nbr_smear_val = inline_smear_NCFG(nodelistid[i]^xmttime);
        if (nbr_smear_val > smear_val1)
        {
            return 0;
        }

        else if (nbr_smear_val==smear_val1)
        {
            //printf("El primer desempate\n");

            nbr_smear_val=inline_smear_NCFG(nodelistid[i]+xmttime);
            if (nbr_smear_val > smear_val2)
            {
                return 0;
            }
            else if (nbr_smear_val==smear_val2)
            {
                if ((xmttime % 2 == 0 && nodelistid[i] > mynodeid) || (xmttime % 2 == 1 && nodelistid[i] < mynodeid))
                {

                    printf("Este nodo pierde contra %lu!", nodelistid[i]);
                    return 0;
                }
            }
        }


    }
    return 1;
}

double ConvTime_NCFG (unsigned long xmttime){
    unsigned long x,temp;
    double realtime;
    x = xmttime / NCFG_FRAME;
    temp = xmttime - (NCFG_FRAME * x);
    realtime = x * (0.1) + (temp * TIME_7_SYMB) + TIME_7_SYMB;

    if (temp = 0)  ////////////////////////////////////////
        realtime = x * (0.1) ;
        return realtime;
    }
```

```
unsigned long ConvTO_NCFG (double realtime){

    unsigned long x; //para truncar un valor, i. e. 1.26 [s] seria x = 12.6, se multiplica por 10 segun el procedimiento
    double TO, prueba;  //por eso se asigna un double a un long
    double temp1, temp2;
    unsigned long xmttime = 0;
    TO = 0;
    x = 0;
    temp1,temp2 = 0;
    unsigned long no_mframes; //para truncar un valor, i. e. 1.26 [s] seria x = 12.6, se multiplica por 10 segun el procedimiento
    double time_mframe, time_q;

    time_mframe = NO_FRAMES * TIME_FRAME;
        no_mframes = (unsigned long)(realtime / time_mframe);
        time_q = realtime - no_mframes * (time_mframe);


        if ((0 <= time_q) && (time_q <= TIME_7_SYMB*(NCFG_FRAME-1)) ){

            x = (unsigned long)(time_q / TIME_7_SYMB);
            x = x+1;

        }else{
            no_mframes  = no_mframes + 1;

        }

        xmttime = (no_mframes* 7) + x;

    return xmttime;

}
```

D.2. Algoritmo de selección de la siguiente TO para mensajes MSH-NENT

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "time.h"

#include "mesh_support.h"
#include "opnet.h"

#define TIME_7_SYMB 0.00000976557
#define NENT_FRAME
#define NO_FRAMES 10
#define TIME_FRAME 0.01


double selected_time_NENT ( int node_id, double realtime, double
dist){
    unsigned long TO;
    unsigned long slotTx;
    double time;

    TO = ConvTO_NENT(realtime);
    slotTx = choiceSlotTO(node_id, realtime,TO);
    time = ConvTime_NENT(slotTx + TO);

    //printf("Tiempo para transmitir: %10.8f,TO: %d, el slot ganador es: %d \n",time, TO, slotTx);
    return time;
}

unsigned long choiceSlotTO(int node_id,  unsigned long xmttime,
double dist){


    unsigned long x,z;
    unsigned long temp;
    int y;
    z = 0;
    temp = (xmttime/NENT_FRAME);
    x = xmttime - (temp*NENT_FRAME);
    y = inline_smear_NENT(node_id ^ xmttime);

    srand(xmttime * node_id *dist);
     z = (rand()*y) % 7;
```

```
     z = x + abs(z);

    if (temp == 0)
       z = NENT_FRAME + choiceSlotTO(node_id,NENT_FRAME+xmttime, dist);

    return z;
}


double ConvTime_NENT (unsigned long xmttime){

    double realtime;

    unsigned long cociente, temp;

        cociente = xmttime / (NENT_FRAME);
        temp = xmttime - (NENT_FRAME * cociente);
        realtime = cociente * (0.1) + (temp * TIME_7_SYMB);
        if (temp == 0)
            realtime = cociente * (0.1);
        return realtime;
}


unsigned long ConvTO_NENT (double realtime){

    unsigned long x;

    unsigned long xmttime = 0;
    unsigned long no_mframes;
    double time_mframe, time_q;

    x = 0;

    time_mframe = NO_FRAMES * TIME_FRAME;
    no_mframes = (unsigned long)(realtime / time_mframe);
    time_q = realtime - (no_mframes * (time_mframe));

    if ((0 <= time_q) && (time_q <= TIME_7_SYMB*(NENT_FRAME-1)) ){

        x = (unsigned long)(time_q / TIME_7_SYMB);

    }else{
        no_mframes  = no_mframes + 1;

    }
    xmttime = (no_mframes* NENT_FRAME) + x;
    return xmttime;
}

unsigned long inline_smear_NENT(unsigned long val) {
    val += (val << 12);
    val ^= (val >> 22);
    val += (val << 4);
    val ^= (val >> 9);
    val += (val << 10);
    val ^= (val >> 2);
    val += (val << 7);
    val ^= (val >> 12);
    return val;
}
```

Bibliografía

[1] IEEE Std **802.16-2001** IEEE Standard for Local and Metropolitan area networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems

[2] IEEE **Std 802.16TM-2004** · "Part 16: Air interface for Fixed Broadband Wireless Access System"· The Institute of Electrical and Electronic Engineers

[3] IEEE Std **802.16a** Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed Broadband Wireless Access

[4] IEEE Std **802.16e-2005** Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems

[5] Shetiya, H.; Sharma, V.· **Algorithms for Routing and Centralized Scheduling in IEEE 802.16 Mesh Networks** · Wireless Communications and Networking Conference ·WCNC 2006· IEEE Volúmen 1, páginas 147 - 152 ·2006.

[6] Shetiya, H.; Sharma V.· **Algorithms for Routing and Centralized Scheduling to Provide QoS in IEEE 802.16 Mesh Networks** · Performance ad hoc and IEEE 802 networks table of contents · Montreal, Quebec, Canada · páginas 140-149 · 2005.

[7] Hung-Yu Wei; Ganguly, S.; Izmailov, R.; Haas, Z.J. · **Interference-Aware IEEE 802.16 WiMAX Mesh Network**· Vehicular Technology Conference · VTC 2005-Spring · 2005 IEEE 61st Volume 5, 30 May-1 Jun, páginas 3102-3106 Vol. 5 · 2005.

[8] Min Cao; Raghunathan, V.; Kumar, P.R. · **A Tractable Algorithm for Fair and Efficient Uplink Scheduling of Multi-hop WiMAX Mesh Networks** ·Wireless Mesh Networks · WiMesh 2006 · 2nd IEEE Workshop on 2006, páginas 93-100· 2006.

[9] Min Cao, Wenchao Ma, Qian Zhang, Xiaodong Wang, Wenwu Zhu · **Modelling and Performance Analysis of the Distributed Scheduler in IEEE 802.16 Mesh Mode** · International Symposium on Mobile Ad Hoc Networking & Computing archive Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing table of contents · Urbana-Champaign, IL, USA ·Topology control & mobility table of contents, páginas 78 - 89· ISBN:1-59593-004-3 ·2005.

[10] Abu Ali, N.A.; Taha, A.-E.M.; Hassanein, H.S.; Mouftah, H.T.· **IEEE 802.16 Mesh Schedulers: Issues and Design Challenges**· Network, IEEE Volume 22, Issue 1, Ene.-Feb., páginas 58-65 ·ISSN: 0890-8044 ·2008.

[11] Schwingenschlogl, C.; Dastis, V.; Mogre, P.S.; Hollick, M.; Steinmetz, R.· **Performance Analysis of the Real-time Capabilities of Coordinated Centralized Scheduling in 802.16 Mesh Mode** · Vehicular Technology Conference · VTC 2006-Spring · IEEE 63rd Volume 3, páginas 1241-1245 · 2006.

[12] Shie-Yuan Wang; Chih-Che Lin; Ku-Han Fang; Tens-Wei Hsu ·**Facilitating the Network Entry and Link Establishment Processes of IEEE 802.16 Mesh Networks** · Wireless Communications and Networking Conference ·WCNC 2007 ·IEEE 11-15 Marzo 2007, páginas 1842 - 1847 ·2007

[13] Caballero Jose Manuel · **Redes de banda ancha** · ISBN: 978-9701502663 · Alfaomega ·1998.

[14] Editado por Syed Ahson; Mohammad Ilyas · **WiMAX Handbook - Building 802-16 Wireless Networks** · ISBN: ISBN-13: 978-0071454018 · McGraw-Hill · 2005.

[15] Jeffrey G. Andrews; Arunabha Ghosh; Rias Muhamed · **Fundamentals of WiMAX** · Understading Broadband Wireless Networking ·ISBN: 978-0132225526 ·Prentice Hall · 2007.

[16] Nuaymi Loutfi ·**WiMAX Technology for Broadband Wireless Access** · ISBN: 978-0470028087 ·Wiley · 2007.

[17] Editado por Syed Ahson; Mohammad Ilyas · **WiMAX Technologies, Performance Analysis, and QoS** · ISBN: 978-1420045253 · CRC Press · 2008.

[18] Godínez González Antonio; Siten Salgado Roberto· **Análisis del Comportamiento Dinámico del Protocolo IEEE 802.16 (WiMAX)** ·Tesis · 2005

[19] Página oficial de ETSI (European Telecommunications Standards Institute). Resumen del proyecto de estandarización BRAN (Broadband Radio Access Networks) ·http://portal.etsi.org/bran/Summary.asp

[20] Página de especificaciones DOCSIS · http://www.cablemodem.com

[21] Página oficial de estandares ITU (International Telecommunication Union) · http://www.itu.int

[22] Página del Consorcio Internacional de Ingenieros, especificaciòn de LMDS(Local Multipoint Distribution System) ·http://www.iec.org/online/tutorials/lmds/