UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

"ANÁLISIS Y MODELADO DEL COMPORTAMIENTO DE REDES MESH EN BASE A TECNOLOGÍA WIMAX"

T E S I S

QUE PARA OBTENER EL GRADO DE:

INGENIERO EN TELECOMUNICACIONES

P R E S E N T A N:

FERNANDO NAVARRO ESTRADA
SERGIO SALAS FRANCISCO

DIRECTOR DE TESIS:
DR. VÍCTOR RANGEL LICEA

CIUDAD UNIVERSITARIA, MÉXICO D.F. **AGOSTO, 2008**

A mis padres:

Salvador Navarro Romero y Alicia Estrada Flores,

por el amor, cariño y apoyo que me han brindado,

con profundo agradecimiento.

A mis hermanos y familia:

que siempre han estado a mi lado para ayudarme

A mis amigos:

que me han acompañado por mi camino y de quienes he aprendido mucho.

Muchas gracias.

Fernando.

A mis padres:

Mario Salas Chávez y Gudelia Francisco Osorio,

que me apoyaron a lo largo de mi carrera con su amor y comprensión.

A mis hermanas:

Quienes me han dado su apoyo en los buenos y malos momentos.

A mis amigos:

Quienes me han acompañado y apoyado durante la licenciatura.

Muchas gracias.

Sergio

Agradecimientos

A la Universidad Nacional Autónoma de México por brindarnos la oportunidad de formar parte de la comunidad Universitaria y de la máxima casa de estudios.

Al Dr. Victor Rangel, por abrirnos las puertas del laboratorio de redes, por brindarnos su amable asesoría y sus conocimientos sobre el tema de WiMAX para la realización de este trabajo.

A la DGAPA por el apoyo recibido a través del proyecto PAPIIT 1N-104907-3.

Un sincero agradecimiento a mi compañero por sus sugerencias e ideas durante el desarrollo de este trabajo.

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1 Definición del Problema

Actualmente, las necesidades de comunicación requieren de soluciones que integren en un solo servicio la capacidad de enviar distintos tipos de información de diversa índole, tales como voz, datos y video. Entre las distintas opciones que existen, el canal inalámbrico es una de las opciones más atractivas ya que emplea un medio que es universalmente accesible y resulta una excelente opción para las áreas donde, por sus condiciones geográficas, resulte elevado el costo de instalación de algún medio alámbrico.

Existen diversos medios alámbricos, tales como el cable coaxial, fibra óptica y xDSL; por los cuales los hogares y negocios tienen un acceso de última milla. La desventaja de éstos consiste en que se invierte demasiado tiempo en la instalación de su infraestructura, además de ser demasiado costosa y no proveer ninguna flexibilidad.

Una de las tecnologías que ha tomado fuerza en los años recientes y que su objetivo es poder cumplir con toda las expectativas que los usuarios finales requieren, es decir, una gran capacidad de recepción y transmisión de datos, así como un área de cobertura extensa, es la tecnología *Broadband Wireless Access* (BWA).

Dos estándares diferentes son los que rigen la tecnología BWA en la actualidad:

- *ETSI BRAN (European Telecommunications Standards Institute Broadband Radio Access Networks).*
- El estándar *IEEE 802.16.*

El estándar *IEEE 802.16* presenta características mucho más robustas que el protocolo *ETSI BRAN*, ya que se derivó del protocolo americano usado para la transmisión de datos de la televisión por cable *DOCSIS*. El estándar *IEEE 802.16* es mejor conocido como *WiMAX* (del inglés *Worldwide Interoperability for Microwave Access, "Interoperabilidad Mundial para Acceso por Microondas*"). Cuenta con seis diferentes tipos de reservación que son conocidos como QoS (calidad de servicio) y con diferentes tipos de modulación para la transmisión de diferentes servicios en tiempo real. WiMAX es una solución que por sus características es más conveniente para los usuarios y para los operadores. Gracias a las capacidades que presenta, se está creando un mercado masivo para soluciones inalámbricas, tanto para el hogar como para los negocios.

En la figura 1.1 podemos ver la clasificación de los estándares americano y europeo.

Existen dos versiones diferentes para la red HiperLAN, las dos trabajan en la frecuencia de 5 GHz y tienen movilidad limitada (local), la diferencia es que una tiene velocidades de transmisión de 20 Mbps y es compatible

con redes Ethernet y Token Ring; y la otra tiene velocidades de transmisión de 54 Mbps, es compatible con redes de banda ancha basadas en IP y ATM y tiene mecanismo para el soporte de QoS. Por otro lado, la HiperACCESS se utiliza para acceso residencial (fijo) y tiene velocidades de transmisión de 25 Mbps. Finalmente la última jerarquía se le conoce con el nombre de HiperLink y trabaja a frecuencias de 17 GHz, tiene un acceso fijo y alcanza velocidades de transmisión de 155 Mbps.

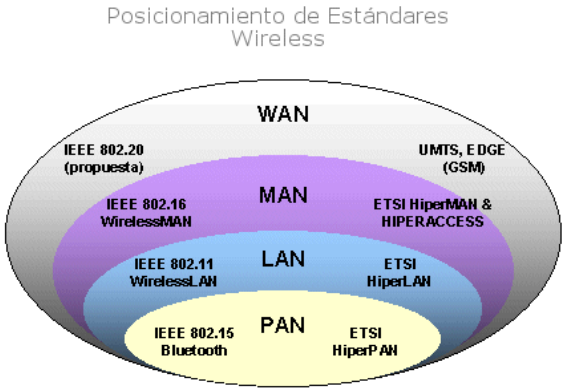


Figura 1.1. Clasificación de estándares.

En la actualidad existen tres versiones del estándar 802.16. En la primer versión del estándar se estableció el rango de frecuencia de operación de 2 – 60 GHz, además el enlace de las estaciones suscriptoras con la estación base era de línea de vista (LOS) para intercambiar información. Esta limitación elevaba el costo y reducía la cobertura del servicio, por lo que años después salió la segunda versión, 802.16 – 2004, que solucionó lo anterior, ya que la nueva versión soporta enlaces sin línea de vista (NLOS) porque se trabaja en el rango de frecuencias de 2 – 11 GHz y como resultado de lo anterior el rango de cobertura es mayor. Las dos versiones anteriores son específicamente para dispositivos fijos. La versión más reciente, 802.16e, se enfoca para tener movilidad en los dispositivos finales similar al funcionamiento de la tecnología celular. Las velocidades de transmisión del protocolo IEEE 802.16 son de entre los 32Mbps y 134Mbps aproximadamente con un ancho de banda de 28 MHz (en la banda de 10 a 66 GHz), mientras que el 802.16a puede llegar a los 70 Mbit/s, operando en un rango de frecuencias más bajo (<11 GHz).

	WiMAX 802.16	WiFi 802.11	MBWA 802.20	UMTS y CDMA2000
Velocidad	124 Mbit/s	11-54 Mbit/s	16 Mbit/s	2 Mbit/s
Cobertura	40-70 km	300 m	20 km	10 km
Licencia	Si/No	No	Si	Si
Ventajas	Velocidad y Alcance	Velocidad y Precio	Velocidad y Movilidad	Rango y Movilidad

Tabla 1.1. Comparación del estándar 802.16 frente a otras tecnologías

Las redes que utilizan el protocolo IEEE 802.16 consisten de una estación base y múltiples estaciones suscriptoras. La estación base (*BS*) funciona como Gateway hacia la red externa para las estaciones suscriptoras (*SS*). El protocolo soporta dos modos de operación: punto a multipunto (PMP) y en modo Mesh. En la modalidad PMP cada estación suscriptora se comunica con la estación base a través de un enlace simple, que requiere que todas las estaciones suscriptoras tengan línea de vista con la estación base. En contraste, en el modo Mesh, las estaciones suscriptoras pueden comunicarse con la estación base y con otras estaciones suscriptoras a través de múltiples rutas.

Con la topología Mesh no sólo se extiende la cobertura de la red y la capacidad en ambientes de NLOS, además provee mayor respuesta de la red si algún nodo o enlace falla o cuando las condiciones en algún canal no son las adecuadas. Con la topología Mesh es posible permitir a otros dispositivos que estén fuera del rango de cobertura, por medio de los nodos que sí se encuentren dentro del rango de cobertura.

En la tabla 1.2 podemos comparar algunas de las características de las diferentes versiones del estándar 802.16.

Este tipo de topología es adecuada en zonas rurales con poca densidad de población, ya que para este caso las estaciones suscriptoras primarias actúan como estaciones base para poder darle servicio a las estaciones secundarias y así lograr una amplia cobertura.

Para apoyar y promover el estándar se fundó en el año 2003 un proyecto titulado WiMAX Forum el cual tiene como objetivos principales promover el estándar 802.16, la certificación de los equipos y promover la interoperabilidad de WiMAX entre diferentes marcas para soluciones de última milla. Actualmente el foro WiMAX integra a más de 400 miembros entre los cuales figuran fabricantes de chip, fabricantes de equipos y prestadores de servicios.

	802.16	**802.16a y 802.16d**	**802.16e**
Espectro	10 – 66 GHz	< 11 GHz	< 6 GHz
Condiciones del Canal	Línea de vista solamente	No se necesita línea de vista	No se necesita línea de vista
Bit rate	32 – 134 Mbps con canales de 28 MHz	Arriba de 75 Mbps con canales de 20 MHz	Arriba de 15 Mbps con canales de 5 MHz
Modulación	QPSK, 16QAM y 64 QAM	OFDMA con 256 subportadoras, QPSK, 16QAM, 64 QAM	OFDMA Escalable
Movilidad	Fijo	Fijo	Movilidad regional
Ancho de banda del canal	20, 25 y 28 MHz	Canales ajustables de 1.25 MHz a 20 MHz.	Similar al 802.16a con subcanales de subida
Rango típico de celda	1 – 3 millas	3 – 5 millas. Alcance máximo de 30 millas con LOS	1-3 millas

Tabla 1.2. Estándares 802.16

Para la elaboración de este proyecto nos basaremos en el estándar 802.16 – 2004, nos define la interfaz física y la capa MAC (Medium Access Control), específicamente nos centraremos en la capa MAC para el acceso a sistemas de banda ancha donde emplearemos la topología Mesh.

Esta tesis forma parte del proyecto PAPIIT: “Diseño de técnicas de mejoramiento de capacidad de redes inalámbricas de banda ancha tipo Mesh”. Con este proyecto se busca realizar un modelo base para el estudio de las redes Mesh utilizando el programa de simulación OPNET, que nos proporciona las herramientas necesarias para lograr cumplir los objetivos. Además el proyecto servirá como pilar de nuevas tesis que pretendan profundizar en el comportamiento de las redes Mesh.

1.2 Modelado de Redes de Comunicaciones

El acceso de banda ancha para cubrir los accesos de ultima milla surgió de la necesidad de tener nuevos medios y tecnologías para poder transmitir la información de una forma más eficiente y para poderla manejar de una manera más sencilla, lo que implica en tener cantidades mayores de información en menos tiempo. Entre algunas de estas tecnologías tenemos el DSL (*Digital Subscriber Line*) la cual provee una velocidad de 2 Mbps, para el canal de subida y para el canal de bajada. A su vez, ADSL *(Asymmetric Digital Subscriber Line)* ofrece velocidades entre 6 y 8 Mbps para el canal de bajada y cientos de kilobits por segundo para el canal de subida. Otra de las tecnologías desarrolladas fue el cable coaxial, que fue utilizado al principio para la transmisión de televisión por cable y ahora nos brinda servicio de datos a alta velocidad y telefonía. Estudios realizados en el 2004 indican que cerca de una cuarta parte de la población adulta tienen o usan servicios de banda ancha y la mayoría de estos son servicios a través de enlaces E1, cable o DSL.

Actualmente el éxito del estándar 802.11 (WiFi) nos muestra que el mercado tiene una necesidad creciente de un acceso de banda ancha inalámbrico que tenga más velocidad y más distancia, y que sea utilizado tanto en el hogar como en el mercado empresarial.

Por sus características, las redes BWA están surgiendo como una alternativa a las tecnologías mencionadas anteriormente. Entre ellas su fácil y rápida implementación al utilizar el espacio libre como medio de propagación, su escalabilidad al no tener que instalar nueva infraestructura y bajos costos de mantenimiento y actualización ya que sólo hay que dárselos a los equipos terminales.

El estándar IEEE 802.16 – 2004 usa el esquema TDMA (*Time Division Multiple Access*) para el canal de subida (de la SS a la BS) donde este canal está dividido en frames. Al mismo tiempo, cada frame está dividido en *time slots* que pueden ser asignados para la BS o para las diferentes SS. Por otro lado, el canal de bajada (de la SS a la BS) utiliza la técnica TDM (*Time Division Multiplexing*).

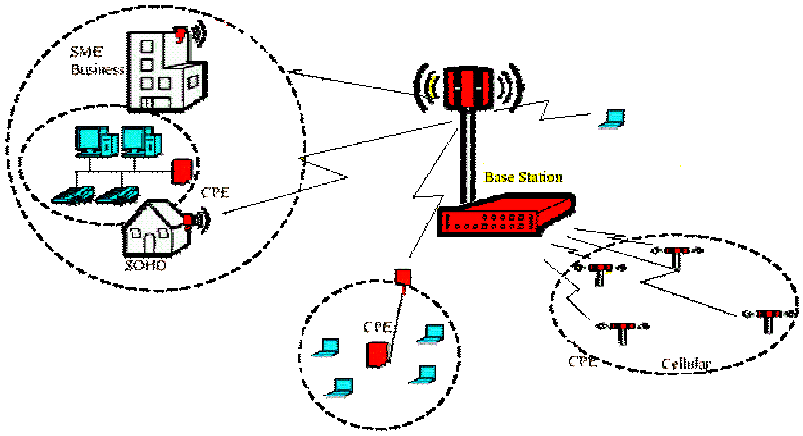


Figura 1.2. Red de Acceso Inalámbrico de Banda Ancha.

Para que un usuario sea capaz de poder transmitir información a través de la red es necesario que envíe una petición para poder realizar una reservación a la BS, este procedimiento es realizado a través de un módem inalámbrico. En este mensaje de reservación hay que señalar que la estación suscriptora indica cuál es el número de slots que requiere para poder enviar información. Este procedimiento se detallará más adelante.

1.3 Investigación en el área de modelado de redes Mesh

En el diseño de una red inalámbrica no siempre es posible tener una línea de vista entre las estaciones suscriptoras y la estación base, ya sea por cuestión de recursos financieros o por que físicamente es imposible realizarlo. Con el estándar IEEE 802.16 – 2004 se puede hacer realidad el hecho de tener redes inalámbricas creadas fácil y rápidamente con un rango de alcance mucho mayor que si tuviéramos una configuración Punto a Multipunto (PMP). No solo es posible extender el rango de operación de la red sino que se elimina la necesidad de tener forzosamente línea de vista para poderse comunicarse con la estación base.

En la figura 1.3 se muestra un posible esquema para poder utilizar la configuración Mesh descrita anteriormente. Se puede observar que la estación base está rodeada de múltiples estaciones suscriptoras, algunas de ellas no tienen un rango de vista directo hacia la BS, lo que produciría que no sea posible que algunas SS se puedan comunicar utilizando la configuración PMP.

Sin embargo, si se utiliza la configuración Mesh es posible que se establezca una comunicación entre estaciones vecinas para poder brindar acceso a la red a las estaciones que se encuentran más lejanas. Por esta razón se incrementa la capacidad de cobertura que la red nos puede brindar.

Un nodo *vecino* se define en la configuración Mesh como un nodo que se encuentra exactamente a un salto de distancia del nodo de interés. A su vez, un conjunto de nodos vecinos se le conoce como *vecindad*. Una *vecindad extendida* incluye a todos los nodos que se encuentren a dos o tres saltos, depende de cómo se tome este parámetro.

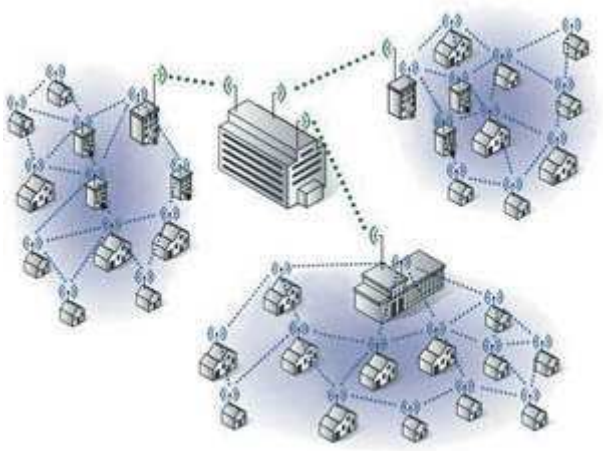


Figura 1.3. Configuración Mesh

Un nodo huésped se define como aquel nodo vecino que se encarga de validar y controlar el tráfico y las transmisiones que van de la BS a un nodo vecino fuera del alcance de la BS. Se asume que estos nodos, por su función, son los que se encuentran más cerca de la BS y por tanto cuentan con condiciones más favorables para poder comunicarse directamente con la BS.

Una diferencia importante entre el estándar que nos describe sólo la configuración PMP y el estándar IEEE 802.16d es que solo se soporta *TDD (Time Division Duplex)* para el canal de subida y el de bajada. De esta forma la BS se debe de considerar como si fuera otro nodo para poder transmitir y para no tener ninguna colisión con las SS se emplean algoritmos.

El estándar para el modo Mesh describe dos formas de planeación (*scheduling)*, de forma centralizada y de forma descentralizada. Si se utiliza la forma centralizada la BS será la encargada de controlar todos los accesos al canal aunque los nodos se encuentren a más de una salto de distancia. Además si una SS hace una petición de ancho de banda será necesario que primero llegue a la BS y ésta autorice según la disponibilidad de la red. Si se utiliza la forma descentralizada entonces las SS establecen comunicaciones entre ellas mismas para lograr negociar la transmisión y el ancho de banda. Además la forma descentralizada está dividida en dos modos de operación, coordinado y descoordinado. Los tres esquemas anteriores pueden coexistir en una misma red al mismo tiempo, lo que implica que en un *frame* vienen mensajes de control para los tres esquemas.

1.4 Sistemas de tercera generación 3G

El propósito de la tercera generación consiste en superar las técnicas de las tecnologías precedentes. El propósito de la tercera generación es unificar la voz y datos con acceso inalámbrico a Internet, además de manejar aplicaciones multimedia y utilizar altas tasas de transmisiones de datos. El impulso de los estándares de la 3G está siendo apoyando por la Unión Internacional de Telecomunicaciones ITU “International Telecomunications Union”, conocida como IMT-2000.

Los principales requerimientos para esta tecnología incluyen:

- Calidad de voz comparable a la que ofrece una red telefónica pública (PSTN).
- Velocidades de transmisión de datos de 144 kb/s para usuarios en vehículos en movimiento viajando a una velocidad de 120 Km/h en ambientes exteriores.
- Velocidades de transmisión de datos de 384 kb/s para peatones, que se encuentren en un solo lugar o bien moviéndose sobre áreas pequeñas.
- Soporte para operaciones de 2.048 Mb/s en oficinas, es decir en ambientes estacionarios de corto alcance o en interiores.
- Soporte para ambos servicios de datos conmutación por paquetes y conmutación por circuitos.
- Mayor eficiencia del espectro disponible.
- Introducción flexible a los nuevos servicios y tecnologías.

Entre las tecnologías contendientes de la tercera generación se encuentran UMTS “Universal Mobile Telephone Service”, CDMA2000, IMT-2000, ARIB (3GPP), UWC-136, entre otras.

UMTS

UMTS (*Universal Mobile Telecommunication System*) es una tecnología para voz y datos a alta velocidad. La tecnología usada es WCDMA (*Wideband Code Division Multiple Access*) que permite alcanzar velocidades de transmisión de hasta 2 Mbps. Cuando se utiliza esta tecnología se hace un uso más eficiente del espectro ya que utiliza una combinación de las tecnologías CDMA y TDMA. Está siendo desarrollado por 3GPP (3rd Generation Partnership Project), un proyecto común en el que colaboran: ETSI (Europa), ARIB/TTC (Japón), ANSI T-1 (USA), TTA (Korea), CWTS (China).

Entre las características que posee UMTS está la de ser más barato para poder incrementar el número de usuarios. Además proporcionará nuevos servicios y servicios de mayor calidad con tendencia al crecimiento de los servicios multimedia. El acceso rápido es otra de sus características, se pueden alcanzar hasta 144 Kbps en vehículos a gran velocidad, 384 Kbps en espacios abiertos y 1.72 Mbps con baja movilidad. Esto se combina al soporte del protocolo IP para prestar servicios multimedia interactivos y aplicaciones de banda ancha.

HSDPA

El HSDPA (*High Speed Downlink Packet Access*) es la primer evolución de 3G conocida como 3.5 G. Es una tecnología que puede dar una velocidad de transmisión de datos de 14 Mbps teórica y soportar tasas de *throughput* cercanas a 1 Mbps. De forma similar a la que EDGE incrementa la capacidad de GSM, HSPDA incrementa la capacidad espectral de UMTS hasta 3.5 veces, con esto es posible crecer en velocidad de transmisión y en la capacidad de usuarios que soporta la red. Para lograr esto se utiliza una modulación de mayor orden que es la 16 QAM, codificación variable de errores y redundancia incremental. Además se enfatiza un mecanismo de programación para determinar qué usuario obtendrá recursos para transmitir.

HSUPA

La tecnología HSUPA (High-Speed Uplink Packet Access) es una mejora de HSDPA y es calificado como 3.75G y su objetivo es incrementar la velocidad para el canal de subida UMTS/WCDMA. Está definido por 3GPP como una tecnología que ofrece una mejora sustancial en la trama de subida, desde la terminal a la estación base.

Ofrece prestaciones de alta capacidad para voz y datos y se enfoca en la creación de un mercado enfocado a aplicaciones multimedia inalámbricas. Algunas serán nuevas y otras se beneficiarán del aumento de ancho de banda.

1.5 Objetivos y Contribuciones

Principalmente la trama para *IEEE 802.16* Mesh está formada por una subtrama de control y otra de datos; sin embargo, la subtrama de control se subdivide en una subtrama de control de red y de control de “*scheduling*”. En la subtrama de control de red se manejan dos tipos de mensajes llamados, *MSH-NCFG (Mesh Network Configurations) y MSH-NENT (Mesh Network Entries).*

Esta tesis forma parte del proyecto PAPIIT número 1N-104907-3: “Diseño de técnicas de mejoramiento de capacidad de redes inalámbricas de banda ancha tipo Mesh”. Como contribución al proyecto, se estudiará el comportamiento del algoritmo de selección mencionado posteriormente para el envió de mensajes de control.

El principal objetivo de esta tesis es implementar el algoritmo de elección para los mensajes *MSH-NCFG* y los mensajes *MSH-NENT* en la subtrama de control de red, para la etapa de registro entre las estaciones suscriptoras y la estación base, con el fin de lograr el registro de un gran número de estaciones suscriptoras sin tener colisiones en el envío de los mensajes, y después que los nodos empiecen a enviar tráfico.

Con el algoritmo de elección implementado, se podrá analizar el comportamiento de distintos parámetros que intervienen en el tiempo de registro de un nodo en la red Mesh y determinar los mejores y peores parámetros dependiendo de las características deseadas en la red.

Con lo anterior se pretende implementar las bases para futuros estudios en el modelo de redes WiMAX bajo la configuración Mesh. Además, lograr que esta línea de investigación siga avanzando para obtener un modelo general sobre el funcionamiento de las redes Mesh. Al finalizar este modelo se puede implementar físicamente para que comunidades rurales puedan tener acceso a Internet o a otros servicios, permitiendo que lugares remotamente alejados de las ciudades tengan comunicación.

1.6 Estructura de la Tesis

Esta tesis se divide en 6 capítulos, en donde el primer capítulo es una breve introducción al tema a desarrollar donde se brinda el panorama general de las redes BWA y de las redes Mesh.

En el capítulo 2, se da la descripción general de la tecnología Mesh en cuanto a su arquitectura, protocolos y estándares. Además se hace una comparación entre la tecnología Mesh y tecnologías alternativas.

En el capítulo 3, se hace una descripción general de la estructura del frame utilizado en el estándar IEEE 802.16 – 2004 así como una descripción detallada de la estructura de la trama de control y de los distintos mensajes que existen para que un nodo pueda darse de alta en la red Mesh.

En el capítulo 4 encontraremos el algoritmo descrito en el estándar y la forma en que se implementó para su análisis. Se describe el comportamiento de cada una de las variables que intervienen y de la forma en que se pretende elegir los valores óptimos para alcanzar tiempos de registro lo menor posible. Se brinda un panorama general de las funciones que interviene en la programación del algoritmo de elección ya que estas funciones nos ayudarán a llegar al modelo final del proceso.

En el capítulo 5 encontraremos el modelo final de registro de nodos en la red Mesh en donde se incluye el algoritmo utilizado en el capítulo anterior. Se analizarán a detalle el número de mensajes que se tienen que transmitir y las veces que el algoritmo es utilizado, para las veces que no sea necesario, se hará una propuesta para transmisión de mensajes de control. Se analizará a fondo los tiempos de registro y cómo es que intervienen los distintos parámetros para incrementar o decrementar estos tiempos.

Finalmente, en el capítulo 6 se presentarán las conclusiones del modelo y del algoritmo desarrollado y se hablará de las principales utilidades de este análisis y cómo puede ser aprovechado para su trabajo futuro.

Capítulo 2

Descripción del Estándar IEEE 802.16

2.1 Introducción

Los servicios y las características que soporta la tecnología de acceso inalámbrico de banda ancha (BWA) se definen en el protocolo IEEE.802.16 dentro de la capa de control de acceso al medio (MAC) y la capa física (PHY). Dentro de estas dos capas se contempla la posibilidad de ofrecer distintos niveles de servicio para el transporte de información, esto implica el retraso que va a sufrir ésta y el ancho de banda disponible. Para lograr esto es necesario utilizar diversas técnicas que permitan tener prioridades para cada conexión que se realiza pero que, además, permita tener transmisión efectiva para todas las conexiones. El protocolo cuenta la característica de proporcionar, por medio del funcionamiento de la capa MAC, las prioridades de conexión para poder establecer estas calidades de servicio.

Para lograr esto existe un envió constante de mensajes de control que intercambian la estación base y las estaciones suscriptoras. De esta forma es posible reservar el ancho de banda que cada estación suscriptora va a utilizar. Las estaciones suscriptoras utilizan tiempos específicos donde intentan realizar a través de un algoritmo esa solicitud a la estación base para poder ganar la reservación, el período de tiempo en que pueden enviar las solicitudes se le conoce como *tiempo de contención*. Una vez que estas solicitudes llegan a la estación base, ésta asigna el tiempo en que las estaciones suscriptoras pueden transmitir datos y de esta forma evitar que existan colisiones en la red. Y para poder tener diferentes QoS la estación base asigna, dependiendo de diferentes criterios, este tiempo de transmisión.

Existen otras características que permiten hacer del estándar un protocolo de comunicaciones más robusto y versátil. Un algoritmo de resolución de colisiones basado en el algoritmo exponencial Back-off, que se encarga de reducir los periodos de contención desperdiciados por el envió constante de paquetes de información o de mensajes solicitando un ancho de banda desde equipos distintos. Además, existen las solicitudes *Piggyback* que optimizan el uso del canal inalámbrico que básicamente son solicitudes de oportunidades de transmisión que están dentro de un periodo de transmisión de información y no dentro del periodo de contención.

Entre la capa PHY y la capa MAC existe una capa que se encarga de la convergencia de la transmisión. Esta capa se encarga de la transformación de *PDUs (Protocol Data Units)* Mac en bloques fijos. En caso de que exista algún error irrecuperable con esta capa se puede sincronizar el siguiente MAC PDU.

2.2 Evolución del estándar 802.16

La IEEE ha establecido un conjunto de estándares para comunicaciones inalámbricas, las principales, o más conocidas son: el estándar IEEE 802.15 [1], conocido como *Bluetooth*, el cual se refieren a redes de área personal (PAN); El estándar IEEE 802.11 [2], conocido como WiFi, para redes de área local (LAN) y por último el estándar IEEE 802.16 [3] para redes de área metropolitana (MAN).

La primer versión del estándar IEEE 802.16 – 2001 fue aprobada en diciembre del 2001 y publicada en el año 2002. El estándar IEEE 802.16 se trata de una especificación para las redes de acceso metropolitanas inalámbricas de banda ancha *(Wireless MAN)* [4] punto multipunto. Además la banda de frecuencias en la cual trabaja se encuentra entre 10 – 66 GHz por lo que se requiere línea de vista (LOS) por trabajar en frecuencias altas. El tipo de modulación utilizado fue QPSK, 16QAM y 64 QAM.

La primer revisión del estándar fue en el año 2003 y se nombro IEEE 802.16a, en la cual se introdujeron mejoras para obtener mayores beneficios. Algunas de las mejoras fueron, que la banda de operación se redujo de 2 – 11 GHz, por lo que ahora no era necesario la línea de vista (NLOS) y como consecuencia se logró tener alcances de 50 Km con una tasa de transferencia de 70 Mbps. Además se mejoró la capa física (PHY) para introducir las modulaciones OFDM y OFDMA.

La última revisión del estándar fue IEEE 802.16 – 2004 [5], este estándar es el que se utiliza en la actualidad con las mismas características que la revisión del año 2003, la única diferencia es que en esta última revisión se introdujo el soporte para redes Mesh. Es de importancia recalcar que el estándar IEEE 802.16 del 2001 – 2004 fueron realizadas para sistemas fijos, tanto para sistemas punto a multipunto (PMP) y Mesh.

Como necesidad de contar con un sistema que diera acceso a redes inalámbricos de banda ancha a dispositivos móviles se creó un nuevo estándar llamado IEEE 802.16e – 2005 [6], aprobada en diciembre de 2005. Las características principales del estándar son que la banda de operación debe ser menor a 6 GHz, por tratarse de un estándar que dará acceso a dispositivos móviles no se requiere de línea de vista (NLOS) y la cobertura es de 2 a 5 Km.

2.3 Operación del protocolo IEEE 802.16

2.3.1 Control de acceso al medio

El estándar IEE 802.16 proporciona un acceso al medio inalámbrico de tal forma que se pueda compartir con cientos de usuarios. Para proveer de estas características se usa TDD y FDD ya que son soportadas por el estándar. Cuando se utiliza FDD se puede tener una comunicación full-duplex y half-duplex.

El canal de subida utiliza una combinación de las técnicas TDMA (T*ime Division Multiple Access*) y DAMA (Acceso Múltiple por Asignación de Demanda), este canal es dividido en múltiples timeslots. El número de slots usados para la región de contención, para registro, de guarda o bien para el tráfico de datos es controlado por la capa MAC por la estación base y puede variar en el tiempo para tener un rendimiento óptimo. El canal de bajada utiliza TDM con la información de cada estación suscriptora multiplexada en un solo frame de datos y recibida por todas las estaciones suscriptoras al mismo tiempo y en el mismo sector.

2.3.2 Direccionamiento y tipo de conexiones

Para el caso de las redes Mesh al igual que para el caso de la configuración PMP, cada nodo debe de tener una dirección universal MAC que consta de 48 bits, tal como se definió en el estándar IEE 802.16 – 2001. La función de esta dirección es sólo identificarla entre todos los dispositivos de todos los posibles tipos y vendedores. La dirección MAC es usada durante el procedimiento de acceso a la red y como parte del proceso de autorización por el cual se verifica la identidad de cada nodo.

Cuando un nodo ha sido autorizado a entrar a la red éste recibe un identificador que consta de 16 bits llamado *NodeID* por parte del la Mesh BS. El *NodeID* es la base para identificar nodos durante la operación normal de la red. El *NodeID* es transmitido en mensajes unicast y broadcast a la red.

Para el direccionamiento de nodos en una vecindad se usa un identificador llamado *LinkID* y consta de 8 bits. Para cada enlace que se crea entre nodo y nodo vecinos se le asigna un *LinkID* distinto. Cuando se intenta establecer un enlace entre nodo y nodo el *LinkID* es transmitido como parte CID (*Connection ID*) y dentro de la cabecera MAC en mensajes unicast. El *LinkID* se usa dentro de un esquema *distributed scheduling* para identificar los recursos entregados y solicitados. Cuando los mensajes son entregados en forma de *broadcast* los nodos receptores de este mensaje pueden identificar la forma en que deben de transmitir usando el *NodeID* en la subcabecera Mesh y el *LinkID* dentro del mensaje MSH-DSCH (Mesh Mode Schedule with Distributed Scheduling). De esta forma cada nodo sabe cuando se le llama para poder transmitir.

Cuando se va a inicializar los servicios, la estación base proporciona tres diferentes tipos de conexiones en ambas direcciones utilizado los mensajes *RNG-REQ (Ranging Request)* y *REG-RSP (Registration Response).* Cada una de estas conexiones cuenta con diferentes QoS.

La primera es la conexión básica (*Basic Connection ID*) y es utilizada para mensajes cortos y urgentes de control MAC. La segunda es la conexión primaria (Primary Management Connection) que soporta mensajes más largos de control MAC y agrega una mayor tolerancia a la latencia. Finalmente, la conexión secundaria (Secondary Management Connection) está hecho para la transmisión de mensajes de control con mayor tolerancia a los retardos, está basado en estándares como *DHCP* (*Dynamic Host Configuration Protocol*), *TFTP* (*Trivial File Transfer Protocol*), *SNMP* (*Simple Network Management Protocol*), etc.

La función que tiene el Connection ID es apuntar hacia el destino de cada conexión, puede considerarse similar a la IP. El uso de un CID de 16 bits permite un total de 64 000 conexiones para el canal de subida y bajada en una red. Para hacer solicitudes de transmisión se utiliza el CID y cada vez que se realiza una conexión diferente se tendrá un ancho de banda distinto. Muchas sesiones de capas superiores pueden utilizar el mismo CID.

El Connection ID en el modo Mesh lo podemos encontrar detallado en el estándar.

2.3.3 MAC Protocol Data Unit

Es la unidad básica de comunicación que es intercambiada por las capas MAC de las estaciones suscriptoras y de la estación base. Como se puede observar en la figura 2.1 está dividida en tres campos, el primero es el encabezado genérico (*Generic MAC Header*) y tiene un tamaño fijo. El segundo campo es la carga útil (*Payload*) que puede tener subencabezados como el MAC SDU, este campo es opcional y puede tener una longitud variable. El tercer campo es el CRC (*Cyclic Redundancy Check*) que nos ayuda a detectar errores y es opcional.

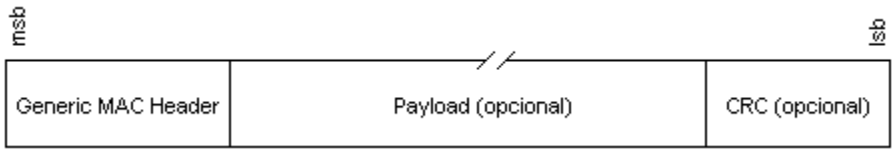


Figura 2.1. MAC PDU

2.3.4 Paquetes de control MAC

Tres formatos de cabecera MAC están definidos en el estándar. El primero es el encabezado genérico MAC, que está constituido en parte por el encabezado MAC SDU, se le conoce como Encabezado Genérico *GMH (Generic MAC Header)*. El segundo es utilizando para solicitud de ancho de banda y su nombre es Encabezado de Solicitud de Ancho de Banda *BRH (Bandwidth Request Header).* El tercero es para administración y los encabezados son incluidos dentro del Payload del MAC PDU.

Encabezado genérico

Este tipo de encabezado es usado cuando se necesita incluir información de usuario de control para los paquetes transmitidos. El formato se puede ver en la figura 2.2 la información que puede contener es un subencabezado, que es aplicable en el caso de las transmisiones de paquetes de control o cuando se emplean funciones de empaquetamiento, concatenación o fragmentación.

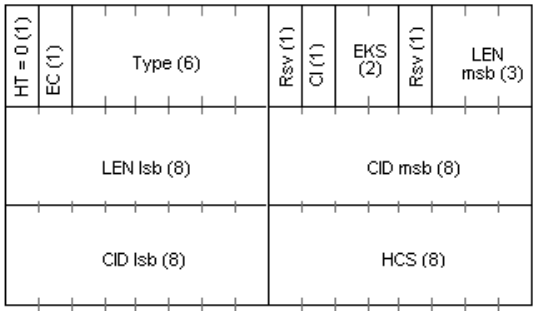


Figura 2.2. Encabezado Genérico.

Para este encabezado se tiene que especificar el HT = 0. El campo CI nos indica si va a ser incluido o no el CRC para detectar errores. El CID como se dijo anteriormente es de 16 bits. El campo EC nos indica si la información de la carga útil está encriptada o no. El campo EKS nos da la descripción de las posibles llaves utilizadas y se utiliza si el campo EC esta activado. HCS detecta errores dentro del encabezado. LEN nos indica la longitud en bytes del la carga útil y también del CRC si está incluido. Type nos indica el tipo de subcabecera utilizada y el tipo especial de carga usada.

Encabezado de Solicitud de Ancho de banda

Es exclusivo para las peticiones de oportunidad para transmitir información en el canal de subida. Su estructura la podemos ver en la figura 2.3. El encabezado de solicitud de ancho de banda tiene las siguientes propiedades: la longitud del encabezado debe de ser de 6 bytes, el campo EC debe de estar en cero y nos indica que no hay encriptación. El CID debe de indicar la conexión del canal de subida en el que se está haciendo la solicitud, el campo BR nos indica el número de bytes que se desea para al ancho de banda solicitado, el valor de HT debe de ser de 1.

Hay que notar que en este caso, el encabezado de solicitud de ancho de banda no tiene carga útil, podemos decir que su tamaño es constante, por eso siempre va a tener 6 bytes.

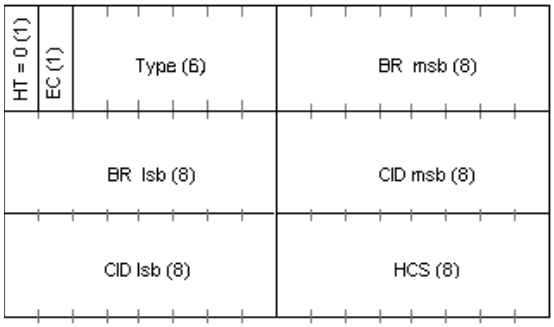


Figura 2.3. Encabezado de Solicitud de Ancho de Banda.

Mensajes de Administración MAC

Existen varios tipos de mensajes de administración MAC. Estos mensajes son añadidos en la carga útil de MAC PDU. Todos los mensajes de este tipo comienzan con un tipo de campo de mensaje de administración y después pueden contener campos adicionales. La estructura de este mensaje es muy simple y la podemos ver en la figura 2.4.

Figura 2.4. MAC Management Message.

El campo *type* nos indica los diferentes tipos de mensajes de administración que existen y posteriormente viene el campo de carga del mensaje de administración. Dependiendo del tipo puede ser enviado en mensajes para conexiones primarias, conexiones de broadcast, o conexiones primarias. En la tabla 2.1 podemos ver los principales mensajes de administración MAC.

Tipo	Nombre	Descripción	Conexión
0	UCD	Uplink Channel Descriptor	Broadcast
1	DCD	Downlink channel descriptor	Broadcast
2	DL-MAP	Downlink Access Definition	Broadcast
3	UL-MAP	Uplink Access Definition	Broadcast
4	RNG-REQ	Ranging request	Initial Ranging or Basic
5	RNG-RSP	Ranging response	Initial Ranging or Basic
6	REG-REQ	Registration request	Primary Management
7	REG-RSP	Resistration response	Primary Management
11	DSA-REQ	Dynamic Service Addition request	Primary Management
12	DSA-RSP	Dynamic Service Addition response	Primary Management
13	DSA-ACK	Dynamic Service Addition Acknowledge	Primary Management
26	SBC-REQ	SS Basic Capability request	Basic
27	SBC-RSP	SS Basic Capability response	Basic
30	DSX-RVD	DSx Received Message	Primary Management
31	TFTP-CPLT	Config File TFTP Complete message	Primary Management
32	TFTP-RSP	Config File TFTP Complete response	Primary Management
50-255		Reservados para usos futuros	

Tabla 2.1. Mensajes de Administración MAC

Mensaje Descriptor del Canal Descendente (DCD)

Es un mensaje que constantemente es enviado por la estación base. Nos indica las características usadas en el canal físico del canal descendente mediante los siguientes parámetros.

- Contador de Cambio de Configuración (*Configuration Change Count*): Es un valor que permite ignorar a la estación base el mensaje cuando este parámetro es igual al anterior. Se va incrementando de uno en uno (modulo 256) cada vez que el canal cambia.
- Identificador del Canal Descendente (*Downlink Channel ID*): Es un identificador del canal para operaciones específicas.

Este mensaje nos indica además el número de canal de radio frecuencia, número de configuraciones activas, la tasa de transmisión de símbolos, el inicio de la región activa en la trama y el final de la región activa en la trama.

Mensaje de Mapa del Canal Descendente (DL-MAP)

Este mensaje define nos indica los parámetros para poder acceder a la información en el canal descendente. Se incluye en el mensaje la siguiente información.

- Sincronización física (*PHY Synchronization*): Es variable según la especificación de la capa física.

- Contador *DCD* (*DCD Count*): Nos indica el valor del contador utilizado en el DCD para definir el perfil de transmisión descendente.

- Identificador de la estación base (*Base Station ID*): Este parámetro consta de 48 bits, la mitad de éstos están destinados a formar un identificador de operación que al combinarse con el Identificador del Canal Descendente que utiliza el mensaje DCD permite atender situaciones muy específicas.

- Número de Elementos (*Number of Elements*): Nos sirve para indicar la cantidad de elementos de información que están incluidos dentro de un mensaje.

Mensaje Descriptor del Canal Ascendente (UCD)

Es un mensaje que es transmitido periódicamente por la estación base, nos indica las características de un canal ascendente. Entre la información que se transmite por este mensaje se encuentra el tipo de modulación usada, la codificación, la longitud de preámbulo, bytes de información y bytes de paridad. Lo anterior lo podemos resumir en lo siguiente:

- Contador de Cambio de Configuración (*Configuration Change Count*): En un valor que se incrementa en uno módulo 256 cada ocasión que se realiza alguna variación en el canal. Con esto, las estaciones suscriptoras ignoran el resto del mensaje en caso de que el campo no hay cambiado de valor.

- Tamaño de ranuras (*Mini-slot Size*): Indica el número de ranuras físicas que equivalen a una ranura en el canal ascendente.

- Identificador del Canal Ascendente (Uplink Channel ID): Es el canal ascendente al que se está haciendo referencia.

- Fin de Backoff en Ranging (Ranging Backoff Start): Se expresa en potencia de 2 y varía entre 0 y 15. Es el tamaño de la ventana límite para la resolución de colisiones. Utiliza el algoritmo exponencial Backoff durante un periodo de Ranging.

- Inicio de Backoff en Solicitud (Request Backoff Start): Se expresa en potencia de 2 y varía entre 0 y 15. Es el tamaño de la ventana inicial para la resolución de colisiones. Utiliza el algoritmo exponencial Backoff durante un periodo de Ranging.

- Inicio de Backoff en Ranging (Ranging Backoff Start): Se expresa en potencia de 2 y varía entre 0 y 15. Es el tamaño de la ventana inicial para la resolución de colisiones. Utiliza el algoritmo exponencial Backoff durante un periodo de contención de asignación de ancho de banda.

- Fin de Backoff en Ranging (Ranging Backoff Start): Se expresa en potencia de 2 y varía entre 0 y 15. Es el tamaño de la ventana límite para la resolución de colisiones. Utiliza el algoritmo exponencial Backoff durante un periodo de contención de asignación de ancho de banda.

Mensaje Mapa del Canal Ascendente (UL-MAP)

Cuando una conexión se ha establecido entre la estación suscriptora y la estación base, se puede crear una o más conexiones para mandar datos a la estación base.

Para esto, se utiliza el canal UL para que las estaciones suscriptoras hagan la solicitud para tener oportunidades de transmisión. Todas estas peticiones las analiza la estación base y determina cuál es el número de time slots que le va a asignar a cada estación suscriptora en el canal de subida. Toda esta información es difundida a la red por medio de mensajes broadcast en el canal DL usando un mensaje UL-MAP.

Este tipo de mensaje proporciona la información para el acceso del canal ascendente y contiene la siguiente información: contador UCD, identificador del canal ascendente, tiempo de inicio de asignación (Alloc Start Time), número de elementos, elementos de información de mapa (MAP IE, Information Elements).

En cada UL-MAP debe de estar incluido un IE, por lo menos. Estos campos son los que definen la asignación de ancho de banda. En cada UL-MAP debe estar incluido al menos un IE, que indica el fin de la asignación de ancho de banda. Tienen la siguiente información y tienen la característica de que su orden es cronológico:

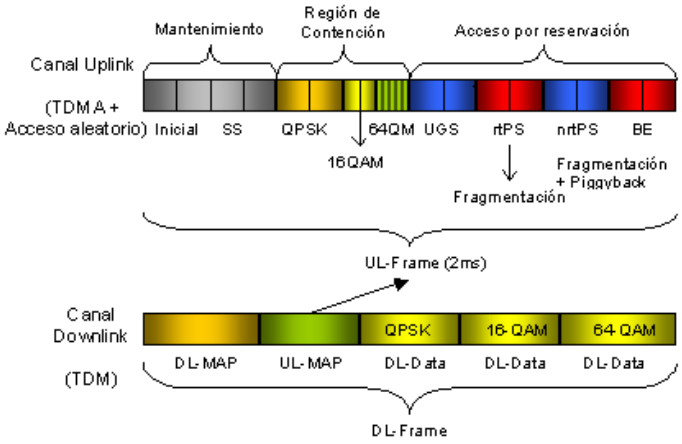


Figura 2.5. Estructura del canal DL y UL

- UIUC (Uplink Interval Usage Code).- Es el tipo de uso que se le está asignando al canal, por ejemplo, solicitud, mantenimiento, reservado, IE nulo o vacío o reservación del flujo de datos.

- Offset.- Indicador del intervalo en slots.

Mensaje de Solicitud de Ranging (RNG-REQ) y Respuesta de Ranging (RNG-RSP)

Este tipo de mensaje se utiliza en el momento en que se está iniciando la conexión, y es transmitido por la estación suscriptora. También puede darse el caso de que se transmita periódicamente si es solicitado por la estación base. Este mensaje nos sirve para realizar modificaciones a la potencia de transmisión y para los perfiles de transmisión descendentes.

Cuando la estación base recibe de la estación suscriptora el mensaje RNG-REQ contesta con un mensaje RNG-RSP. Este mensaje incluye información como potencia y frecuencia, ajustes para la sincronía y el estado de proceso de Ranging. Existe la posibilidad de que tenga el CID básico y primario así como la dirección MAC de la estación suscriptora durante el Ranging inicial.

Mensaje de Solicitud de Registro (REG-REQ) y Respuesta de Registro (REQ-RSP)

El mensaje REG-REQ es transmitido por la estación suscriptora en la inicialización. Este mensaje tiene la finalidad de informar a la estación base del número de CID's que puede soportar la estación suscriptora (al menos tres, la básica, la primario y la secundaria) así como la Secuencia Numérica de la Llave *HMAC (Hashed Message Authentication Code).* En este mensaje no se incluye la información que describe los canales físicos ni la asignación de ancho de banda.

Cuando este mensaje llega a la estación base, ésta responde con el mensaje (REQ-RSP) que nos indica si llegó exitosamente el primer mensaje, también incluye el CID para la conexión secundaria con la estación suscriptora, la versión MAC utilizada y la dirección MAC se añaden en el caso de redes Mesh, el código HMAC, así como una enumeración de las capacidades de la SS. Todo esto sirve para verificar las capacidades de la estación base con las capacidades que ha informado la SS.

En la siguiente figura se puede apreciar el orden en que los mensajes de control son intercambiados entre estación base y estación suscriptora.

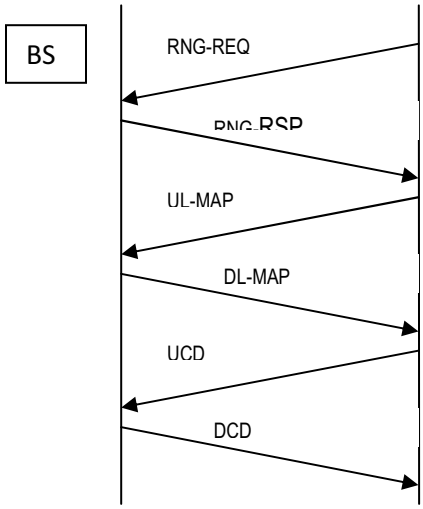


Figura 2.6. Mensajes entre BS y SS.

2.3.5 Fragmentación, Concatenación y Empaquetamiento

En el estándar IEE 802.16 MAC cada paquete de datos proveniente de su correspondiente subcapa de correspondencia, son empaquetados en MAC SDU (*Service Data Units*) y éstos a su vez son empaquetados en MAC PDU para poder ser transportados sobre una o más conexiones dependiendo del protocolo MAC. Posteriormente, después de viajar por la interfaz de aire, los MAC PDU son interpretados para generar los MAC SDU originales.

Para mejorar el uso del canal inalámbrico se cuentan con las siguientes funciones que permiten controlar la longitud de estos paquetes transmitidos y disminuir la sobrecarga.

- **Concatenación.**

Múltiples MAC PDU pueden ser unidos en una sola transmisión en la dirección de subida o de bajada. Es posible identificarlos mediante un CID único.

- **Fragmentación.**

Cuando un paquete MAC SDU excede la longitud disponible puede dividirse en múltiples segmentos y cada segmento es encapsulado en un MAC PDU. La capacidad para realizar fragmentaciones se indica al momento de establecer la conexión.

- **Empaquetamiento.**

Se utiliza cuando se quiere enviar múltiples MAC SDU pero todos compartiendo el mismo encabezado MAC. Tiene la característica de que los paquetes MAC SDU pueden ser de tamaño fijo, por ejemplo los usados en ATM y que usen la misma conexión. También puede darse el caso de que los paquetes MAC SDU sean de diferente tamaño, por ejemplo para paquetes IP.

2.3.6 Servicio de calidad en la transmisión ascendente

El estándar 802.16 está orientado a la conexión. Para hacer más eficientes los mecanismos de solicitud y asignación de la distribución del ancho de banda, existen cuatro servicios en el estándar que funcionan en el canal de subida. Cada uno de estos servicios nos proporciona un tráfico de datos especial para cada estación suscriptora. Cada servicio tiene diferentes niveles de latencia y ancho de banda.

Los servicios son:

***UGS (Unsolicited Grant Service)*:** Se utiliza para servicios orientados a conexiones en tiempo real que generan paquetes de longitud fijas en intervalos periódicos de tiempo. Por ejemplo, enlaces T1/E1 o voz sobre IP. En este caso la estación suscriptora no tiene que mandar ninguna solicitud de ancho de banda ya que previamente esta contemplado cuántos recursos va a utilizar. Tiene el tráfico tolerado máximo, latencia máxima y jitter tolerado.

***rtPS (Real-Time Polling Service)*:** Esta orientado al soporte de servicios en tiempo real que utilicen una longitud de paquetes variable y que sean transmitidos en intervalos periódicos de tiempo. Por ejemplo para servicios de video MPEG. Las estaciones suscriptoras no pueden usar la región de contención para realizar solicitudes, pero sí pueden utilizar PiggyBack. Los parámetros utilizados tienen tráfico tolerado máximo, tráfico reservado mínimo y latencia máxima.

***nrtPS* (*Non Real-Time Polling Service)*:** Similar a rtPS pero la consulta se realiza en espacios de tiempo más grandes y no necesariamente periódicos. Se utiliza para aplicaciones que no requieran transmisión de datos en tiempo real pero sí de un gran ancho de banda como la transmisión de datos FTP. La estación suscriptora puede hacer uso de la región de contención para solicitudes de ancho de banda y de PiggyBack.

***BE* (*Best Effort)*:** Se utiliza para la transmisión de tráfico ligero como HTTP, SMTP o Light FTP. Se utiliza donde no haya garantía de velocidad de los datos. En este caso la región de contención es utilizada para la asignación de ancho de banda y PiggyBack.

Para cada conexión de una estación suscriptora con la estación base existe una conexión que es identificada por el CID, que consta de 16 bits. Cada CID tiene asociado un Service Flow ID (SFID) que determina los parámetros de QoS que tendrá el CID.

Tres funciones son las que intervienen en la caracterización de los servicios: solicitudes anidadas (PiggyBack), robo de Ancho de Banda (Bandwidth stealing) y consulta (Polling).

PiggyBack Request

Son solicitudes de asignación de ancho de banda que están incluidas en los encabezados de los paquetes enviados que intercambia la estación suscriptora con la estación base. Para esto, la estación suscriptora tiene que hacer la solicitud dentro del periodo de contención para que la estación base pueda proporcionarle cierto número de minislots para transmitir, pero para volver a solicitar ancho de banda ya no es necesario esperar a la región de contención sino que los envía en los encabezados de la información a la cual ya se obtuvo ancho de banda para transmitir. Los usuarios que cuenten con el servicio de UGS no pueden implementarlo.

Bandwith Stealing

En este caso se utiliza parte del ancho de banda asignado para una transmisión para realizar la solicitud de más ancho de banda. Este proceso se realiza cuando la estación suscriptora tiene garantizada la transmisión de paquetes. Este servicio no es permitido para usuarios que cuenten con UGS.

Polling

Este proceso se caracteriza por tener oportunidades específicas para que la estación suscriptora pueda transmitir sus solicitudes de ancho de banda. Dicho de otra forma, la estación base asigna periódicamente cierto número de minislots para que la estación suscriptora pueda hacer dentro de los minislots las solicitudes necesarias.

2.3.7 Mecanismo de solicitud de ancho de banda

Durante la inicialización de las estaciones suscriptoras para poder ingresar a la red se utilizan tres diferentes CIDs para el propósito de enviar y recibir mensajes de control. Esas conexiones son utilizadas para alojar diferentes niveles de QoS para diferentes conexiones. El incremento o decremento del ancho de banda es

necesario para todos los servicios descritos anteriormente, excepto para UGS ya que utiliza un rango periódico de transmisión de paquetes constantes e implica que se garantice siempre un acceso sin variaciones mientras dure la conexión.

El mecanismo para solicitar por parte de las estaciones suscriptoras ancho de banda se le conoce como DAMA (*Demand Assigned Multiple Access*), consiste en que cada una de las estaciones suscriptoras envían mensajes a la estación base para solicitar minislots para realizar una transmisión en el canal ascendente. Las oportunidades de transmisión concedidas dependen del tráfico generado en ese momento así como los parámetros de QoS que se establecieron en la conexión inicial con la estación base. Los métodos por los que las estaciones suscriptoras hacen esta petición son: Solicitudes, Reservaciones por Conexión, Reservaciones por Estación Suscriptora y Consulta.

Solicitudes

Las solicitudes se refieren al mecanismo por el cual las estaciones suscriptoras le indican a la estación base que necesitan ancho de banda para el canal de subida. Todas las solicitudes de ancho de banda deben de ser en términos del número de bytes requeridos para la carga útil y la cabecera MAC, y no se incluye a la capa PHY. Las solicitudes realizadas durante una región reservada se realizan mediante solicitudes Piggyback.

Las solicitudes pueden ser incrementales o totales, esto se indica en el encabezado del mensaje de solicitud de ancho de banda. Ya que las peticiones Piggyback no cuentan en el subencabezado con un campo para definir el tipo, siempre serán del tipo incrementales. Con la solicitud incremental podemos aumentar el número de bytes que la estación suscriptora demanda a la estación base y este número adicional de bytes sólo se modifica enviando mensajes de este tipo.

En el caso de las solicitudes totales podemos observar que se renueva el registro de la estación suscriptora en la estación base de acuerdo a la última petición enviada. Aquí no importa si el ancho de banda requerido anteriormente se haya cubierto en una porción. La estación base puede solicitar que se generen solicitudes totales en un esquema periódico en las estaciones suscriptoras y depende de los QoS que se tenga para un servicio en particular. Todas las solicitudes generadas en la región de contención siempre serán totales.

Modo de Asignación por Conexión (GPC)

Las asignaciones por conexiones (*Grant per Connection*) son especificadas sólo para un CID. Una estación base cuenta con diversas conexiones durante su operación por lo que el GPC implica que se pueda modificar la transmisión de tráfico generado por alguna de las conexiones o también descartarla. Esto sucede porque no siempre se puede asegurar para una conexión que van a existir los recursos suficientes para satisfacer su demanda.

Modo de Asignación por Estación Suscriptora (GPSS)

En este caso, a las estaciones suscriptoras se les asigna ancho de banda extra dentro del mismo *grant*. Cuando la estación suscriptora está manejando el modo *GPSS (Grant per Subscriber Station),* las asignaciones realizadas por la estación base se generan por medio del CID básico de la conexión de la estación suscriptora. Con esto se logra que la estación suscriptora pueda establecer cuál o cuáles de las conexiones de las que tiene en ese momento pueden contar con la reservación de ancho de banda o con una parte de ella. Esto aumenta la capacidad del sistema ya que se puede decidir cuánto ancho de banda se le asigna a cada conexión de la estación base.

Puede darse el caso de que las asignaciones hechas por la estación suscriptora no cumpla con el ancho de banda que se requiere, en ese caso, al igual que en GPC, puede descartarse la asignación y solicitar una nueva y considerar a la anterior como ignorada.

Típicamente el ancho de banda que se le ofrece a cada conexión de la estación suscriptora se utilizará para ésta, aunque no necesariamente. Esto implica que las estaciones suscriptoras deben de ser capaces de decidir si lo usan o no, estableciendo QoS más rigurosos.

Consulta (Polling)

Como se mencionó anteriormente, aquí la estación base establece tiempos específicos para que las estaciones suscriptoras puedan transmitir los mensajes de solicitud de ancho de banda. Esto es establecido dentro del *IE* de cada mapa.

Las consultas pueden dividirse en Unicast y Multicast. Para este último caso las estaciones suscriptoras tendrán que competir para el envío de los mensajes. En el caso de una consulta Unicast para una estación suscriptora en particular la estación base incluye en el mensaje UL-MAP una oportunidad única para que la estación suscriptora solicite ancho de banda. Para el caso de una consulta Multicast, se utiliza cuando la estación base no cuenta con el ancho de banda necesario para atender a un grupo de estaciones suscriptoras. Cada grupo se identifica mediante CIDs en específico y las oportunidades se incluyen en el mensaje UL-MAP.

Las estaciones suscriptoras sólo enviarán consultas si lo necesitan, participando en el proceso de contención. Para este proceso donde compiten se emplea el algoritmo exponencial backoff tanto para determinar el minislot donde solicitarán el ancho de bando como asignarle uno nuevo en caso de que se presente alguna colisión.

2.3.8 Característica de la capa Física

Las principales características en la capa PHY, para la banda de frecuencias 10 – 66 GHz es que se considera una propagación por línea de vista, además en este rango se utiliza la modulación por portadora única. En el caso en que la banda de frecuencia sea de 2 – 11 GHz, la capa física fue diseñada para que no se utilice línea de vista para la propagación. [7]

El estándar IEEE 802.16 soporta TDD (*Time Division Duplex*) y FDD (*Frequency Division Duplex*), principalmente estos dos tipos de duplexación se utilizan para mejorar el uso del espectro electromagnético. Ambos utilizan un formato de transmisión en ráfagas las cuales soportan perfiles de transmisión adaptativo en donde se incluyen parámetros de transmisión, incluyendo esquemas de modulación y codificación, además estas características pueden ser ajustadas individualmente a cada SS.

En esta capa se definieron cinco especificaciones diferentes, las cuales son:
- WirelessMAN-SC PHY
- WirelessMAN-SCa PHY
- WirelessMAN-OFDM PHY
- WirelessMAN-OFDMa PHY
- WirelessHUMAN PHY

WirelessMAN-SC PHY

Esta diseñada para operar en la banda de frecuencia de 10-66 GHZ, permitiendo el uso flexible del espectro, dando soporte a las configuraciones TDD y FDD. Ambos casos usan el formato de transmisión burst cuyos mecanismos de framing soportan burst adaptativos en el cual los parámetros de transmisión, incluyendo los esquemas de modulación y codificación, pueden ser ajustados individualmente para cada SS en la base de trama por trama. El caso FDD soporta SS's full-duplex y también SS's half-duplex.

En canales de frecuencia que pueden estar entre 20MHz y 28MHz, y los anchos de banda para la transmisión están entre 32Mbps y 134.4Mbps; estos valores varían dependiendo de la técnica de modulación utilizada, la cual puede ser QPSK, 16-QAM ó 64-QAM.

WirelessMAN-SCa PHY

Basado en una tecnología de portadora simple y diseñado para operación NLOS en bandas de frecuencias por debajo de 11 GHz. Para bandas licenciadas los anchos de banda de canal permitidos deben ser limitados por el ancho de banda regulado dividido por cualquier potencia de dos no menor que 1.25 MHz.

Esta especificación usa cinco términos para la organización de transmisiones: payload, burst, burst set, burst frame y MAC frame.

- Payload, es el contenido de unidades de transmisión.
- Burst, contiene datos del payload, además la existencia del burst es bien conocida por el receptor a través del contenido de los mapas de uplink o downlink.
- Burst set, es una entidad de transmisión autoconcatenada consistente de un preámbulo, el burst set es sinónimo de burst cuando se encuentra en el canal de uplink.
- Burst frame, contiene toda la información incluida en una transmisión simple.
- MAC frame, son los intervalos de ancho de banda fijo reservados para el intercambio de datos.

WirelessMAN-OFDM PHY

Basado en la modulación OFDM para ambientes NLOS, con bandas de frecuencia por debajo de los 11GHz. Está orientado hacia los accesos fijos, tales como residencias y empresas.

Los símbolos OFDM están constituidos por cierto número de subportadoras, el cual depende de la FFT (*Fast Fourier Transform*) que se aplique. Las modulaciones que se usan son: BPSK, QPSK, 16- QAM y 64-QAM.

WirelessMAN-OFDMa PHY

Diseñado para enlaces NLOS en bandas de frecuencia por debajo de los 11GHz, sin ser menor a 1.0 MHz utilizando modulaciones QPSK, 16-QAM ó 64-QAM.

WirelessHUMAN PHY

Diseñado para canales frecuencia entre 10MHz y 20MHz, con una separación de 5 MHz entre sí, en la banda de frecuencia de 5 GHz a 6 GHz [8]**.**

FDD (Frequency Division Duplex)

En la operación de FDD, el canal de subida (uplink) y el de bajada (downlink) trabajan en frecuencias diferentes, así que la estación base (BS) puede transmitir en ráfagas, ya que del canal de bajada DL tiene la capacidad de ser transmitida en ráfagas, además de facilitar el uso de diferentes tipos de modulación, también permite el funcionamiento simultaneo de full–dúplex y half-duplex en las estaciones suscriptoras (SS).

TDD (Time Division Duplex)

En el caso de TDD, la transmisión del canal de subida y bajada comparten la misma frecuencia, pero tienen separaciones en el tiempo, como se muestra en la figura 2.7. La trama TDD tiene una duración fija, también ésta contiene un UL – subframe y un DL – subframe.

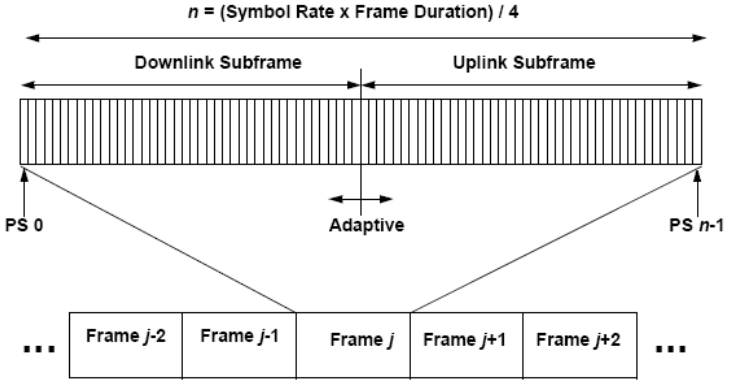


Figura 2.7. Estructura del frame para TDD

Además este esquema no soporta la transmisión y recepción de forma simultánea, es decir, opera de modo "half – duplex".

Canal de Subida

En el canal de subida se lleva a cabo una combinación entre las multiplexaciones TDMA y DAMA para su funcionamiento.

El ancho de banda disponible es definido por medio de minislot, éste se calcula con la expresión:

$$2^m(PS) \qquad ;donde\ 0 \leq m \leq 7$$

Este canal utiliza tres tipos de ráfagas para transmitir en un SS, los cuales son:

a) Cuando las ráfagas son transmitidos en oportunidades de contención reservados para "ranging"

b) También, aquellos que son transmitidos en oportunidades de contención definidos por el intervalo reservado de petición con el fin de responder al multicast y broadcast.

c) Y, aquellos que son transmitidos definidos en intervalos por los datos IE (*Information Element*) que está alojado en cada SS's.

Cualquiera de estos tipos de ráfagas pueden estar presentes en un frame en cualquier cantidad (limitado por la cantidad de PS's disponibles en el frame) determinado por la BS como lo indica el UL – MAP en la sección de control del frame.

La asignación del ancho de banda para "ranging" y la solicitud de ancho de banda pueden agruparse, pero siempre son utilizadas en el perfil de las ráfagas del canal de subida. Además van marcados con UIUC = 2 (*Uplink Interval Usage Code*) y UIUC = 1 respectivamente.

El SSTGs (*Subscriber Station Transition Gap*) Separa la transmisión de varios SS's durante el subframe "uplink". Esta separación (gap) le permite a la BS lograr sincronizarse con el nuevo SS.

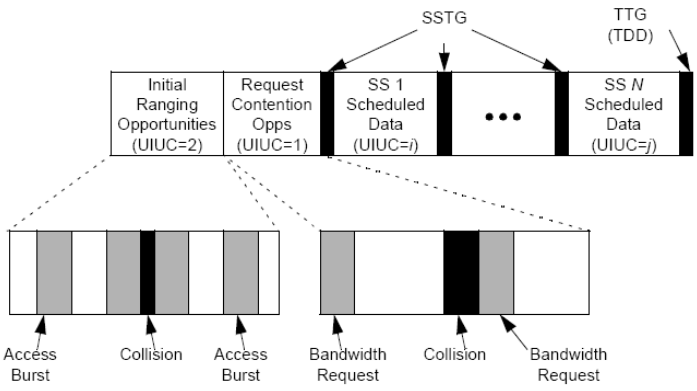


Figura 2.8. Estructura del frame del canal de subida

Canal de Bajada

En este canal se utiliza la multiplexación TDM (*Time Division Multiplexing*), debido a que solo hay una señal multiplexada que contiene la información de cada SS, así que todas las SS's conocen las características de los demás SS's.

El ancho de banda disponible en el canal de bajada se define mediante slots físicos PS (*Physic Slot*). El numero de PS's de un frame está en función de la tasa de modulación, además esta tasa de modulación se selecciona para que se logre obtener un numero entero de PS's en cada frame.

La estructura de un subframe utilizando TDD en el canal de bajada se ilustra en la figura 2.9, donde se observa que el subframe inicia con una sección de preámbulo, el cual se encarga de la sincronización y ecualización de la capa PHY, después se encuentra la sección de control que contiene los mensajes DL – MAP (Downlink map) y UL – MAP (Uplink Map), estos mensajes indican el inicio de la transmisión a través de los PS's. En la siguiente sección se encuentran los datos organizados por ráfagas con diferentes niveles de transmisión.

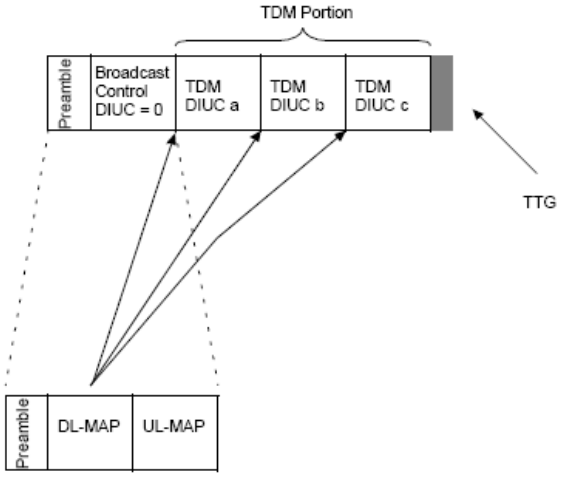


Figura 2.9. Estructura del frame del canal de bajada utilizando TDD

La estructura del subframe utilizando FDD se ilustra en la figura 2.10. Al igual que el caso de FDD, el subframe inicia con una sección de preámbulo seguido por una sección de control, además existe una sección TDM (*Time Division Multiplexing*), el cual se organiza en ráfagas en orden decreciente de un perfil de robustez. La sección TDM contiene los datos transmitidos a los SS's; mediante una conexión full – dúplex, una conexión half – dúplex donde se ha programado la transmisión, también se utiliza para transmisiones no programadas.

Después de la sección TDM, el frame cuenta con una porción TDMA para transmitir datos hacia las diferentes SS's bajo la conexión half – dúplex. Las ráfagas de esta sección inician con un preámbulo para la sincronización de fase, además en estas ráfagas no se necesita que se coloquen por orden de robustez.

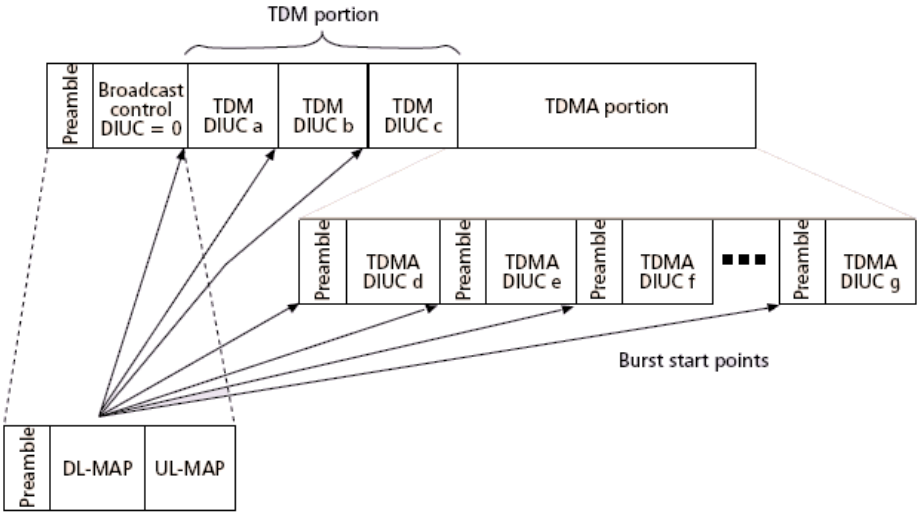


Figura 2.10. Estructura del frame del canal de bajada utilizando FDD

2.4 Colisiones

Las colisiones se presentan cuando dos o más SS's transmiten información hacia la BS en el mismo tiempo, esto hace que la estación base no logre entender ninguna de las SS's. Como consecuencia de esto, hay desperdicio de ancho de banda en el canal ascendente además, todo el sistema se ve afectado porque hay pérdida de información al tratar de seguir transmitiendo.

Principalmente las colisiones ocurren durante los intervalos de *Backoff* inicial y *Backoff* final, estas regiones de colisión las especifica la BS a través del UL-MAP. Los métodos requeridos para solucionar las colisiones son aplicados por las SS's que estén involucradas en este proceso.

La resolución de colisiones se basa en el algoritmo *Exponencial Backoff Binario Truncado,* cuyos parámetros necesarios son el intervalo inicial y final de Backoff. Éstos son especificados por el BS dentro de los mensajes UCD (*Uplink Channel Descriptor*) utilizando potencias binarias.

El algoritmo *Exponencial Backoff Binario Truncado* sigue el siguiente mecanismo:

- Si un SS tiene información para enviar y necesita entrar en el proceso de resolución de colisiones, éste selecciona aleatoriamente un número "n" (indica el número de oportunidades de transmisión que se deberá dejar pasar antes de volver a transmitir) que se encuentre dentro de la ventana de *backoff inicial*.

- Si el nodo no recibe algún mensaje de notificación para transmitir en un tiempo definido por parte de la BS, se considera que la transmisión de solicitud de BW colisionó con un mensaje de otro nodo. Al ocurrir esto, el SS intentará incrementar la venta de *backoff utilizando un* factor de 2 por lo que vuelve a escoger un número "n" dentro de la nueva ventana.

- En el caso de que el SS reciba una asignación de BW, por parte de la BS, concluye el proceso de resolución de colisiones.

- Pero si vuelve a pasar un tiempo especificado sin recibir una asignación de BW, el nodo considera que hubo otra colisión y vuelve a incrementar la ventana de *backoff* repitiendo el proceso hasta que el nodo reciba una asignación de BW o hasta que la ventana de *backoff* sea igual al valor de ventana de *backoff* final. [9]

Capítulo 3

Descripción y Operación de la Tecnología Mesh

3.1 Introducción

Las redes BWA han presentado un crecimiento exponencial en crecimiento durante las últimas décadas. Cuando se utiliza el término de banda ancha se refiere a velocidades de transmisión mayores a 1.5 Mbps aunque muchas de las redes BWA soportan velocidades aún mayores para transmitir la información. Una característica de las redes BWA es que el usuario nunca se da cuenta de que se está entregando la información por medio de un enlace de radio. Además una red BWA soporta la conexión de varias estaciones dentro de un área de cobertura, y al tener varias conexiones el ancho de banda se comparte entre todos los usuarios. Otra de las características es que las redes BWA utilizan el reuso de frecuencias para poder optimizar la eficiencia del sistema.

Las redes BWA ofrecen una conectividad que soluciona los requisitos de comunicación de diversas aplicaciones, dentro de las cuales están los servicios multimedia. En algunos lugares, estos sistemas son mejor conocidos como *Multimedia Wireless Systems* (MWS), ya que estos sistemas reflejan la convergencia de los servicios tradicionales de telecomunicaciones con los servicios de entretenimiento.

Existen dos formas básicas de poder implementar una red BWA, un sistema PMP y un sistema Mesh. Los sistemas PMP contienen a la estación base, las estaciones de abonado y pueden incluir algunos repetidores. En este caso las estaciones base cuentan con antenas que usan un ancho de haz grande, a su vez, estas antenas se complementan con otras antenas y se les conoce como sectores, en conjunto todos los sectores proveen una cobertura de 360°. Puede darse el caso de que exista más de una estación base, y para conectarlas se puede utilizar algún enlace de radio o fibra óptica, a esta conexión de las estaciones base ya no se le considera dentro de la red BWA y pueden alcanzar velocidades en el rango de DS-*3* (44.74 Mbps) y *OC-3* (155.52 Mbps). Para las estaciones de abonado se utilizan antenas direccionales para poder apuntar a las estaciones base, y todas las estaciones de abonado comparten el uso de los canales. Existen los métodos de multiplexación por división de tiempo, por división de frecuencia y por división de código para lograr compartir los canales.

El sistema Mesh también es conocido como sistema MP y básicamente tiene las mismas características que los sistemas PMP, la diferencia radica en las estaciones de abonado que pueden ser utilizadas también como un repetidor para que el tráfico pueda pasar a través de los repetidores y puedan llegar a la línea de abonado. Las antenas también pueden ser direccionales y están dirigidas estratégicamente. Un repetidor trabaja a la misma o diferente frecuencia que la estación base.

3.2 Características relevantes

Las redes Mesh presentan un conjunto de características deseables, como la robustez y la capacidad de extender la cobertura de redes fijas o inalámbricas, apoyadas en una estación base, hacia usuarios remotos que no pueden acceder a ellas directamente por causa de obstáculos, distancia o carencia de infraestructura.

Una red Mesh es compuesta por una colección de nodos que se comunican entre sí, de manera directa. En este sistema la comunicación se realiza entre los nodos directamente y cada nodo puede ser al mismo tiempo fuente o destino de los datos o un enrutador de la información de otro nodo.

Si los nodos de la red se conectan de manera autónoma, la red opera en modo *ad hoc.* Si los nodos tienen movilidad, entonces se conocen como redes móviles *ad hoc* o MANET (Mobile ad hoc NETwork). Su característica principal es que existe un continuo cambio en la topología de la red, con enlaces que aparecen y desaparecen de modo permanente.

Las principales características de las redes Mesh inalámbricas son:

- **Robustez:** La presencia de enlaces redundantes entre los usuarios permite que la red se reconfigure automáticamente ante fallas.
- **Topología dinámica:** Las redes Mesh tienen la capacidad de reaccionar ante cambios de la topología de la red.
- **Ancho de banda limitado:** Las redes Mesh cuentan con enlaces que usualmente permanecen en condiciones de congestión.
- **Seguridad:** La información transmitida se encuentra expuesta a la amenaza de viajar a través de un medio compartido. El estándar define una subcapa de seguridad para proteger la información de los usuarios y evitar el acceso de usuarios no autorizados.
- **Canales de comunicación aleatorios:** Las redes inalámbricas cuentan con la incertidumbre propia de los canales de comunicación de radio. El estándar define aspectos como la modulación y codificación adaptativas para hacer frente a este problema.
- **Carencia de modelos de dimensionamiento apropiados**

Las redes Mesh proveen, sin embargo, condiciones que permiten el acceso a usuarios en regiones apartadas.

A medida que un usuario se aleja, la velocidad del canal disminuye por cuenta de la reducción del esquema de modulación y el aumento de la robustez de la codificación para compensar la disminución de la potencia recibida. Si este usuario se puede conectar a la estación base por medio de dos o más saltos, de menor distancia, se podría contar con enlaces de mayor velocidad, aumentando su capacidad. Este argumento no es completamente válido si no se tiene en cuenta que al aumentar los usuarios que acceden de la misma forma, la agregación de tráfico en el nodo intermedio hace que la capacidad no pueda incrementarse de la manera pensada. Sin embargo, las estaciones base de los sistemas de acceso serán ubicadas en zonas de alta densidad de usuarios, pero la cobertura de las regiones alejadas, puede lograrse por medio de mallas que extiendan la cobertura de la celda.

El estándar 802.16-2004 define las diferentes tramas, campos y procesos necesarios para el establecimiento y operación de las redes enmalladas, así como las condiciones de las tramas y subtramas para la comunicación.

Este tipo de redes se compone de un conjunto de tramas donde los usuarios pueden transmitir, previa asignación de rangos de tiempo. Las tramas se dividen en subtramas de control y de datos.
Las primeras permiten controlar la operación de la malla y realizar procesos de solicitud y asignación de capacidad.

Las subtramas de datos son divididas en unidades de tamaño fijo, llamadas *minislots,* las cuales son la unidad de asignación de la capacidad a los usuarios. Un usuario puede recibir entre 1 y 256 unidades anteriores, especificando el *minislot* de origen y la cantidad de *minislots* asignados.

Los procesos desarrollados en las redes enmalladas se pueden dividir en tres fundamentalmente, entrada a la red, establecimiento de enlaces y negociación de la capacidad. [8]

3.3 Estructura de la trama para modo Mesh

El estándar IEEE 802.16 en su capa MAC utiliza dos tipos de configuraciones, la punto a multipunto (PMP) y el modo Mesh. En esta última configuración los nodos se encuentran en *ad hoc*. Todas las estaciones suscriptoras pueden actuar como routers para enviar paquetes de sus vecinos. Un nodo puede elegir los enlaces con la mejor calidad y junto a un eficiente protocolo de ruteo, el tráfico puede ser dirigido para evitar un área congestionada.

Como se ha mencionado anteriormente, el estándar IEEE 802.16 en modo Mesh utiliza TDMA (*Time Division Multiple Access)* para el canal de subida entre las estaciones suscriptoras y la estación base. Para mandar la información el canal de radio se divide en varias partes conocidas como Frames. Cada Frame a su vez es dividido en *time slots* que puede ser asignado a la estación base o bien, a las estaciones suscriptoras. A su vez, cada Frame es dividido en 256 *minislots* que son utilizados para transmisiones de mensajes de control y de datos, los cuales se especificarán más adelante.

Cada frame en modo Mesh puede tener una duración variable, puede ser un Frame de 10 ms o bien, puede ser un frame de 2.5 ms. Este valor es definido en la configuración de la red Mesh.

3.3.1 Subtrama de Control y Datos

En la subtrama de control, las oportunidades, que típicamente consisten de múltiples minislots, son usadas para transportar mensajes de señalización para poder establecer una configuración de la red Mesh. Existen dos tipos de frames de control, el subframe de control de la red y el subframe de *scheduling*.

Todos los nodos compiten para poder transmitir un mensaje en la subtrama de control. Los mensajes de control son enviados después de que un número N de mensajes de *scheduling* son enviados, donde N es un valor configurable de la red.

Dentro de los mensajes de control de la red existen los mensajes *Mesh Network Configuration (MSH-NCFG)* y *Mesh Network Entry (MSH-NENT)* que son mensajes transmitidos para la añadir nuevos nodos y para mantenimiento de la configuración de la red. Las estaciones suscriptoras periódicamente envían mensajes MSH-NCFG que contienen un parámetro llamada Network Descriptor que es el encargado de mandar la información de configuración de la red. [7]

Después de que se envía un frame con los mensajes *MSH-NENT* y *MSH-NCFG* vienen un número *N* de frames de scheduling, donde *N* es un valor ajustable en la red. En estos frames observaremos los mensajes *MSH – CSCH* (*Mesh Centralized Schedule*) y *MSH – DSCH (Mesh Distributed Schedule*). Estos mensajes serán usados dependiendo del tipo de tráfico que se vaya a utilizar en la red Mesh, si el tráfico es externo entonces se usará la configuración centralizada, si el tráfico es interno se usará la configuración distribuida.

Como se dijo anteriormente, cada Frame en el modo Mesh utiliza 256 minislots. Se puede asignar un número variable de oportunidades de Transmisión (TO). Cada estación suscriptora utiliza una oportunidad de transmisión para poder enviar el mensaje de control. Cada oportunidad de transmisión consta de siete símbolos OFDM, a su vez, siete símbolos OFDM constan de 504 bytes. Así mismo, cuatro símbolos OFDM forman un minislot.

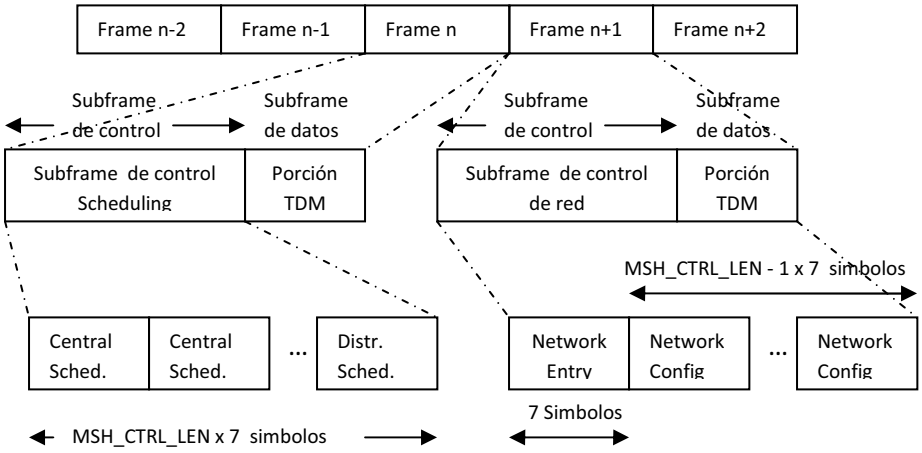


Figura 3.1. Subtrama de control y datos.

Existe un parámetro llamado *MSH-CTRL-LEN* que nos indica la longitud o bien, las oportunidades de transmisión para poder enviar mensajes de control. Como el MSH-NENT sólo es transmitido una ocasión en algunos frames entonces se tendrán *MSH-CTRL-LEN* – 1 oportunidades de transmisión para los mensajes MSH – NCFG.

Además, los mensajes NENT no son enviados en todos los frames, sino que este parámetro es ajustable en el sistema pudiéndose enviar cada cierto número de frames, por ejemplo cada 10 frames que es el valor al que lo ajustamos para las simulaciones realizadas.

Después de un subframe de control viene un subframe de datos. Dentro de este subframe de datos es donde las estaciones suscriptoras enviarán la información.

3.4 Mensajes del IEEE 802.16 modo Mesh

3.4.1 MSH – NCFG (Mesh Network Configuration Message)

Los mensajes MSH-NCFG proveen un nivel básico de comunicación entre los nodos. Todos los nodos que se encuentren en la red y utilizando la configuración Mesh transmitirán este tipo de mensajes tal como se muestra en la tabla 3.1.

Syntax	Size
MSH-NCFG_Message_Format() {	
Management Message Type = 39	8 bits
NumNbrEntries	5 bits
NumBSEntries	2 bits
Embedded Packet Flag	1 bit
Xmt Power	4 bits
Xmt Antenna	3 bits
NetEntry MAC Address Flag	1 bit
Network base channel	4 bits
reserved	4 bits
NetConfig Count	4 bits
Timestamp Frame Number Network Control Slot Number in frame Synchronization Hop Count	 12 bits 4 bits 8 bits

NetConfig schedule info Next Xmt Mx Xmt Holdoff Exponent	 3 bits 5 bits
if (NetEntry MAC Address Flag) NetEntry MAC Address	48 bits
for (i = 0: i< NumBSEntries: +-+1) {	
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()	Variable
}	

Tabla 3.1 Encabezado del mensaje MSH-NCFG

En donde:

- **NumNbrEntries:** Representa el número de vecinos reportados en el mensaje. Este valor puede ser variable ya que al principio sólo va a reconocer a algunos vecinos y conforme se vayan dando de alta los nodos se podrá incrementar este valor enviado en mensajes NCFG siguientes.
- **NumBSEntries:** Nos indica el número de estaciones base que se encuentran.
- **Xmt Power:** En pasos de 2 dBm, empezando de 8 dBm, por ejemplo se puede poner 1111 que nos indica que son 38 dBm.
- **Xmt Antenna:** Nos indica 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 pase 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. Hay que tomar en cuenta que:

$$\text{Xmt Holdoff Time} = 2^{(\text{Xmt Holdoff Exponent} + 4)}$$

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

- **Next Xmt Mx:** Es el tiempo en que un nodo puede enviar los mensajes MSH-NCFG y se define de la siguiente manera:

$$2^{\text{Xmt Holdoff Exponent Next}} \bullet \text{Xmt Mx} < \text{Next Xmt Time} \leq 2^{\text{Xmt Holdoff Exponent}} \bullet (\text{NextXmtMx} + 1)$$

- **NetEntry MAC Address:** Nos indica la presencia de un nuevo nodo en la red.
- **BS node ID:** Es el número de nodo que se le está asignando 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:** Es el número de nodo de la estación suscriptora.

El último campo es importante ya que en éste se especifica de qué tipo va a ser el mensaje MSH-NCFG ya que pueden existir varios. Los tipos que existen son:

- Network Descriptor
- Network Entry Open
- Network Entry Reject
- Network Entry Ack
- Neighbor Link Establishment Protocol

Más adelante se describirá cómo es que funciona cada uno de ellos.

3.4.2 MSH – NENT (Mesh Network Entry Message)

El mensaje MSH – NENT se utiliza cuando se responde por parte de una estación suscriptora un mensaje MSH – NCFG. Tiene la estructura que se muestra en la tabla 3.2. Este tipo de mensajes provee de sincronización y además son enviados para que nuevos nodos entren a la red.

Cuando un mensaje MSH – NENT es enviado, el sub encabezado Mesh tiene el valor 0x0000 hasta que se le asigne un Node ID al nodo que lo envía [1].

Syntax	Size
MSH-NENT_Message_Format() {	
Management Message Type = 40	8 bits
Type	3 bits
Xmt counter for this Type	3 bits
reserved	2 bits
Sponsor Node ID	16 bits
Xmt Power	4 bits
Xmt Antenna	3 bits
reserved	1 bit
if (Type = 0x2)	
MSH-NENT_Request_IE()	variable
}	

Tabla 3.2. Encabezado del mensaje MSH-NENT

Donde:

- **Sponsor Node ID:** Es el número de nodo al cual esta accediendo la estación suscriptora para poder incorporarse en la red.
- **Xmt Power:** Se asigna en pasos de 2 dBm y empieza en el nivel de 8 dBm.
- **Xmt Antenna:** Soporta ocho direcciones.

A su vez, la parte de MSH-NENT_request_IE se describe de la manera siguiente:

Syntax	Size
MSH-NENT_Request_IE() {	
MAC Address	48 bits
OpConfInfo	64 bits
Operator Authentication Value	32 bits
Node serial Number	32 bits
}	

Tabla 3.3. Encabezado del mensaje MSH-NENT_request_IE

Donde:

- **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.

3.4.3 MSH – DSCH (Mesh Distributed Scheduling Message)

Los mensajes MSH – DSCH son transmitidos cuando se utiliza una configuración de parámetros distribuidos. El mismo algoritmo utilizado para transmitir los mensajes MSH – NCFG es utilizado para enviar este tipo de mensajes, ya que no pueden existir colisiones cuando se transmite. Estos mensajes son usados para indicar a todos los demás vecinos los recursos disponibles. La finalidad de este mensaje es que los demás vecinos sepan en qué momento se va a transmitir para evitar colisiones. Al igual que ocurre con los mensajes MSH - NCFG, se pueden configurar el número de mensajes MSH – DSCH que existirán en un frame. El formato del frame se puede consultar en el estándar IEEE 802.16.

En este caso, el valor del parámetro Next Xmt Time está por la variable Next Xmt Mx y va dentro de MSH – DSCH Scheduling IE(). Se define de la siguiente manera:

$$2^{\text{Xmt Holdoff Exponent Next}} \cdot \text{Xmt Mx} < \text{Next Xmt Time} \leq 2^{\text{Xmt Holdoff Exponent}} \cdot (\text{Next Xmt Mx} + 1)$$

$$\text{Xmt Holdoff Time} = 2^{(\text{Xmt Holdoff Exponent} + 4)}$$

3.4.4 MSH – CSCH (Mesh Centralized Scheduling Configuration Message)

Este tipo de mensajes es utilizado para el tráfico centralizado de la red. Un ejemplo de esto es cuando se quiere transferir información de la estación base a las estaciones suscriptoras para tráfico externo, como internet. La estación base transmitirá los mensajes MSH – CSCH por broadcast. Los nodos pueden utilizar este mensaje para hacer solicitudes de ancho de banda, además cada nodo reporta el ancho de banda de los nodos hijos en la configuración Mesh. El formato del frame se puede consultar en el estándar.

3.5 Arquitectura de las redes Mesh.

3.5.1 Modo centralizado

Una red Mesh está constituida por una colección de nodos que se comunican entre sí, de manera directa.
La comunicación se realiza entre los nodos directamente y cada nodo puede ser al mismo tiempo fuente o destino de los datos o un enrutador de la información de otro nodo.

Puede existir una entidad central que administre las condiciones de operación de la red, en cuyo caso se conoce como centralizado. En modo centralizado, la estación base es la que realiza todos los cálculos para indicarles a las estaciones suscriptoras cómo deben administrar los diferentes time slots para poder transmitir información. Debido a que todos los paquetes de control y de datos tienen la necesidad de ir a través de la estación base, el mecanismo es sencillo sin embargo la conexión es más lenta. Además, el modo centralizado no es óptimo para mejorar el tráfico que existe en la red debido a sus características.

3.5.2 Modo descentralizado

Si no hay necesidad de una entidad centralizada que los controle, el modo de operación se conoce como distribuido. En este modo, cada nodo compite por acceso al canal usando un algoritmo pseudoaleatorio basado en la información de sus dos nodos vecinos más cercanos. Los subframes de datos son asignados en base a la solicitud, entrega y confirmación. Además, este modo de operación, a diferencia del modo centralizado, es más eficiente en la conexión y en la transmisión de datos. Otro aspecto importante es que el esquema descentralizado permite una mejor flexibilidad y escalabilidad para la red, sin embargo, es más complejo debido a que cada nodo determina su tiempo de transmisión sin ningún tipo de información global de la red.

En este caso, los mensajes de control y los paquetes de datos están colocados en diferentes time slots en un frame. En este esquema, un nodo selecciona su siguiente tiempo de transmisión basado en el tiempo actual. Esta es la razón del por cual se hace necesario emplear un algoritmo de selección. Si el nodo gana, el nodo manda un mensaje de broadcast y repite el procedimiento en el siguiente tiempo de transmisión. Si pierde, el nodo selecciona el siguiente tiempo para transmitir y continúa el procedimiento hasta que éste gane. [10]

3.6 Procedimiento del proceso "Network Entry"

En la figura 3.2 se describe el procedimiento del proceso "Network Entry". En primer lugar, un nuevo nodo tiene que escuchar el mensaje MSH – NCFG, que transmite el nodo vecino, o bien la estación base. Además este último nodo debe monitorear continuamente el mensaje MSH – NCFG de todos sus nodos vecinos y hacer una lista con sus respectivas direcciones físicas cuando reciba el mensaje MSH – NCFG por segunda vez.

Por lo que cada nuevo nodo que se vaya a dar de alta debe de de escoger un vecino patrocinador, el cual debe de elegirlo de la lista y después debe de iniciar el proceso de sincronización, es decir tener el mismo tiempo que el nodo patrocinador.

Además, el nuevo nodo debe de competir para ganar una oportunidad de transmitir su mensaje MSH – NENT hacia su vecino patrocinador. Este mensaje contiene la información del MSH – NENT: NetEntry Request en donde se envía el nodo ID de este nodo. Después que el nodo patrocinador recibe el mensaje, éste puede aceptar o rechazar el mensaje. Cuando el mensaje es rechazado este nodo envía un mensaje de MSH – NCFG con la información de MSH – NENT: NetEntryReject. Pero si el nodo acepta el mensaje el nodo patrocinador enviará un mensaje MSH – NCFG con la información de MSH-NCFG: NetEntryOpen.

Después de transmitir el mensaje MSH - NCFG: NetEntryOpen, el nuevo nodo debe transmitir el mensaje MSH - NENT: NetEntryAck hacia el nodo patrocinador, al terminar de mandar estos mensajes se inicia el proceso de negociación, en donde se cubren los siguientes puntos: el proceso de autorización y el proceso de registro.

Una vez que el nuevo nodo haya terminado con los procedimientos anteriores, éste debe transmitir el mensaje MSH - NENT: NetEntryClose para notificar al nodo patrocinador que los procesos necesarios y las transmisiones temporales han terminado.

Finalmente el nodo patrocinador transmitirá el mensaje MSH - NCFG: NetEntryAck hacia el nuevo nodo para terminar este proceso. Por lo que al ser recibido este mensaje, el nuevo nodo ya forma parte de la red [11].

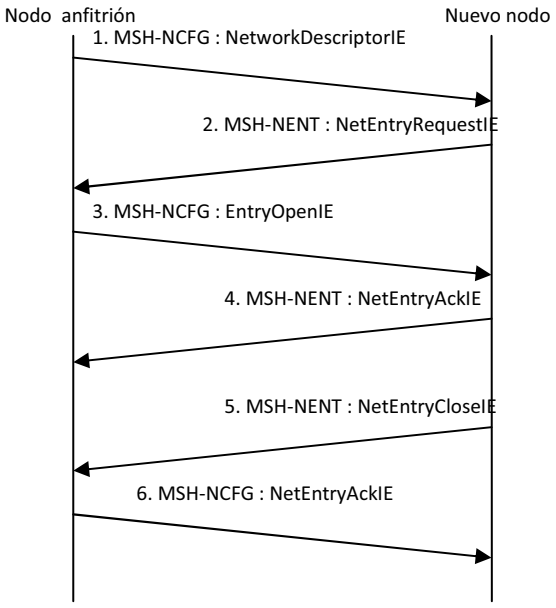


Figura 3.2. Procedimiento del proceso "Network Entry"

Diagrama de flujo del proceso de inicialización

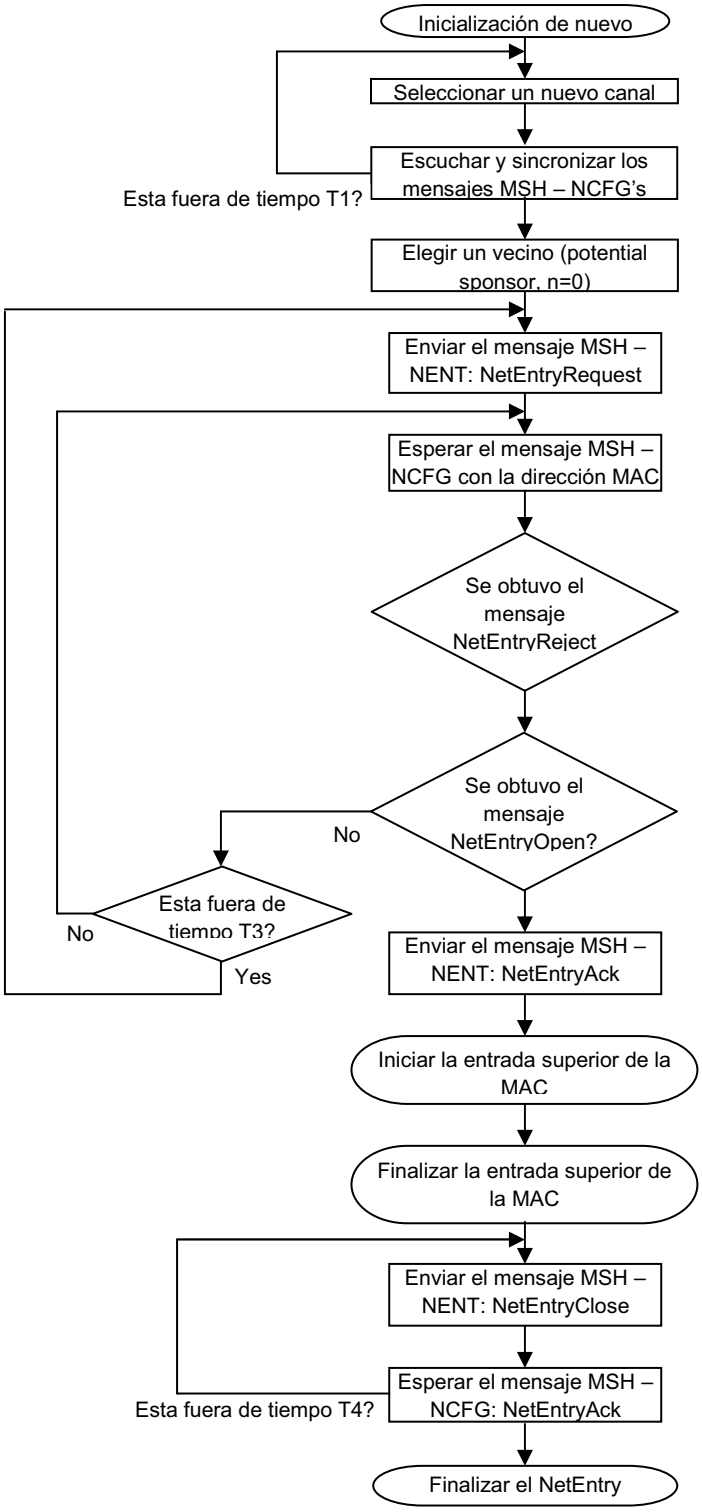


Figura 3.3. Diagrama de Flujo

Este diagrama de flujo muestra los pasos necesarios para iniciar el proceso de cada nodo, además muestra todos los 6 mensajes que se intercambian durante la inicialización de cada nodo, tanto los mensajes NENT y NCFG.

3.6.1 Mensajes MSH - NCFG

Este tipo de mensaje provee un nivel básico de comunicación entre los nodos en diferentes redes del mismo o de diferentes proveedores de equipos. Todos los nodos, ya sean BS o SS en la red Mesh transmiten este mensaje.

3.6.1.1 MSH – NCFG: Network Descriptor IE

Este tipo de mensaje lo envía el nodo patrocinador o bien la estación base hacia un nodo que va empezar el proceso de integración a la red.

Los campos de este mensaje se describen a continuación:

Syntax	Size
MSH-NCFG_embedded_data_IE() {	
Frame Length Code	4 bits
MSH-CTRL-LEN	4 bits
MSH-DSCH-NUM	4 bits
MSH-CSCH-DATA-FRACTION	4 bits
Scheduling Frames	4 bits
Num_Burst_Profiles	4 bits
Operator ID	16 bits
XmtEnergyUnitsExponent	4 bits
Channels	4 bits
MinCSFonrwardingDelay	7 bits
ExtendedNeighborhoodType	1 bit
if (Channels) MSH-NC FG_ Channel_IE()	*variable*
for (i=0: i < Num_Burst_Profiles :i++) {	
FEC Code Type	8 bits
Mandatory Exit Threshold	8 bits
Mandatory Entry Threshold	8 bits
}	
}	

Tabla 3.4. Encabezado del mensaje MSH-NCFG: Network Descriptor IE

3.6.1.2 MSH – NCFG: Entry Open IE

Este mensaje es usado para responder al mensaje MSH – NENT: NetEntry Request IE. El cual contiene los siguientes parámetros.

Syntax	Size
MSH-CNFG_embedded_data_IE() {	
Minislot Start	8 bits
Minislot Range	8 bits
Frame number	12 bits
Channel	4 bits
Schedule validity	12 bits

Channel	4 bits
Estimated Propagation Delay	4 bits
reserved	4 bits
}	

Tabla 3.5. Encabezado del mensaje MSH-NCFG: Entry Open IE

3.6.1.3 MSH – NCFG: NetEntry Ack IE

Este mensaje se genera para contestar a los mensajes MSH – NENT: NetEntry Close IE, con este mensaje se termina el proceso de alta a la red.

Syntax	Size
MSH-NCFG_embedded_data() {	
Extended embedded_data	1 bit
reserved	3 bits
Type	4 bits
Length	8 bits
Embedded data IE()	*variable*
}	

Tabla 3.6. Encabezado del mensaje MSH-NCFG: NetEntry Ack IE

Donde:
El campo de “type” es definido como: 0x4 Network Entry Ack y en este caso el campo embedded data IE es nulo.

3.6.2 Mensajes MSH – NENT

Este tipo de mensaje provee características para que un nuevo nodo gane sincronización e inicie Network Entry en la red Mesh.

3.6.2.1 MSH – NENT: NetEntry Request IE

Este mensaje se genera para contestar al mensaje MSH – NCFG: Network Descriptor.
Los campos se muestran a continuación:

Syntax	Size
MSH-NENT_Request_IE() {	
MAC Address	48 bits
OpConfInfo	64 bits
Operator Authentication Value	32 bits
Node serial Nuinber	32 bits
}	

Tabla 3.7. Encabezado del mensaje MSH-NENT: NetEntry Request IE

Donde:

MAC Address: Este campo contiene la dirección MAC del nuevo nodo.
OpConfInfo: Este campo describe la información de la configuración del operador.

Operator Authentication Value: Este campo tiene la siguiente estructura, HMAC {Dirección MAC | Número de serie del nodo | Llave}

3.6.2.2 MSH – NENT: NetEntry Ack IE

Mensaje que se genera para contestar al mensaje MSH – NCFG: Entry Open IE.

Los campos se describen:

Syntax	Size
MSH-NENT_Message_Format() {	
Management Message Type = 40	8 bits
Type	3 bits
Xmt counter for this Type	3 bits
reserved	2 bits
Sponsor Node ID	16 bits
Xmt Power	4 bits
Xmt Antenna	3 bits
reserved	1 bit
if (Type = 0x2)	
MSH-NENT_Request_IE()	variable
}	

Tabla 3.8. Encabezado del mensaje MSH-NENT: NetEntry Ack IE

Donde:

El campo “type” define el tipo de mensaje, en este caso 0x1 NetEntry Ack

3.6.2.3 MSH – NENT: NetEntry Close IE

Los campos se describen:

Syntax	Size
MSH-NENT_Message_Format() {	
Management Message Type = 40	8 bits
Type	3 bits
Xmt counter for this Type	3 bits
reserved	2 bits
Sponsor Node ID	16 bits
Xmt Power	4 bits
Xmt Antenna	3 bits
reserved	1 bit
if (Type = 0x2)	
MSH-NENT_Request_IE()	variable
}	

Tabla 3.9. Encabezado del mansaje MSH-NENT: NetEntry Close IE

Donde:

El campo “type” define el tipo de mensaje, en este caso 0x3 NetEntry Close

3.7 Arquitectura de Seguridad

La seguridad a través de la red inalámbrica de banda ancha ocurre mediante el encriptado de las conexiones entre las SS y BS.

La privacidad emplea un protocolo de autenticación cliente – servidor, en donde la BS controla la distribución de información a los SS's. Además, los mecanismos de privacidad son reforzados al control de protocolo de autenticación de SS´s basados en certificados digitales.

Existen dos tipos de protocolos de seguridad, los cuales son:

- Protocolo de encapsulación donde hay encriptación de paquetes a través de la red fija BWA.
- Protocolo de manejo principal (PKM), el cual realiza la distribución segura de datos desde la BS a la SS. Además la BS usa el protocolo para reforzar el acceso a los servicios de la red.

Encriptación de Paquetes

Los servicios de encriptado están definidos en un campo dentro de la subcapa de seguridad MAC. Por lo que la información específica de encriptado en el encabezado MAC está localizada en el formato de encabezado MAC genérico. El encriptado también es aplicado a la carga MAC – PDU, pero el encabezado MAC genérico no está encriptado.

Protocolo de Autenticación e intercambio de llaves (PKM)

Una SS usa el protocolo PKM para obtener autorización y tráfico de información codificada de la BS, además debe soportar autorizaciones periódicas y renovaciones de llaves. El protocolo PKM usa certificados digitales X.509 (IETF RFC 3280).

El protocolo PKM se añade al modelo cliente – servidor, en donde la SS requiere de llaves además la BS responde a estos requerimientos asegurando que un cliente SS recibe solamente el material de llaves para el cual está autorizado. También el protocolo PKM usa encriptación de llaves públicas para establecer una conexión segura compartida entre la SS y la BS. La información compartida es usada para asegurar el intercambio subsecuente de PKM de TEKs. Estos dos mecanismos son utilizados para la distribución de las llaves que permiten la actualización de los TEKs sin tener que utilizar encabezados de operaciones de llaves públicas.

Una BS autentica a un cliente SS durante el intercambio inicial de autorización. Cada SS lleva un certificado digital único X.509 emitido por el fabricante de la SS.

El certificado digital contiene la llave pública de las SSs y la dirección MAC de la SS. Cuando es requerido un AK, una SS presenta su certificado digital a la BS. La BS verifica su certificado digital y luego usa la llave publica verificada para encriptar el AK, el cual luego es reenviado por la BS a las SSs que los requirieron.

Todas las SSs deben tener instalado de fábrica un par de llaves RSA privadas/públicas o proveer un algoritmo interno para generar las llaves de forma dinámica. Si una SS confía en un algoritmo interno para generar su par de llaves de RSA, la SS debe generar el par de llave antes de su primer intercambio de AK. Todas las SSs con pares de llaves RSA instaladas en fábrica deben también tener instalada de fábrica los certificados X.509.

Todas las SSs que confían en algoritmo interno para generar un par de llaves RSA deben soportar un mecanismo para la instalación de certificados X.509 emitidos por fabricantes seguido de la generación de llaves.

Asociaciones de Seguridad

Una asociación de seguridad (SA) es un conjunto de información de seguridad que un BS y varios clientes SSs comparten. Existen tres tipos de SA definidos: Primario, Estático y Dinámico.

- Cada SS establece una asociación primaria de seguridad durante el proceso de inicialización de SS.
- Las SAs estáticas son provisionadas dentro de la BS,
- Las SA dinámicas son establecidas y eliminadas en respuesta a la iniciación y terminación de servicios específicos.

Ambas SAs, estáticas y dinámicas, pueden ser compartidas por múltiples SSs.

Una información de SA que es compartida debe incluir un conjunto de criptografía empleada dentro del SA. La información compartida puede incluir los vectores de inicialización y los TEKs. El contenido exacto de un SA es dependiente de conjunto de criptografía del SA.

Los SA son identificados usando SAID. Cada SS administrable puede establecer un SA primario exclusivo con sus BS. El SAID de cualquier SA primario de la SS debe ser igual al CID de la SS.

Usando el protocolo PKM una SS peticiona desde su BS el material de llave SA. El BS debe asegurar que cada cliente SS sólo posee acceso a los SAs si están autorizados al acceso.

El material de llave SA tiene un tiempo de vida limitado. Por lo que cuando el BS entrega el material de llave SA a una SS, también debe proveer al SS el tiempo de vida. Además es responsabilidad del SS de requerir nuevo material de llave a la BS antes de que el material de llave que actualmente tiene la SS se venza en la BS. El material de llave corriente debería vencer antes de que un nuevo conjunto sea recibido. Por último, el protocolo PKM especifica como la SS y la BS mantiene sincronización de llaves.

Capítulo 4

Algoritmo para Transmisión de Mensajes de Control

4.1 Introducción al algoritmo

Los mensajes de control son necesarios para que un nodo se pueda dar de alta en la red y para brindar mantenimiento a los nodos. Existen dos tipos de frames de control, el subframe de control de la red y el subframe de scheduling. Este capítulo abarcará la descripción del algoritmo de selección para transmitir los mensajes MSH – NCGF. Este algoritmo nos indica en qué momento pueda transmitir cada nodo para evitar colisiones en la red.

Con la configuración Mesh se utiliza también el algoritmo de selección, aunque es necesario hacer algunas variantes ya que es necesario que primero el nodo padre se de de alta en la red antes que el hijo. Esto se explicará más a detalle en el siguiente capítulo.

Además, este algoritmo es usado en el esquema descentralizado, de esta forma, cada nodo puede saber cuándo es necesario que realice su transmisión sin necesidad de que la estación base tenga que realizar estos cálculos.

Tal como se había descrito en el capítulo 3, los mensajes MSH – NCFG y MESH – NENT son indispensables para que se pueda dar de alta un nodo en la red. Para los mensajes MSH – NCFG hay que tener en cuenta los siguientes parámetros del mensaje de control MSH – NCFG (*Mesh Network Configuration Message*):

$$\textbf{Xmt Holdoff Exponent:}\ \text{Xmt Holdoff Time} = 2^{(\text{Xmt Holdoff Exponent} + 4)}$$

$$\textbf{Next Xmt Mx:}\ 2^{\text{Xmt Holdoff Exponent Next}} \bullet \text{Xmt Mx} < \text{Next Xmt Time} \leq 2^{\text{Xmt Holdoff Exponent}} \bullet (\text{Next Xmt Mx} + 1)$$

$$\textbf{Earliest Subsequent Xmt Time} = \text{Next Xmt Time} + \text{Xmt Holdoff Time}$$

En este capítulo solo se abarcará la forma en que opera el algoritmo de selección con el fin de poder plantear distintos escenarios de simulación posteriormente y optimizarlos de manera adecuada.

4.2 Transmisión del mensaje NCFG

Durante el tiempo actual XmtTime de cada nodo el nodo realiza el siguiente procedimiento para determinar su Next Xmt Time:

- Ordena su tabla de vecinos dependiendo su NextXmtTime (es el rango en el que pueden transmitir según las ecuaciones descritas anteriormente).
- Para cada entrada en la tabla, añade para cada nodo en la tabla de NextXmtTime su XmtHoldoffTime para determinar su EarliestSubsequentXmtTime.
- Determina si puede transmitir en ese momento o no puede.

Para determinar si puede transmitir en ese momento o no realiza lo siguiente:

- Determina la lista de nodos competidores, que son todos los nodos que tienen un rango de NextXmtTime que incluye por lo menos un valor en el mismo rango del nodo en cuestión.

- Aplica la función descrita en el estándar IEEE 802.16 *Mesh Election* aplicando el tiempo actual del nodo (de todo el rango posible según Next Xmt Time) y la lista de competidores (cada nodo tiene su NodeID asociado) de la siguiente manera:

MeshElection (TempXmtTime,MyNodeID,CompetingNodeIDsList [ ])

- Si el nodo no gana intenta transmitir en su siguiente oportunidad de transmisión y en base a su NextXmtTime.
- Si el nodo gana, transmite y además asigna su próximo valor para transmitir igual a Earliest SubsequentXmtTime.

Esto lo podemos ver gráficamente en la figura 4.1:

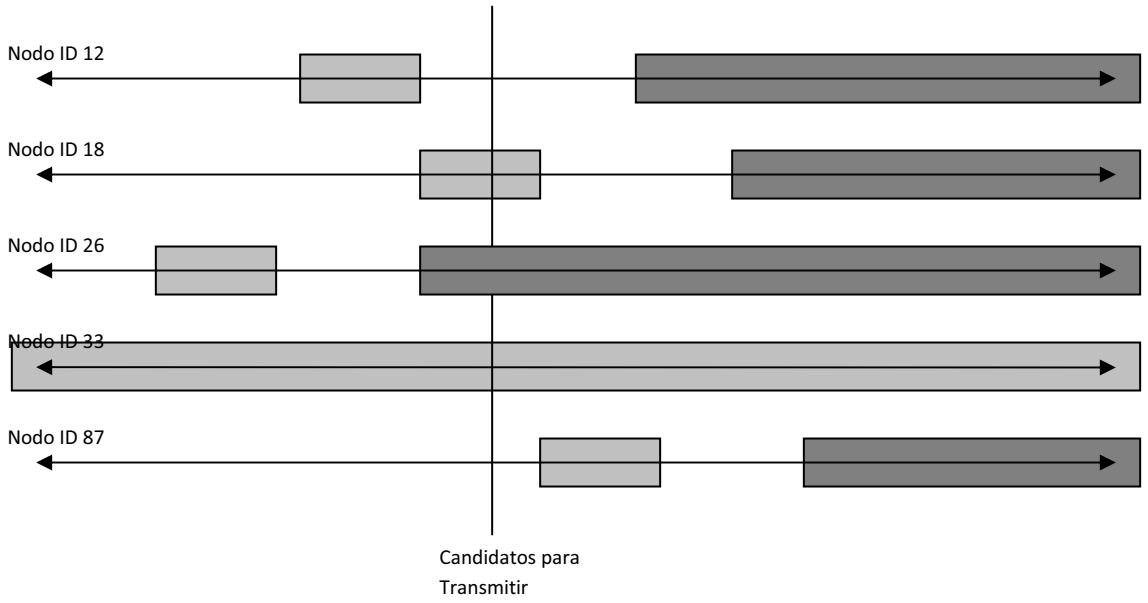


Figura 4.1. Diagrama de la transmisión de mensajes NCFG.

En este caso, Los rectángulos claros nos indican el rango NextXmtTime en el que un nodo puede transmitir. Los rectángulos oscuros nos indican su EarliestSubsequentXmtTime. Según la figura, y tomando en cuenta el tiempo de interés, podemos notar que los nodos 18, 26, y 33 son elegibles en ese rango. Cabe destacar que el nodo 26 abarca su Earliest Subsequent Xmt Time, en ese caso se volverán a calcular sus parámetros y se determinará si finalmente es un candidato a competir o no.

4.2.1 Algoritmo de Transmisión de Mensajes MSH – NCFG

El estándar IEEE 802.16 nos indica el algoritmo a seguir para poder mandar los mensajes MSH – NCFG de tal forma que sólo un nodo transmita a la vez y no existan colisiones. Además, nos brinda el pseudocódigo en c++ a seguir y nos lo indica como la función *Mesh Election*. Esta función determina cuando el nodo local es el ganador para un tiempo específico contra una cierta lista de competidores. Regresa un valor TRUE si el nodo es el ganador, en caso de que pierda regresa un valor de FALSE. El pseudocódigo es el siguiente:

```
boolean MeshElection (uint32 XmtTime,uint16 MyNodeID,uint16 NodeIDList [ ] )) {

        uint32 nbr_smear_val,smear_val1,smear_val2;
        smear_val1 =inline_smear(MyNodeID ^ XmtTime ));
        smear_val2 =inline_smear(MyNodeID +XmtTime );
        For each Node ID nbrsNodeID in NodeIDList Do {
                nbr_smear_val =inline_smear(nbrsNodeID ^ XmtTime ));
                if(nbr_smear_val >smear_val1 ) {
                        return FALSE;//Este nodo pierde.
                }
                else if(nbr_smear_val ==smear_val1 ) {
                        //1st tie-breaker.
                        nbr_smear_val =inline_smear(nbrsNodeID +XmtTime );
                        if(nbr_smear_val >smear_val2 ) {
                        return FALSE;//Este nodo pierde.
                        }

                        else if(nbr_smear_val ==smear_val2 ) {
                                //If we still collide at this point Break the tie based on MacAdr
                                if ((XmtTime is even &&(nbrsNodeID >MyNodeID))||
                                (XmtTime is odd &&(nbrsNodeID <MyNodeID ))) {
                                return FALSE;//Este nodo pierde.
                                }
                        }
                }
                // Este nodo gana sobre un competidor.
        }//Final para todos los nodos competidores-
        .
//Este nodo es el ganador sobre todos los demás nodos.
return TRUE;
}
```

Posteriormente nos indica la función para hacer los valores pseudoaleatorios. Consiste en una función que realiza el corrimiento en bits. La función es la siguiente:

```
uint32 inline_smear(uint16 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);
}
```

El objetivo es implementar este algoritmo con algún software para poder determinar cómo es su comportamiento, es decir, planteado distintos escenarios de simulación y notar cómo variando los distintos parámetros como **Xmt Holdoff Exponent** y **Next Xmt Mx** se obtienen resultados más rápidos. Para esto, y con fines de facilitar su futura implementación usando el software de OPNET Modeler, se decidió implementarlo usando el lenguaje de programación C++.

OPNET es un programa de simulación de redes que proporciona las herramientas necesarias para implementar el comportamiento del estándar. Una de las razones por las que se utiliza esta herramienta es por su sencillez de programación con el interfaz gráfico, además de que cuenta con un modelo para redes que operan bajo el protocolo IEEE 802.16. OPNET cuenta con distintas versiones dependiendo del mercado al que va dirigido, entre ellas se encuentran: ITDecision GURU usado en medianas y grandes empresas con redes de área amplia, el Netbiz usado por proveedores de servicio y vendedores de equipo en donde sus principales

requerimientos incluyen completas soluciones de red y OPNET Modeler (software a utilizar) un software de prototipos virtuales capaz de incrementar la productividad, la calidad de productos, evaluar el desempeño de cualquier red de comunicaciones ya sea fija o móvil, inalámbrica o cableada, así como el plantear distintos escenarios para implementar mejoras o visualizar los efectos de alguna modificación en ella.

4.3 Papel de las distintas variables dentro del algoritmo

A continuación se dará una explicación de las principales variables utilizadas en el algoritmo de elección. Estas variables juegan un papel importante para poder aplicar el algoritmo a una red.

Las variables son:

Número de nodos
Hold off Exponent
Next Xm Mx
Earliest Subsequent Xmt Time
Estructura del Frame Mesh
Número de oportunidades de transmisión NCFG
Duración de un time slot, tamaño del símbolo OFDM.

4.3.1 Número de nodos

La variable número de nodos hace referencia al “ID” de cada nodo, con el fin de tener identificados cada nodo y así llevar un mejor control sobre éstos.

Nodos primer nivel

En la siguiente imagen se aprecia que cada nodo lleva un “Nodo ID” para identificarlos, además se observa que los primeros 10 forman el primer nivel porque están dentro de un rango de la estación base, es decir, estos nodos se comunican primero con la estación base para tener acceso a la red Mesh.

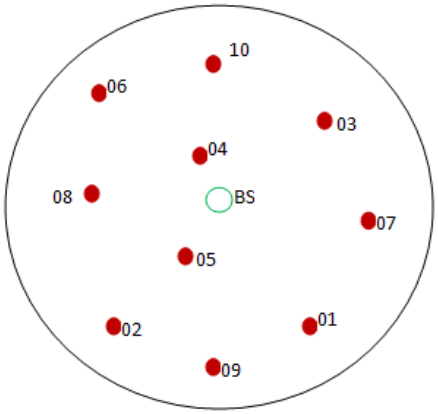


Figura 4.2. Nodos conectados a la BS.

Nodos segundo nivel

Los nodos del segundo nivel también cuentan con un identificador “Nodo ID”, pero estos nodos van estar ligados a cada nodo del primer nivel, es decir, los nodos del primer nivel tendrán asociados a tres nodos del segundo nivel de acuerdo a nuestro modelos que implementamos.

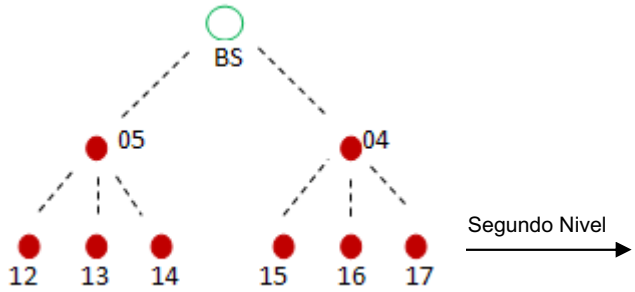


Figura 4.3. Distribución de nodos en el segundo nivel.

Nodos tercer nivel

Estos nodos también cuentan con "Nodo ID" y van asociados a los nodos del segundo nivel, de acuerdo a nuestro modelo tendremos dos nodos asociados a los nodos de segundo nivel como se muestra en la siguiente imagen.

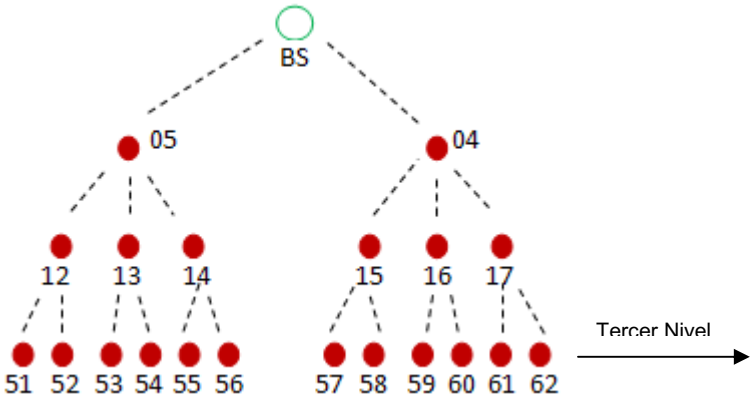


Figura 4.4. Distribución de nodos en el tercer nivel.

De acuerdo a la distribución de los nodos en el algoritmo contamos con 100 nodos en total para el escenario final de simulación que se verá en el capítulo 5, todos estos cuentan con un "Nodo ID" para poder aplicar el algoritmo de elección y así saber que nodo podrá transmitir mensajes NCFG y NENT en cierto tiempo.

4.3.2 Xmt Holdoff Exponent

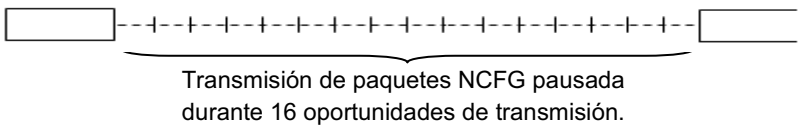
La variable Holdoff Exponent dentro del algoritmo es un valor que tiene un valor máximo de 3 bits, por lo tanto sólo se pueden ocupar los valores del 0 hasta el 7. Este término afecta directamente al Xmt Holdoff Time, esto quiere decir, que dependiendo el valor que tenga el Holdoff exponent la variable Holdoff Time será menor o mayor, es decir, que el tiempo de espera para transmitir paquetes MSH-NCFG depende de esta variable.

La ecuación que describe las variables anteriores es la siguiente:

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

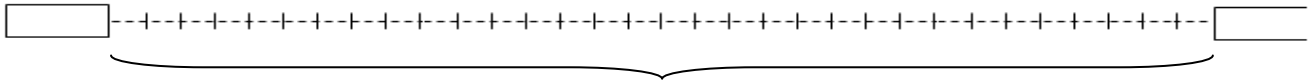
Si Holdoff Exponent = 0

$$Xmt\ Holdoff\ Time = 2^4 = 16$$



- Si Holdoff Exponent = 1

$$Xmt\ Holdoff\ Time = 2^5 = 32$$



Transmisión de paquetes NCFG pausada
durante 32 oportunidades de transmisión

De los ejemplos anteriores podemos observar que mientras mayor sea el valor de Holdoff Exponent, mayor será el tiempo de pausa para la transmisión de NCFG. Así que para aprovechar mejor las oportunidades de transmisión se sugiere utilizar valores de Holdoff Exponent pequeños, aunque a veces se logran mejores resultados los valores mayores esto depende de otras variables.

4.3.3 Next Xm Mx

La variable Next Xm Mx está formada por 5 bits, por lo que los valores que se pueden utilizar en esta variable son del 0 al 31. Además la variable Next Xmt Time depende de las variables Next Xm Mx y Holdoff Exponent.

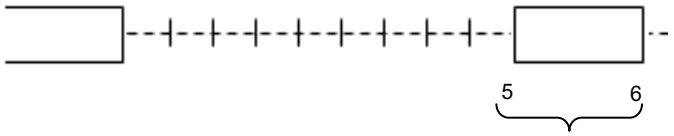
La ecuación que da el valor de Next Xmt Time es la siguiente:

$$(2^{Xmt\ Holdoff\ Exponent})(Next\ Xmt\ Mx) < Next\ Xmt\ Time \leq (2^{Xmt\ Holdoff\ Exponent})(Next\ Xmt\ Mx + 1)$$

Esta variable sirve para encontrar el intervalo de tiempo en el cual los mensajes NCFG se transmitirán, este intervalo lo describe la variable Next Xmt Time, pero el Next Xmt Mx depende directamente para encontrar el intervalo de tiempo en el cual se puede transmitir.

- Si Holdoff Exponent = 0 y Next Xmt Mx = 5

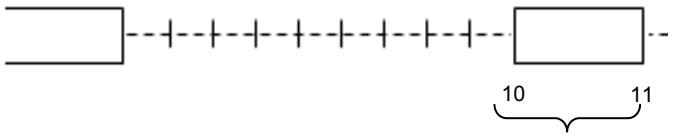
$$5 < Next\ Xmt\ Time \leq 6$$



Intervalo elegible para transmitir
mensajes NCFG (5 – 6)

- Si Holdoff Exponent = 0 y Next Xmt Mx = 10

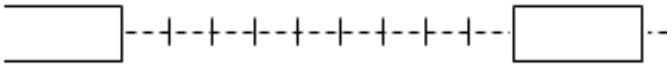
$$10 < Next\ Xmt\ Time \leq 11$$



Intervalo elegible para transmitir
mensajes NCFG (10 – 11)

- Si Holdoff Exponent = 1 y Next Xmt Mx = 5

$$10 < Next\ Xmt\ Time \leq 12$$

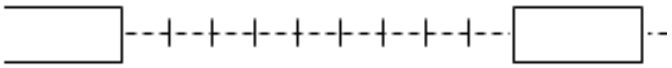


10 12

Intervalo elegible para transmitir mensajes NCFG (10 – 12)

Si Holdoff Exponent = 1 y Next Xmt Mx = 10

$$20 < Next\ Xmt\ Time \leq 22$$



20 22

Intervalo elegible para transmitir mensajes NCFG (20 – 22)

En los ejemplos anteriores observamos que mientras el Holdoff Exponent y Next Xmt Mx sean mayores el intervalo elegible para transmitir los mensajes NCFG es mayor, por lo que el tiempo tiene una mayor rango de elegir una oportunidad de transmisión válida para transmitir NCFG.

4.3.4 Earliest Subsequent Xmt Time

El papel principal de esta variable es determinar el intervalo de transmisión de cada nodo una vez que estos transmitieron por primera vez en el intervalo de Next Xmt Time, es decir, el Earliest Subsequent Xmt Time da el inicio de transmisión del nodo cuando ya transmitió su primer mensaje.

Esta variable se calcula con la siguiente fórmula:

$$EarliestSubsequentXmtTime = NextXmtTime + 2^{XmtHoldoffExponent+4}$$

En la siguiente figura se observa claramente la localización de esta variable.

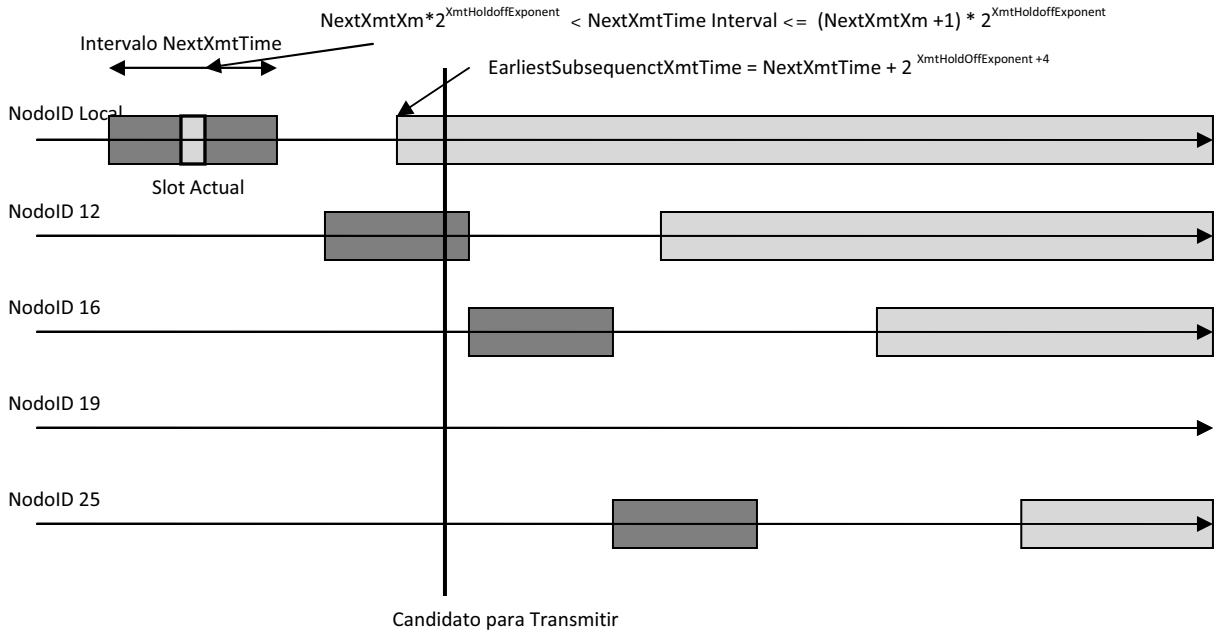


Figura 4.5. En esta figura se muestra la localización de la variable Earliest Subsequent Xmt Time

4.3.5 Estructura del Frame Mesh

De acuerdo al estándar 802.16 – 2004 un frame está compuesto de una sección de control y una de datos. La tesis está enfocada en la sección de control, por lo que se describirá como está formada esta sección y como se utilizó en la implementación del algoritmo de elección de tiempos.

En primer lugar hay dos tipos de subframe de control y se divide en: subframe de control de red y subframe de scheduling, a su vez el subframe de control se divide en los siguientes mensajes: *Mesh Network Configuration (MSH – NCFG) y Mesh Network Entry (MSH – NENT),* estos mensajes son indispensables para darse de alta a la red mesh de WiMAX, además se utilizan para mantener la configuración de la red mediante sincronizaciones continuas. Además estos mensajes se dividen en oportunidades de transmisión, que son intervalos de tiempo en el cual los mensajes MSH – NCFG y MSH – NENT se transmiten para que los nodos se den de alta a la red o bien para mantenerse sincronizados.

A continuación se ilustra la estructura de un frame con el subframe de control y el subframe de datos.

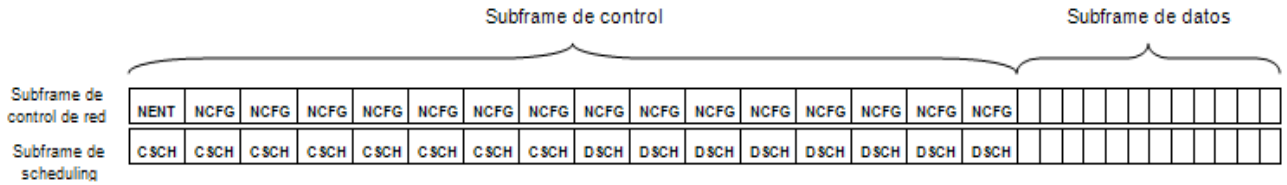


Figura 4.6. En la figura se observa que el subframe de control de subdivide en dos subframes, control de red y scheduling, además se observa el subframe de datos.

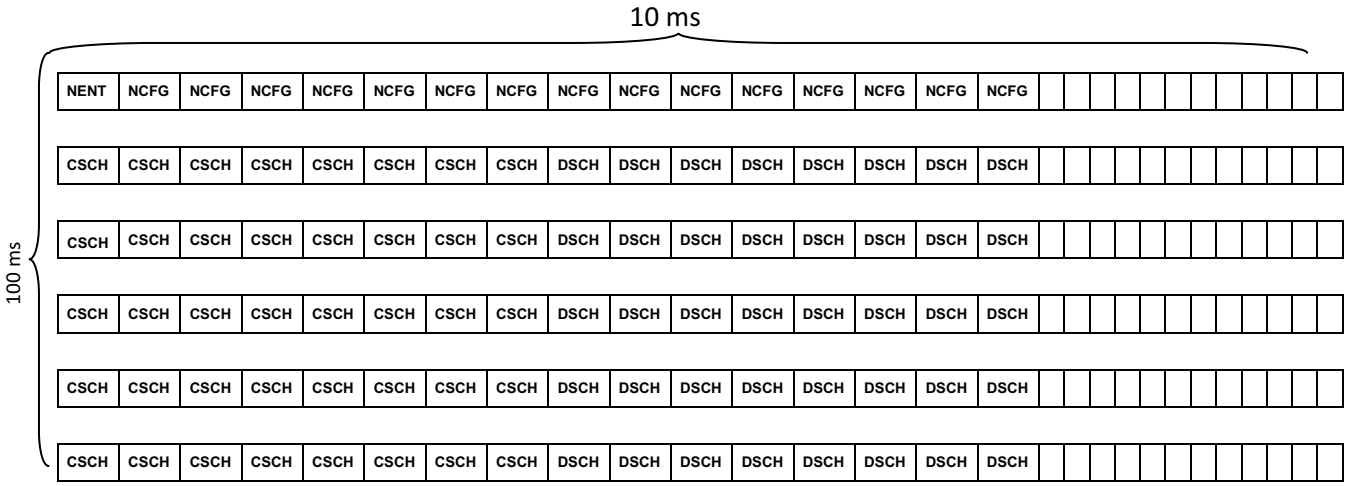


Figura 4.7. En la figura se observa que el frame tiene una duración de 100 ms y que la unión del subframe de control y datos tiene una duración de 10 ms.

De acuerdo al estándar 802.16, los frames deben tener como máximo 15 oportunidades de transmisión para los mensajes NENT y NCFG, es decir, se puede utilizar cualquier configuración de oportunidades de transmisión mientras se cumpla el estándar. Para transmitir más mensajes es necesario utilizar otro frame de acuerdo a la figura anterior.

El algoritmo de elección desarrollado utiliza 8 oportunidades de transmisión en el subframe de control de la red, una oportunidad para los mensajes NENT y 7 oportunidades para los mensajes NCFG, en la siguiente figura observamos de forma gráfica el frame que utilizamos para diseñar el algoritmo de elección.

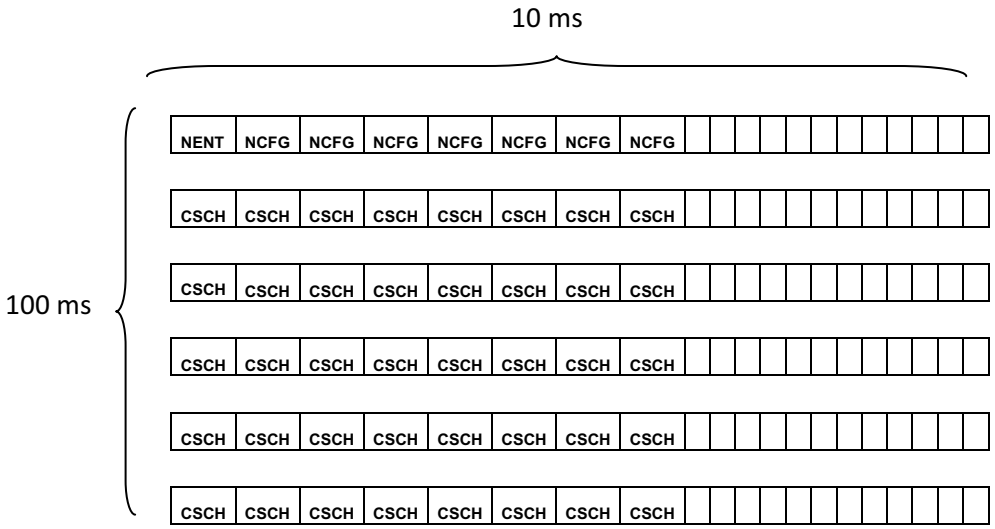


Figura 4.8. Frame utilizado en el diseño del algoritmo de elección.

4.3.6 Número de oportunidades de transmisión NCFG

De acuerdo al estándar WiMAX, 802.16 – 2004, las oportunidades de transmisión se definen en el subframe de control en la red, es decir, en el subframe donde se localizan los MSH – NENT y MSH – NCFG. En cambio, en el subframe de datos la división se hace en términos de minislots.

Las oportunidades de transmisión son usadas para llevar los mensajes de señalización a la configuración de la red. Pero para visualizar las oportunidades de transmisión dentro del algoritmo fue necesario realizar unas ecuaciones de conversión de tiempos, ya que al aplicar el algoritmo los resultados obtenidos siempre son en función de las oportunidades de transmisión y para tener más claro los tiempos de transmisión de cualquier nodo se transformó estas oportunidades en segundos. Estas conversiones se observan en el tema 4.3.6.

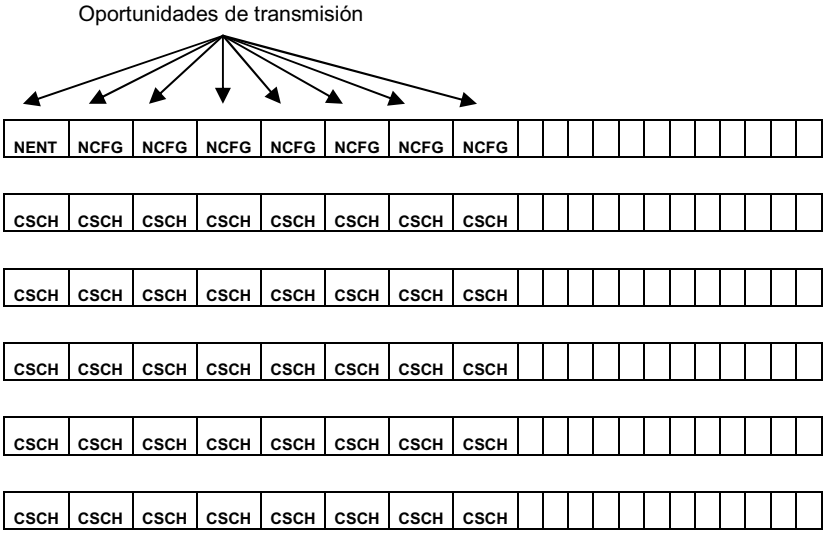


Figura 4.9. Oportunidades de transmisión.

4.3.7 Duración de un time slot, tamaño del símbolo OFDM.

Los parámetros principales que utilizamos para diseñar el algoritmo de elección fueron la duración del subframe, No. De minislots/frame, símbolos OFDM/frame, ya que estos sólo son parámetros que se encuentran en la capa física y la capa MAC de WiMAX.

Como el algoritmo diseñado trabaja en la capa MAC fue necesario tomar los siguientes valores definidos en el estándar para poder adecuar el algoritmo y obtener resultados óptimos:

Duración del subframe = 10 [ms]
Símbolos OFDM / frame = 1024
No. de minislots / frame = 256

De estos datos se calculan algunas relaciones indispensables para conocer exactamente el tiempo en que los nodos se comunicarán con sus nodos vecinos o bien con la estación base.

1024 símbolos OFDM = 10 ms.
7 símbolos OFDM = 68.359 µs.
1 Oportunidad de transmisión = 7 símbolos.
1 Oportunidad de transmisión = 68.359 µs.
7 símbolos OFDM = 504 bytes
1 símbolo OFDM = 72 bytes.

De la relación anterior se obtiene que una oportunidad de transmisión es igual a 68.359 µs, por lo que un mensaje NCFG y NENT se transmite en un intervalo de [0 – 68.359 µs]

4.4 Implementación del algoritmo.

4.4.1 Función para Determinar el Nodo Ganador

Después de ver cuáles son los parámetros óptimos para empezar a realizar simulaciones, de tal forma que se obtengan tiempos de transmisión más cortos para los mensajes MSH – NCFG, fue necesario empezar a implementar el modo en que se transmiten los mensajes y cómo es que se comporta el algoritmo. Lo primero que fue necesario realizar fue la adaptación de pseudocódigo que el estándar nos brinda. Esta adaptación del pseudocódigo se realizó utilizando el lenguaje de programación C++, como se mencionado anteriormente, su ventaja es que se podrá implementar en el software de simulación de redes.

El código que finalmente se usó se puede observar en el Anexo 1. Este código fue utilizado para verificar en qué tiempo le toca transmitir a cada nodo. Podemos observar la función *elected*, equivalente a la función que nos da el estándar.

Hay cuatro parámetros que nos indicarán cuál es el nodo ganador, el tiempo de simulación, el nodo de interés, el número de nodos que van a competir, y la lista de nodos que van a competir (con el identificador de cada nodo *NodeID*). Con el tiempo de simulación podemos ver que no necesariamente obtenemos el mismo resultado cuando una misma lista de nodos compite en diferentes tiempos, puede haber distintos ganadores cuando el tiempo cambie.

El segundo parámetro, el nodo de interés, nos indica el ID del nodo. Estos valores se propusieron de tal forma que el número de nodo es igual al *NodeID* para su fácil identificación, es decir, conforme se agregue un nodo este tendrá un *NodeID* sucesivo para tener una serie consecutiva. El *NodeID* está incluido dentro de la lista de competidores. Con esta consideración, el nodo de interés competirá contra sí mismo, pero debido a las restricciones que se le ponen, no se realiza la competencia y posteriormente compite con el siguiente nodo en la lista.

El tercer parámetro es la cantidad de nodos que van a competir y depende del cuarto parámetro que es la lista de nodos que va a competir. En esta lista se toma su identificador. La forma en que se realiza la competencia es que el nodo de interés compite contra el primer nodo de la lista, si gana, compite contra el siguiente nodo de la lista. Una vez que haya competido con todos los nodos podemos decir que ese nodo ha ganado la oportunidad de transmisión del mensaje MSH – NCFG. Si el nodo pierde contra alguno de los nodos deja de competir.

Esta función utiliza las variables smear_val1, smear_val2 y nbr_smear_val las cuales utilizan corrimientos en bits para realizar la decisión para determinar al nodo ganador.

4.4.2 Función *inline_smear*

El objetivo de esta función es realizar un corrimiento en bits a la izquierda y a la derecha para poder tener los valores pseudoaleatorios que se requieren según el estándar.

Cuando se realiza un desplazamiento en bits << η se desplaza todos los bits η posiciones a izquierda. Cuando se realiza >> η se desplaza todos los bits η posiciones a derecha. Es necesario aclarar que η nunca puede ser negativo, ya que daría error.

Cada vez que se hace un desplazamiento se completa con ceros en el otro lado (no se trata de una rotación). En el caso de un valor con signo, como se representa con un “1” en el bit de orden superior, cada desplazamiento a derecha se introduce un 1 por izquierda. Por ejemplo:

0 0 1 1 << 2 -> 1 1 0 0 >> 1 -> 0 1 1 0

Cada desplazamiento a izquierda, es una multiplicación por 2. Cada desplazamiento a derecha, es una división por 2. La ventaja de los corrimientos es que son menos costosos que hacer las operaciones, de multiplicación y división.

Además, otra de las operaciones que cabe aclarar en este punto es OR exclusivo (^).

XOR	0	1
0	0	1
1	1	0

Si el primer operando ó el segundo operando pero no ambos, es uno, el resultado es uno, de lo contrario el resultado es cero. Si uno de los operandos en la operación lógica OR exclusivo es uno, el resultado es siempre el inverso del otro operando. Ésta característica le permite invertir bits selectivamente en una cadena de bits.

La función la podemos ver a detalle en el anexo 1.

4.4.3 Función Para Convertir el Tiempo

La forma en que se implementó el código fue tomando en cuenta las oportunidades de transmisión (TO) en el que un nodo puede transmitir. Se maneja de tal forma que se asigna la primer TO a un nodo, la segunda TO a otro nodo y así sucesivamente. Por esta razón, es necesario convertir el tiempo de la TO asignada al tiempo real para tener una mejor

Esta función depende del número de oportunidades de transmisión para transmitir mensajes MSH – NCFG, de tal forma que si estamos usando 7 oportunidades de transmisión, se generarán tiempos más altos que si se manejan 15 oportunidades de transmisión. Esto lo podemos ver debido al frame de transmisión.

Otro de los parámetros usados en el tiempo de duración de los siete símbolos OFDM que se manejó como una constante. La forma en que se implementó nos regresa un valor real y es muy útil para determinar realmente

cómo se comporta el algoritmo y las distintas variables que nos pueden optimizar el tiempo en que un nodo se registra.

Gráficamente, podemos ver el en la figura 4.10 un esquema de transmisión de un frame con 7 oportunidades de NCFG. Hay que recordar que, después de un frame de control, vienen *N* frames de scheduling, tal como los describe el estándar. En nuestro caso consideramos 9 frames de scheduling, por lo que transcurren 100 milisegundos.

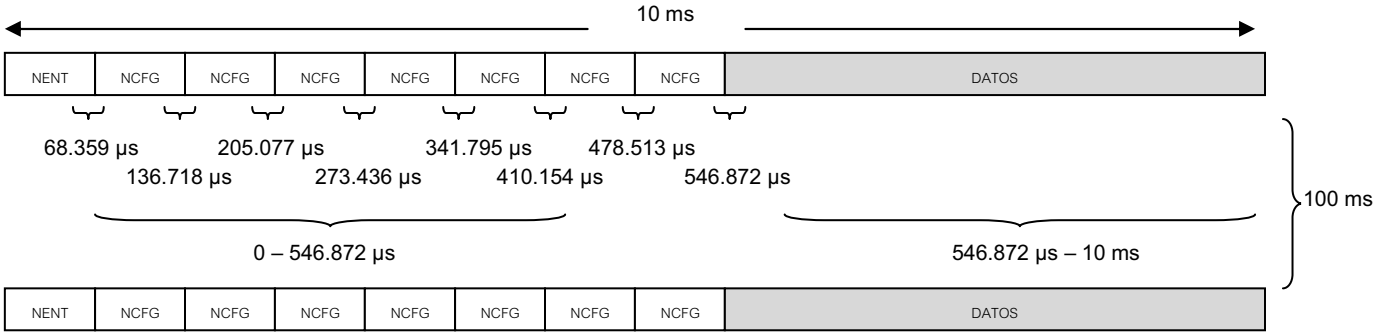


Figura 4.10. Esquema de transmisión de un frame con 7 OT.

Las ecuaciones que se obtuvieron después de analizar la estructura de los frames y que se implementó en el código C++ son las siguientes:

$$Tiempo = x(100\ ms) + y(68.359\ \mu s)$$

Donde:

$$x = \frac{TO}{NumerodeNCFG}$$

$$y = TO - x(NumerodeNCFG)$$

Estas ecuaciones nos indican a qué tiempo le corresponde un TO. Por ejemplo, para la TO = 1; x = 0 (recordemos que **sólo se toma la parte entera**); y = 1 y el Tiempo = 68.359 µs. Ahora para TO = 7; x = 0; y = 7 y Tiempo = 478.513 µs.

Supongamos ahora un cambio de frame, TO = 8; x = 1; y = 1; Tiempo = 0.100068359 s que es el resultado que esperamos ya que entre frame y frame existen 100 ms. Ahora podemos ir hasta el final de ese frame que le corresponde una TO = 14; x = 1; y = 7; Tiempo = 0.100478513 s que corresponde al final del segundo frame. De esta forma, siempre obtendremos el tiempo deseado con estas ecuaciones implementadas.

En el caso de que se decida cambiar a 12 oportunidades de transmisión en el subframe de control (11 de NCFG y 1 de NENT) o a 16 oportunidades sólo hay que especificar esto en las ecuaciones de tal forma que el comportamiento va a ser el mismo aunque se obtienen tiempos menores. Los tiempos se reducen a la mitad cuando en vez de usar 7 oportunidades de transmisión se usan 15 oportunidades. A su vez, los tiempos se reducen una cuarta parte cuando se usan 11 oportunidades en vez de 7 oportunidades.

4.5 Análisis del Algoritmo

4.5.1 Funcionamiento del algoritmo

Existen numerosos escenarios de simulación que se pueden establecer para analizar el algoritmo, todos estos dependen de los parámetros mencionados en el punto 4.3. Para ver realmente cómo es que se comportan las funciones mencionadas anteriormente podemos plantear un escenario donde todos los nodos compitan en el mismo tiempo para determinar el nodo ganador.

Si asignamos el mismo valor de NextXmMx (se asigna cero) y de HoldoffExponent (se asigna cero) lograremos que todos los nodos tengan las mismas ventanas de transmisión para poder hacer que todos compitan en todo momento. Se consideran 10 nodos para este caso. Obtenemos las siguientes ventanas para todos los nodos:

```
XmtHoldoffTime = 16
NextXmtMx = 0
NextXmtTime1 = 1
NextXmtTime2 = 1
EarliestSubsequentXmtTime = 17
```

Lo que gráficamente podemos verlo como en la figura 4.11. En el tiempo 17 todos los nodos volverán a calcular sus parámetros de transmisión dando los mismos resultados debido a sus valores.

Figura 4.11. Distribución de los tiempos para transmitir mensajes NCFG y NENT.

En este caso se obtiene el siguiente proceso de competencia:

```
nbr_smear_val[1] = 231855697 > smear_val1[2] = 0. Este nodo (1) pierde contra 2!
nbr_smear_val[2] = 1582626671 > smear_val1[3] = 231855697. Este nodo (2) pierde contra 3!
nbr_smear_val[3] = 3165253471 > smear_val1[5] = 1582626671. Este nodo (3) pierde contra 5!
nbr_smear_val[4] = 1582626671 > smear_val1[3] = 311529801. Este nodo (4) pierde contra 3!
nbr_smear_val[5] = 3521949857 > smear_val1[10] = 3165253471. Este nodo (5) pierde contra 10!
nbr_smear_val[6] = 3165253471 > smear_val1[5] = 1814290542. Este nodo (6) pierde contra 5!
nbr_smear_val[7] = 1582626671 > smear_val1[3] = 463711522. Este nodo (7) pierde contra 3!
nbr_smear_val[8] = 1582626671 > smear_val1[3] = 369291516. Este nodo (8) pierde contra 3!
nbr_smear_val[9] = 3165253471 > smear_val1[5] = 2034491070. Este nodo (9) pierde contra 5!

El primer desempate
nbr_smear_val[10] = 3521949857 == smear_val1[10] = 3521949857!
Fin de la competencia

GANO EL NODO 10 EN EL TIEMPO 1 (0.0000683590 [s])
```

Donde se puede observar el funcionamiento del algoritmo, en este caso el nodo ganador fue el nodo 10. Si dejamos correr más tiempo la simulación, en el tiempo 17 es necesario volver a calcular los tiempos de transmisión de todos los nodos y seleccionan el tiempo 18. En ese tiempo el nodo ganador es el 2.

4.5.2 Optimización de los valores para obtener mejores resultados

4.5.2.1 Holdoff Exponent y Número de Nodos

Es necesario tomar en cuenta el Holdoff Exponent que nos indica el tamaño de la ventana de transmisión para que un nodo en la red Mesh pueda transmitir. Además, el Holdoff Exponent nos brinda un parámetro para poder saber cuánto tiempo después de la ventana de transmisión un nodo permanece en un estado donde no transmite ningún dato.

Es necesario notar que, el número de nodos y el Holdoff Exponent están relacionados de tal forma que ajustando estos parámetros se pueden obtener tiempos de registro de los nodos en la red más pequeños. Como se vio en el apartado 4.3.2 el tamaño de la ventana de transmisión varía según el Holdoff Exponent, aunque si añadimos el parámetro del número de nodos de la red podemos ver el escenario de la figura 4.12.

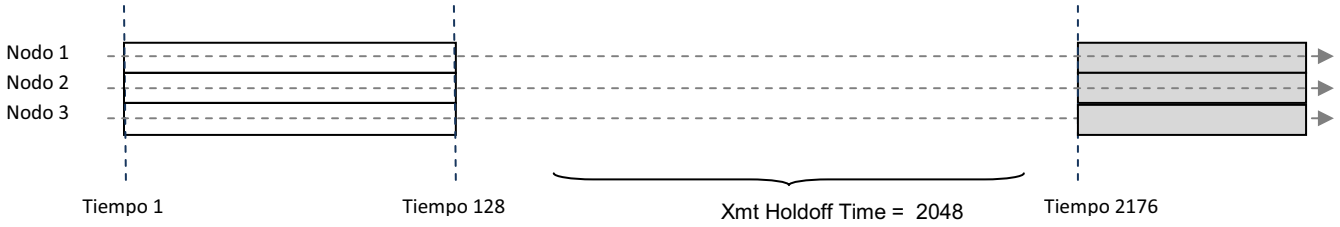


Figura 4.12. Variación de la ventana de transmisión con un Holdoff Exponent=7.

En este caso, se usó un valor de NextXmMx = 0 para los 3 nodos en cuestión y un **Holdoff Exponent = 7**. Podemos notar que el tiempo Xmt Holdoff Time es muy grande y se estaría esperando mucho tiempo. Podemos decir que este caso no es el más óptimo. En contraste, podemos analizar el escenario de la figura 4.13.

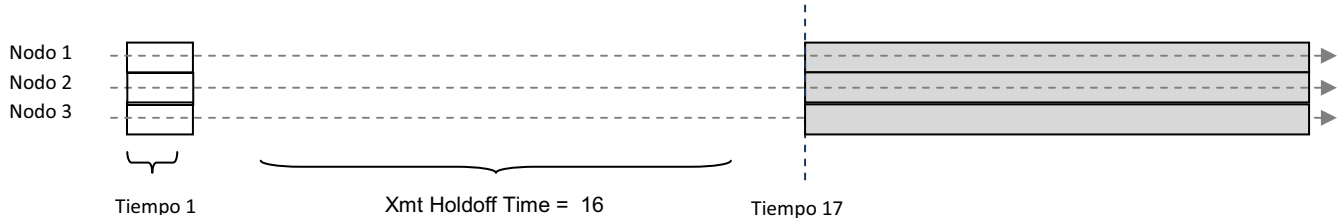


Figura 4.13. Ventana de transmisión con un Holdoff Exponent=0.

En este caso, se usó un valor de NextXmMx = 0 para los 3 nodos en cuestión y un **Holdoff Exponent = 0**. Comparando los dos casos podemos notar que en el segundo escenario se presentan tiempos mucho menores y más adecuados para el número de nodos que se está analizando.

Imaginemos que la red tiene 100 nodos. En este caso parece conveniente seguir usando un Holdoff Exponent pequeño para obtener tiempos de registro de los nodos más pequeños, aunque, como se verá en el siguiente capítulo, conviene usar Holdoff off Exponent más grande para evitar lo más posible las competencias entre los nodos y así, obtener un patrón más definido de comportamiento.

Como conclusión, podemos afirmar que mientras se tenga un número pequeño de nodos se obtienen tiempos de registro menores usando un Holdoff Exponent pequeño. Cuando se incrementa el número de nodos conviene usar un Holdoff Exponent más grande para obtener tiempos de registro más pequeños. Los parámetros óptimos se discutirán en el siguiente capítulo.

4.5.2.2 Distribución de NextXmMx

Además de las consideraciones que se hicieron en el punto anterior, es necesario asignar los valores de NextXmMx de tal forma que los tiempos de registro se obtengan mayores. Para esto hay que analizar el comportamiento de este valor. Recordemos que es un valor de 5 bits por lo que toma valores del 0 al 32. Este

valor nos indica prácticamente el tiempo que se recorre la ventana de transmisión de datos entre un nodo y otro. Es decir, puede que dos nodos tengan los mismos valores de Holdoff Exponent pero que tengan diferentes valores de NextXmMx, en este caso, tendrán la misma duración de la ventana de transmisión y el mismo tamaño de ventana tiempo en el que no transmitirán, pero esos tiempos serán distintos.

En este caso, podemos plantear el peor escenario de simulación tal como en la figura 4.14. Los valores de Holdoff Exponent son de cero para todos los nodos. Todos estos nodos tienen el mismo valor de NextXmMx[1].

Figura 4.14. Ilustración del comportamiento de los nodos con el mismo Holdoff Exponent.

Analizando este último escenario y tomando en cuenta que la ventana de transmisión es de una sola oportunidad y después se duermen todos los nodos 16 tiempos podemos ver que sólo uno resultará ganador y todos los demás se tendrán que esperar, **desperdiciando esos 16 tiempos de transmisión por cada oportunidad de transmisión.**

Cuando corremos el programa que se encuentra en el anexo 1 obtenemos los siguientes resultados:

```
Nodo: 1
 XmtHoldoffTime = 16
 NextXmtMx = 0
 NextXmtTime1 = 1
 NextXmtTime2 = 1
 EarliestSubsequentXmtTime = 17

Nodo: 2
 XmtHoldoffTime = 16
 NextXmtMx = 0
 NextXmtTime1 = 1
 NextXmtTime2 = 1
 EarliestSubsequentXmtTime = 17

Nodo: 3
 XmtHoldoffTime = 16
 NextXmtMx = 0
 NextXmtTime1 = 1
 NextXmtTime2 = 1
 EarliestSubsequentXmtTime = 17

Nodo: 4
 XmtHoldoffTime = 16
 NextXmtMx = 0
 NextXmtTime1 = 1
```

```
Nodo: 6
 XmtHoldoffTime = 16
 NextXmtMx = 0
 NextXmtTime1 = 1
 NextXmtTime2 = 1
 EarliestSubsequentXmtTime = 17

Nodo: 7
 XmtHoldoffTime = 16
 NextXmtMx = 0
 NextXmtTime1 = 1
 NextXmtTime2 = 1
 EarliestSubsequentXmtTime = 17

Nodo: 8
 XmtHoldoffTime = 16
 NextXmtMx = 0
 NextXmtTime1 = 1
 NextXmtTime2 = 1
 EarliestSubsequentXmtTime = 17

Nodo: 9
 XmtHoldoffTime = 16
 NextXmtMx = 0
 NextXmtTime1 = 1
```

[1] *No se incluyen las representaciones gráficas del siguiente tiempo para transmitir como en los casos anteriores.*

```
 NextXmtTime2 = 1
 EarliestSubsequentXmtTime = 17

Nodo: 5
 XmtHoldoffTime = 16
 NextXmtMx = 0
 NextXmtTime1 = 1
 NextXmtTime2 = 1
 EarliestSubsequentXmtTime = 17
```

```
 NextXmtTime2 = 1
 EarliestSubsequentXmtTime = 17

Nodo: 10
 XmtHoldoffTime = 16
 NextXmtMx = 0
 NextXmtTime1 = 1
 NextXmtTime2 = 1
 EarliestSubsequentXmtTime = 17
```

Lo cual nos refleja el mismo panorama que la figura 4.14. Hay que tener en cuenta que el tamaño de la ventana de transmisión esta dado por los valores comprendidos entre NextXmTime 1 y NextXmTime2.

Es posible mejorar este panorama asignando valores diferentes de NextXmMx para que se desplace esta ventana de transmisión cierto número de veces. En el mejor de los casos, y tal como lo podemos ver en la figura 4.15 conviene seleccionar lo mas diferente posible los valores de NextXmMx.

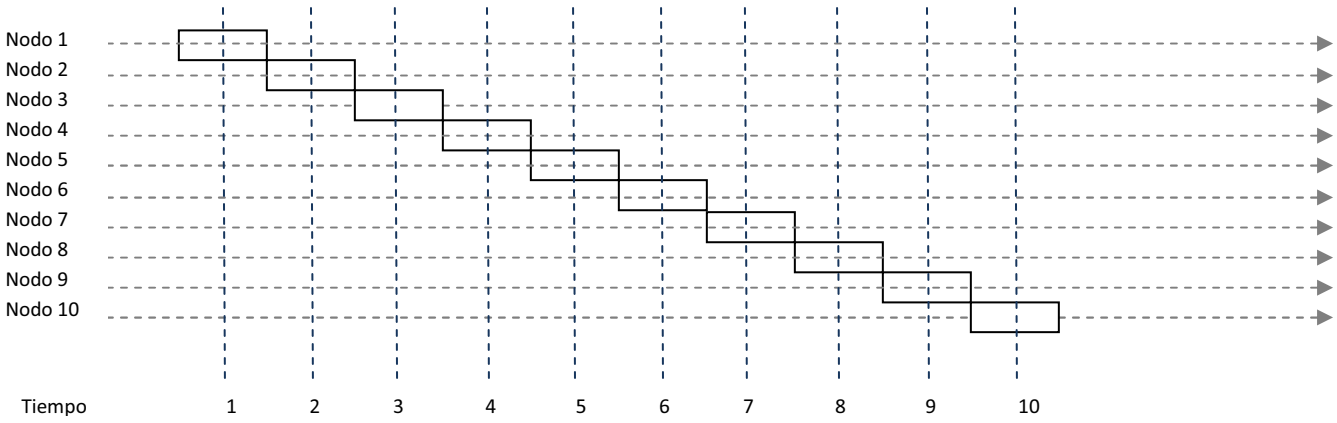


Figura 4.15. Ilustración del comportamiento de los nodos con diferentes NextXmMx.

En este caso, podemos ver que se aprovecha de una mejor forma los tiempos de transmisión ya que existen menos competencias entre los nodos, de hecho, en este caso, no existe ninguna competencia y no es necesario llamar al algoritmo de selección. Con esto podemos observar que ya no se desperdician los 16 tiempos anteriores ya que resulta un nodo ganador para cada nodo.

Cuando corremos el programa que se encuentra en el anexo 1 obtenemos los siguientes resultados:

```
Nodo: 1
 XmtHoldoffTime = 16
 NextXmtMx = 0
 NextXmtTime1 = 1
 NextXmtTime2 = 1
 EarliestSubsequentXmtTime = 17

Nodo: 2
 XmtHoldoffTime = 16
 NextXmtMx = 1
 NextXmtTime1 = 2
 NextXmtTime2 = 2
 EarliestSubsequentXmtTime = 18

Nodo: 3
 XmtHoldoffTime = 16
 NextXmtMx = 2
 NextXmtTime1 = 3
```

```
Nodo: 6
 XmtHoldoffTime = 16
 NextXmtMx = 5
 NextXmtTime1 = 6
 NextXmtTime2 = 6
 EarliestSubsequentXmtTime = 22

Nodo: 7
 XmtHoldoffTime = 16
 NextXmtMx = 6
 NextXmtTime1 = 7
 NextXmtTime2 = 7
 EarliestSubsequentXmtTime = 23

Nodo: 8
 XmtHoldoffTime = 16
 NextXmtMx = 7
 NextXmtTime1 = 8
```

```
 NextXmtTime2 = 3
 EarliestSubsequentXmtTime = 19

Nodo: 4
 XmtHoldoffTime = 16
 NextXmtMx = 3
 NextXmtTime1 = 4
 NextXmtTime2 = 4
 EarliestSubsequentXmtTime = 20

Nodo: 5
 XmtHoldoffTime = 16
 NextXmtMx = 4
 NextXmtTime1 = 5
 NextXmtTime2 = 5
 EarliestSubsequentXmtTime = 21
```

```
 NextXmtTime2 = 8
 EarliestSubsequentXmtTime = 24

Nodo: 9
 XmtHoldoffTime = 16
 NextXmtMx = 8
 NextXmtTime1 = 9
 NextXmtTime2 = 9
 EarliestSubsequentXmtTime = 25

Nodo: 10
 XmtHoldoffTime = 16
 NextXmtMx = 9
 NextXmtTime1 = 10
 NextXmtTime2 = 10
 EarliestSubsequentXmtTime = 26
```

Lo cual nos brinda los valores de las ventanas de transmisión que se mostraron en la figura 4.15.

En este caso también hay que considerar el valor de Holdoff Exponent ya que dependiendo el valor es el tamaño de la ventana que se obtiene pero se presenta el mismo comportamiento “escalonado” que se analizó anteriormente. Gráficamente lo podemos ver en la figura 4.16. Es necesario notar que no nos conviene tener un número pequeño de nodos con un Holdoff Exponent grande porque se agrandarían los tiempos de respuesta.

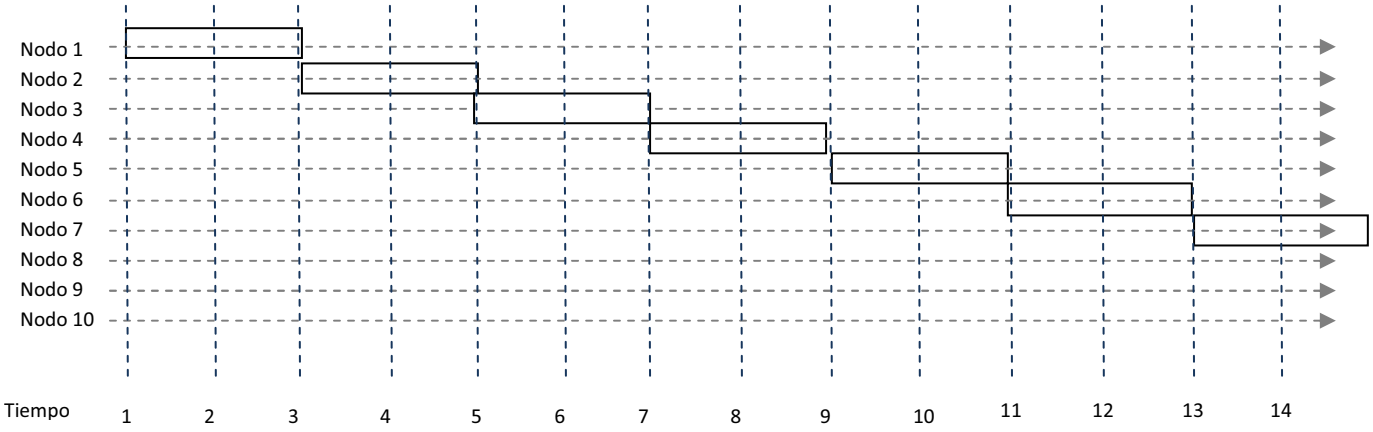


Figura 4.16. Ilustración del comportamiento de los nodos con diferentes Holdoff Exponent

En este caso se puede observar de una manera más clara los distintos papeles que juegan las variables. La diferencia con el caso anterior es que el Holdoff Exponent era de cero y en este caso es de uno, por eso el tamaño de la ventana se incrementó.

En el siguiente capítulo se hablará de la optimización de todas las variables para obtener tiempos de registro más pequeños y tomando en cuenta todas las variables descritas anteriormente.

4.6 Resultados

4.6.1. Nodos ganadores respecto al tiempo

Después de considerar todos los parámetros que se indicaron anteriormente, podemos ver finalmente los resultados que nos arroja el código del anexo 1 una vez depurando las salidas.

Considerando uno de los mejores escenarios de simulación que consiste en Holdoff Exponent de cero, el NextXmMx lo más diferente posible (asignando el valor 0 al nodo 1, el valor 1 al nodo 2 y así sucesivamente, hasta el valor de 31 para el nodo 32 y se repite la secuencia para el nodo 33) y considerando que sólo hay 10

nodos en la red, el resultado en el que transmiten los mensajes NCFG es el mostrado en la tabla 4.1 para los primeros 103 tiempos.

Aquí podemos observar que debido a la asignación del NextXmMx aparentemente no hay aleatoriedad al principio, posteriormente y debido a las ecuaciones comentadas empiezan a existir competencias entre los nodos generando resultados más aleatorios. Hay que tomar en cuenta que existen una infinidad de resultados que se pueden generar, la intención de esto es mostrar los resultados del anexo 1.

Nodo ganador	Oportunidad de transmisión	Tiempo [seg]
1	1	6.8359E-05
2	2	0.00013672
3	3	0.00020508
4	4	0.00027344
5	5	0.0003418
6	6	0.00041015
7	7	0.00047851
8	8	0.10006836
9	9	0.10013672
10	10	0.10020508
1	18	0.20027344
2	20	0.20041015
3	22	0.20054687
4	24	0.30020508
5	26	0.3003418
6	28	0.30047851
7	30	0.30061523
8	32	0.40027344
9	34	0.40041015
1	35	0.40047851
10	36	0.40054687
2	38	0.40068359
3	41	0.50041015
4	44	0.50061523
5	47	0.50082031
6	50	0.60054687
1	52	0.60068359
7	53	0.60075195
8	56	0.70047851
9	59	0.70068359
3	60	0.70075195
10	62	0.70088867
4	64	0.80054687
5	68	0.80082031
1	69	0.80088867
6	72	0.90061523
2	74	0.90075195
7	76	0.90088867
3	79	0.90109374
8	80	1.00068359
4	84	1.00095703
1	86	1.00109374
10	88	1.10075195
5	89	1.10082031
2	92	1.10102539
6	94	1.1011621
3	98	1.20095703
7	99	1.20102539
1	103	1.20129882

Tabla 4.1. Nodos Ganadores respecto al Tiempo

4.6.2 Número de nodos en la red

El comportamiento de los nodos en la red propuesta en nuestra tesis depende de varias variables como ya se vió en los capítulos anteriores, pero para tener la mejor distribución de tiempos fue necesario realizar una serie

de gráficas con el algoritmo que implementamos para llegar a la conclusión de asignarle los mejores valores a las variables que afectan directamente al resultados del registro de los nodos a la red.

Así que a la variable NextXmtMx se decidió asignarle valores corridos hasta el valor de 31 porque esta variable es de 5 bits de acuerdo al estándar, lo de asignarle valores corridos se refiere a que se le asignaron valores de 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, … , 31 y se tiene un número de nodos mayor a 31 se vuelve a repetir la numeración, es decir empezar desde 0 hasta 31.

Lo anterior se decidió implementar de esta forma porque al tener los valores de NextXmtMx distribuidos los nodos no tienen que competir varias veces y así no se gasta tiempo en lo que el algoritmo decide al ganador. Esto se debe a que la fórmula del Next Xmt Time depende del valor de NextXmtMx y como esta variable da el intervalo en el cual los nodos van a transmitir y si el valor del NextXmtMx es el mismo para varias variables, éstas caen en el mismo intervalo de transmisión por lo que tendrán que competir y al hacer esto se pierde bastante tiempo. Es por esta razón que se decidió utilizar el valor de NextXmtMx lo mas distribuido posible con el fin de evitar lo anterior.

En las siguientes gráficas se tienen una serie de datos que se obtuvieron al aplicar el algoritmo de selección con diferentes HoldoffExponent y con diferentes valores de NextXmtMx.

Los siguientes valores servirán para ver mejor el comportamiento de los nodos en la red y el tiempo que tardan en transmitir por lo menos un mensaje NCFG.

- **Tabulación de Nodos vs Tiempo NCFG y con 7 NCFG en un Frame y con NextXmTime diferente**

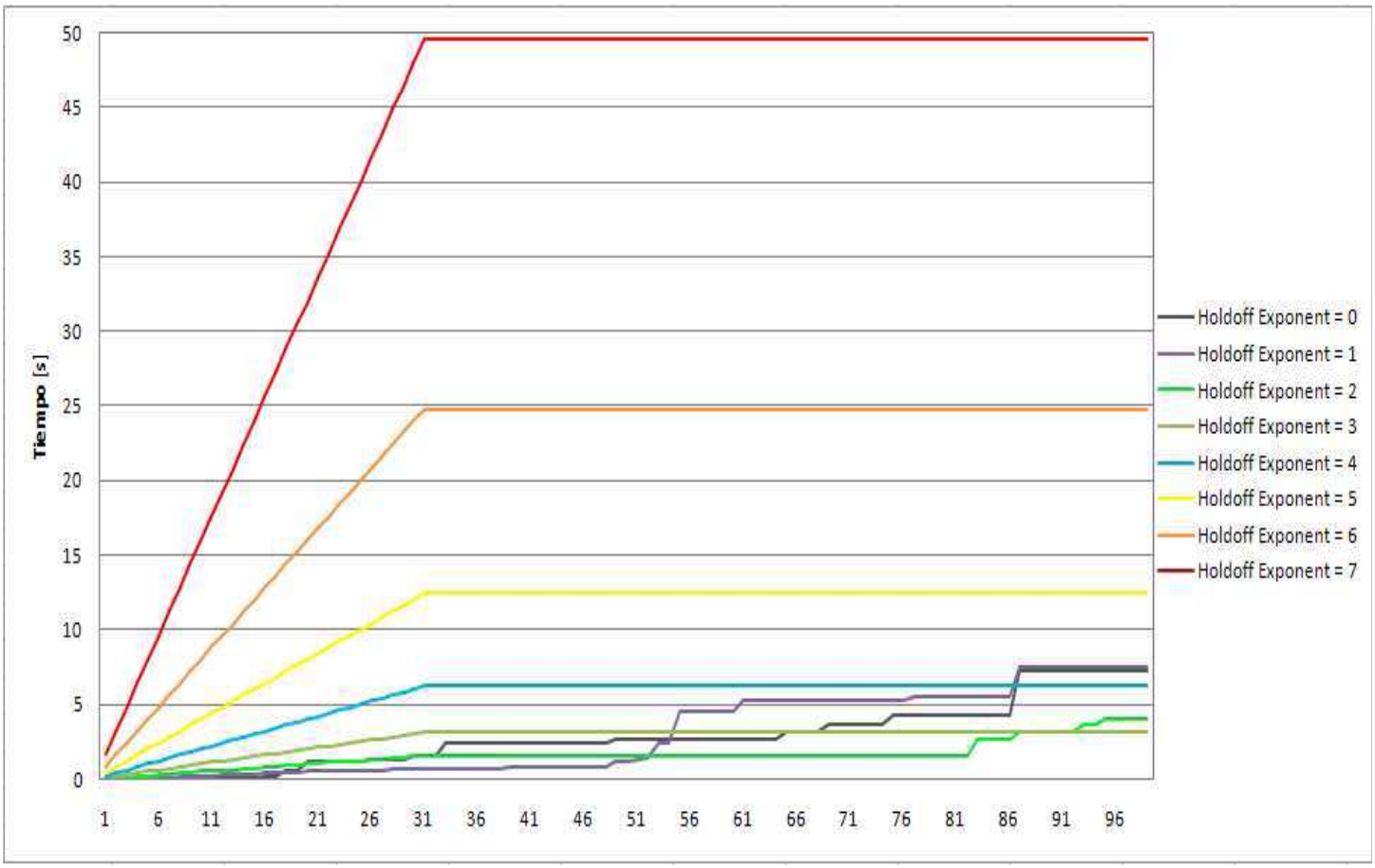


Figura 4.17. Gráfica de Nodos vs Tiempo NCFG (Transmiten una vez) y con 7 NCFG en un Frame y con NextXmTime diferentes (hasta 32)

- **Tabulación de Nodos vs Tiempo NCFG y con 11 NCFG en un Frame y con NextXmTime diferentes**

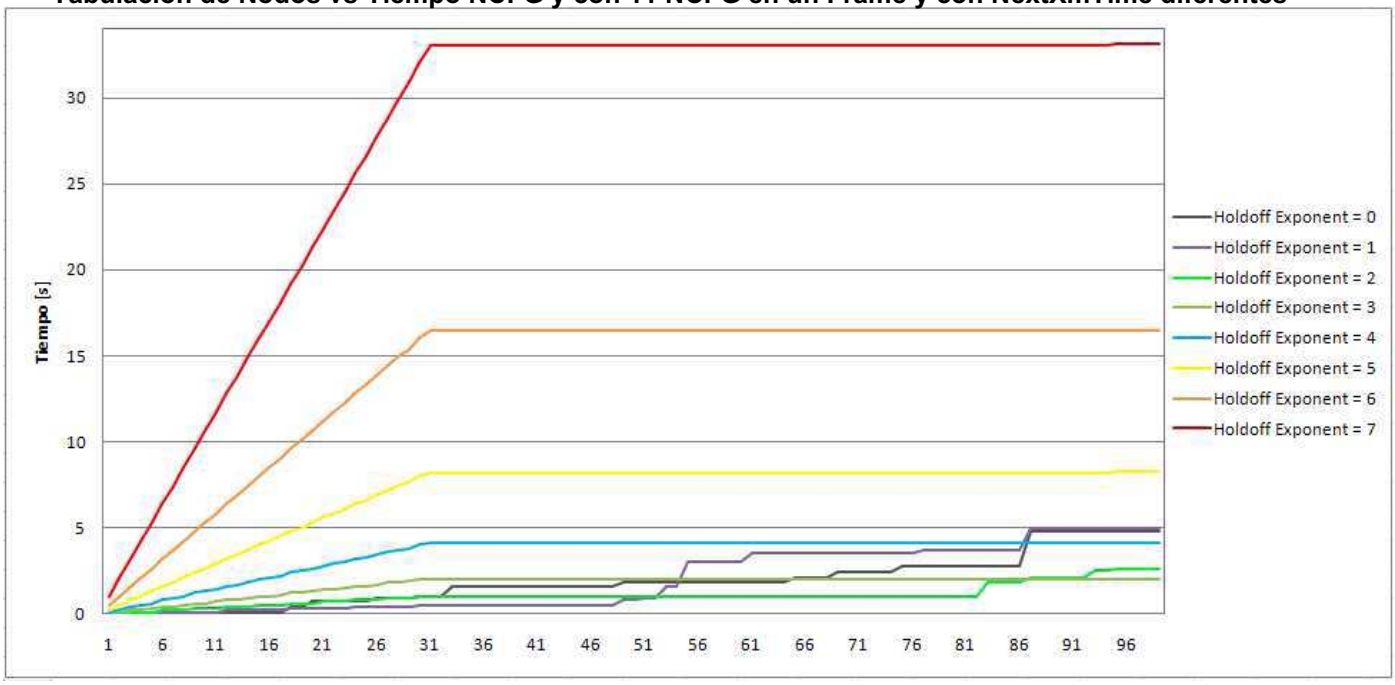


Figura 4.18. Gráfica de Nodos vs Tiempo NCFG (Transmiten una vez) y con 11 NCFG en un Frame y con NextXmTime diferentes (hasta 32).

- **Tabulación de Nodos vs Tiempo NCFG y con 15 NCFG en un Frame y con NextXmTime diferentes**

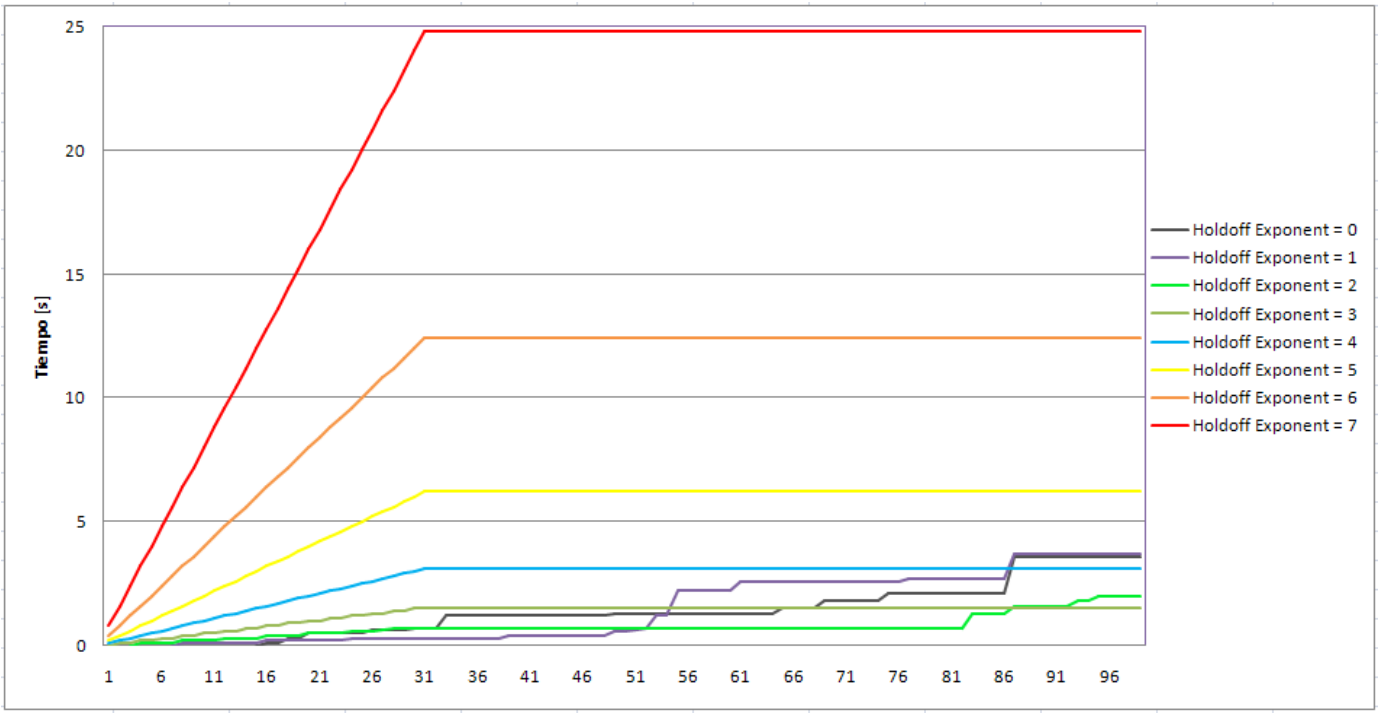


Figura 4.19. Gráfica de Nodos vs Tiempo NCFG (Transmiten una vez) y con 15 NCFG en un Frame y con NextXmTime diferentes (hasta 32).

De las gráficas anteriores se observa claramente que el tiempo de registro varia bastante, esto se debe a que para las tres gráficas se tienen 3 estructuras diferentes del frame de transmisión, la primer gráfica tiene un frame con 7 oportunidades de transmisión de NCFG, para la segunda gráfica se tienen 11 oportunidades de transmisión y la última tiene 15 oportunidades de transmisión en el frame.

Se concluye que al tener un frame con más oportunidades de transmisión el tiempo de registro se reduce bastante, además en las tres gráficas se observa que los mejores tiempos se obtienen al tener un HoldoffExponent = 3. Sin embargo para esta tesis se utilizo un frame con 7 oportunidades de transmisión para los mensajes NCFG y 1 oportunidad de transmisión para mensajes NENT.

4.6.3. Número de transmisiones por nodo

Como se analizó en el punto 4.6.1, cuando se utiliza el algoritmo de selección del nodo ganador para transmitir los mensajes de control, se realiza un proceso pseudoaleatorio donde cada nodo puede saber cuándo debe de transmitir. Aunque se puede observar en la tabla que también la cantidad de veces que transmite es diferente.

Para hacer un análisis más detallado se hizo una modificación al código en C++ utilizado en el punto 4.6.2 para poder contabilizar el número de transmisiones que cada nodo realizó. Se añadieron un par de líneas que realizan la función de sumar una transmisión cada vez que el nodo gana y transmite. El código completamente funcional que se utilizó se muestra en el anexo 2 y junto con el primer ciclo de transmisiones del mensaje MSH – NCFG. Los resultados, al igual, que en puntos anteriores, depende de una gran cantidad de factores, en este caso se tomaron como variables la ID del nodo, el número de nodos y el Holdoff Exponent. Los valores de Next Xm Mx se asignaron de la mejor forma, esto es lo más distribuido posible (asignando el valor 0 al nodo 1, el valor 1 al nodo 2 y así sucesivamente, hasta el valor de 31 para el nodo 32 y se repite la secuencia para el nodo 33). Por ejemplo, se realizó una simulación en donde se cuenta con 100 nodos, estos nodos tienen distintos HoldoffExponent y además tienen los valores de NextXmMx lo más variable posible, tal como se ha comentado en los puntos anteriores con el fin de evitar el mayor número de competencias y aprovechar al máximo los tiempos de transmisión. Los resultados para 100 nodos los podemos ver en la figura 4.20.

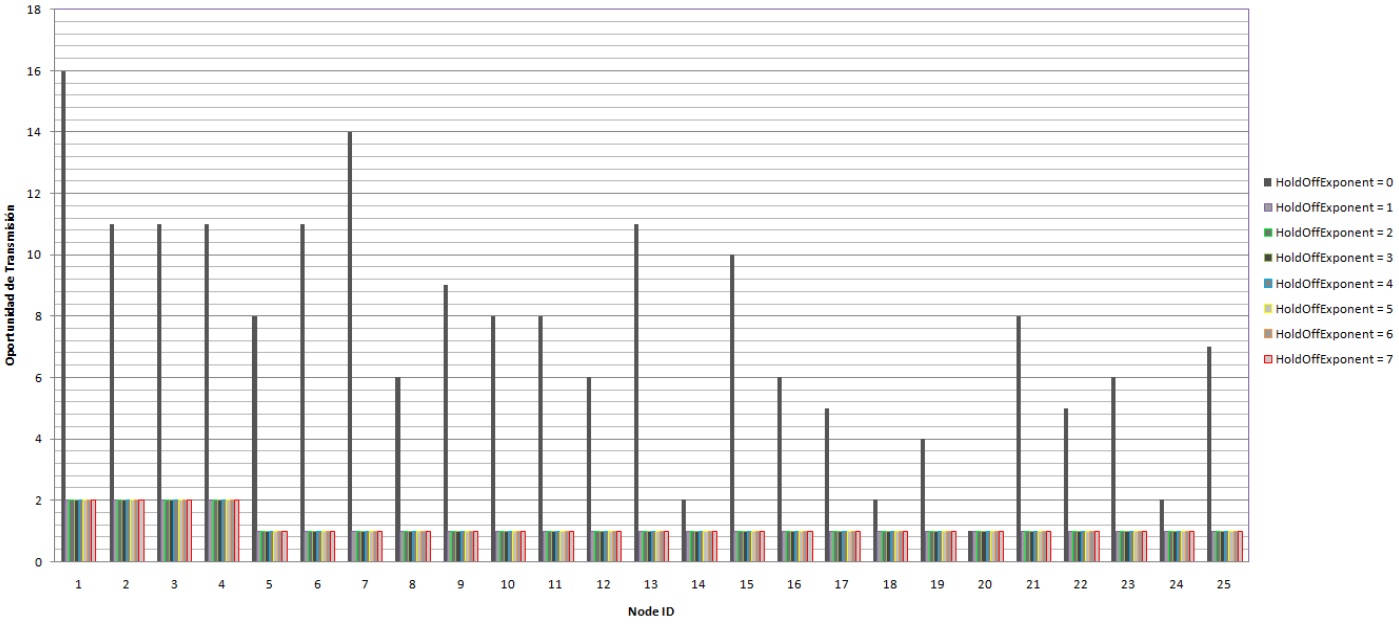


Figura 4.20.Gráfica de número de transmisiones vs ID de nodos (25 nodos) en un ciclo de transmisión (todos transmiten 1 vez por lo menos).

En este caso podemos observar que para el caso de un Holdoff Exponent de cero es bastante diferente el patrón que sigue el número de veces que transmite un nodo. Posteriormente, para los demás Holdoff Exponent transmiten de una forma muy pareja. Esto es debido a que la forma en que se asignó el NextXmMx (lo más diferente posible para evitar competencias) y el tamaño de la ventana (Holdoff Exponent) permiten que casi no haya competencias y cada nodo gane en su tiempo asignado.

Analicemos el mismo caso con los mismo parámetros pero aumentando la cantidad de nodos. Esto lo podemos ver en la figura 4.21. En este caso, al aumentar la cantidad de nodos aumentan las competencias ya que hay que recordar que sólo hay 32 espacios de diferencia para competir, debido a los 5 bits del NextXmMx. Si tenemos 50 nodos aumentará el número de competencias y algunos nodos ganarán más que otros. Al aumentar el tamaño de la ventana de transmisión (Holdoff Exponent) disminuye el número de competencias pero aumenta el tiempo en que transmite un mensaje. Por consecuencia se estabiliza y transmite una cantidad similar de veces.

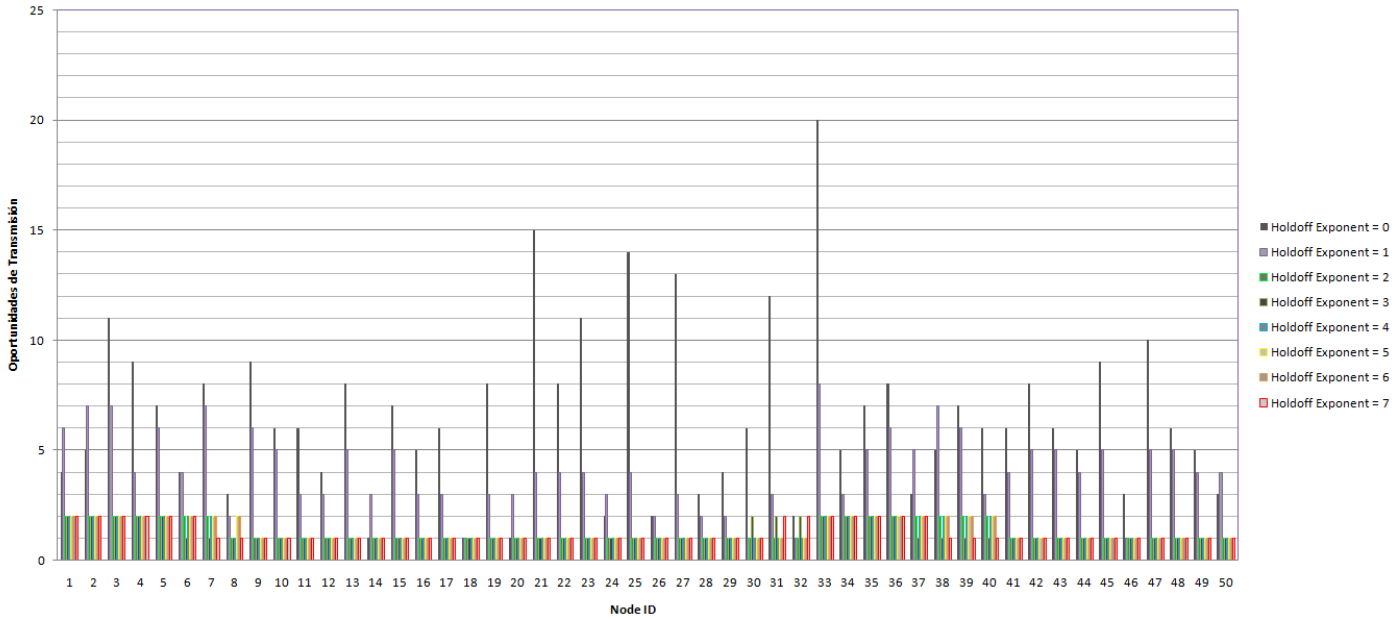


Figura 4.21. Gráfica de número de transmisiones vs ID de nodos (50 nodos) en un ciclo de transmisión (todos transmiten 1 vez por lo menos).

La intención del anexo 2 es poder manipular todas las variables con las cuales se generaron estas gráficas. Por ejemplo, se analizaron los casos óptimos para transmitir, aunque hay que recordar que existen muchas combinaciones posibles de las distintas variables, de tal forma, que es imposible analizar todas.

Capítulo 5

Modelo para Registro de Nodos en la red Mesh

5.1 Numero de mensajes NCFG y NENT.

Para el proceso de registro de los nodos en la red es necesario intercambiar los distintos mensajes de control tal como se explicó en el punto 3.6. Los mensajes que un nodo debe de intercambiar son MSH - NCFG: Network Descriptor IE, MSH - NENT: NetEntryRequestIE, MSH - NCFG: EntryOpenIE, MSH - NENT: NetEntryAckIE, MSH - NENT: NetEntryCloseIE, MSH - NCFG: NetEntryAckIE en ese orden respectivamente.

En este caso es necesario intercambiar con la estación base 6 mensajes de control. Pero si consideramos una configuración tipo Mesh entonces es necesario considerar que el nodo hijo tiene que intercambiar estos mensajes con el nodo padre, éste a su vez tiene que intercambiar mensajes de control con la estación base.

Se genera el esquema de la figura 5.1.

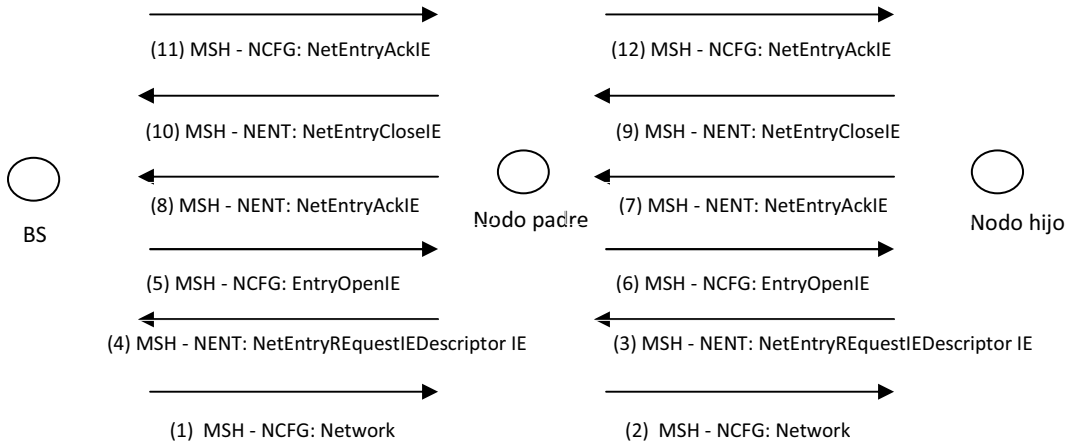


Figura 5.1. Número de mensajes de control necesarios para dar un nodo de alta en configuración Mesh.

Tal como se puede observar, es necesario incrementar el número de mensajes de control involucrados para que un nido hijo se pueda dar de alta. En este caso y tomando en cuenta que hay dos niveles jerárquicos (el primer nivel es el nodo padre y el segundo nivel es el nodo hijo) se requieren intercambiar 12 mensajes de control. Si hacemos una analogía con el caso anterior, si se tienen tres niveles jerárquicos entonces se requerirán 18 mensajes de control para dar de alta un nodo en la red Mesh.

5.2 Funciones adicionales

5.2.1 Función para mensajes MSH – NENT

Los mensajes MSH – NENT que se proponen en el estándar, están contemplados en transmitirse en una oportunidad de transmisión antes de los mensajes MSH – NCFG tal como se analizó anteriormente en la estructura de los frames. Pero hay que tener en cuenta que una oportunidad de transmisión nos representa en información 504 bytes (ver tema 4.3.7). El tamaño de los mensajes que se utiliza varia, dependiendo de los 3 mensajes que se necesitan transmitir:

- MESH – NENT REQUEST: Si analizamos la estructura del mensaje podemos observar que máximo se obtienen 27 bytes para transmitir. Hay que tener en cuenta que todos los mensajes MESH - NENT utilizan un mismo encabezado de 5 bytes. Este mensaje es el único además de este encabezado utiliza más información.

- MESH – NENT ACK: De igual forma, podemos analizar la estructura de este mensaje y podemos observar que utiliza 5 bytes.

- MESH – NENT CLOSE: Utiliza la misma estructura del mensaje anterior por lo que su tamaño es el mismo, 5 bytes.

Teniendo esta información en cuenta podemos ver que se estarían desaprovechando una gran parte de estos mensajes de control ya que el tamaño de la oportunidad de transmisión es de 504 bytes. Debido a esto, se hace la propuesta de dividir estos mensajes de control en 7 oportunidades de transmisión de tal forma que se pueda optimizar y reducir los tiempos de registro de los nodos en la red.

Gráficamente, podemos ver la estructura de los frames antes y después de realizar la división de los mensajes NENT.

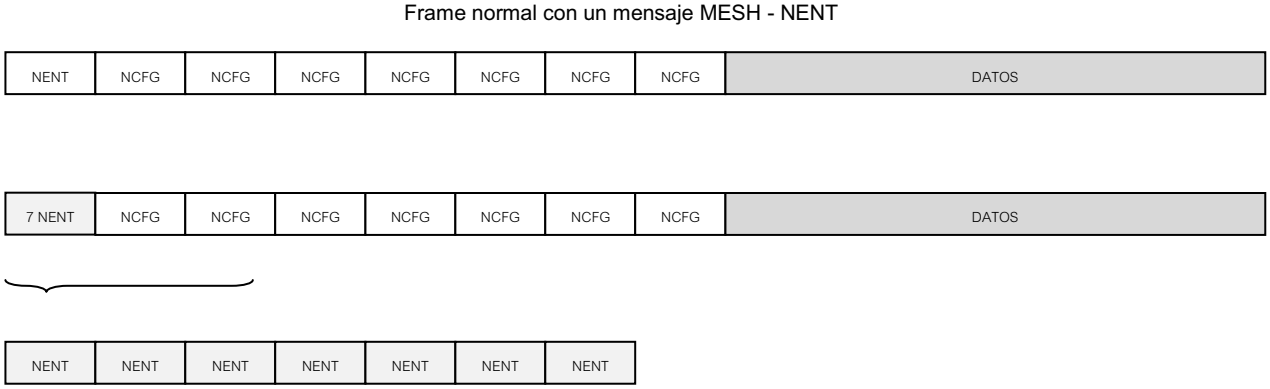


Figura 5.2. Frame con mensaje de control MSH – NENT en 7 partes.

El beneficio directo de hacer esta división es que los tiempos en que se registran los nodos se ven disminuidos.

La forma en que funciona el algoritmo es establecer una función aleatoria. Debido a que esta tesis se enfoca al comportamiento de los mensaje de control MSH – NCFG se hace sólo una propuesta de cómo transmitir los mensajes MSH – NENT.

El código de la función se puede apreciar en el anexo 3. La función opera realizando una lista de competidores, en este caso, en NodeID de cada estación suscriptora. Cuando uno de los competidores gana, sale del proceso de competencia hasta que todos los nodos transmitan una vez. La selección del ganador se basa en la función rand () que nos proporciona c++ y en una semilla para generar siempre el mismo número pseudoaleatorio cada vez que se entra en los ciclos que se tomaron en cuenta para su programación. De esta forma se logra que además de aleatoria sea justo el número de veces que un nodo transmite.

5.2.2 Función Para Convertir Tiempo de Simulación a Oportunidades de Transmisión

Para implementar el código en C++ fue necesario implementar una función la cual nos brindara la característica de convertir el tiempo real de la simulación hacia las oportunidades de transmisión

TO= 7 * x + (realtime + TIME_7_SYMB * x - x * 0.1)/ 0.000068359;

Esta ecuación surge del mismo análisis realizado en el comportamiento del algoritmo y nos es bastante útil en la implementación de la simulación únicamente para los mensajes MSH – NCFG.

Para los mensajes MSH – NENT es necesario tomar otras ecuaciones, para ello fue necesario hacer otro análisis. Dado a que los mensajes MSH – NENT se consideró que pueden dividirse en 7 oportunidades de transmisión. Tomando en cuenta el intervalo de tiempo que ocupan los mensajes MSH – NENT se implementó el siguiente pseudocódigo:

```
if ( realtime < (x * 0.1 + 0.000009766 * 7))
        TO = (x + 1) * (NCFG_FRAME+ 1) - NCFG_FRAME + 5;
if ( realtime < (x * 0.1 + 0.000009766 * 6))
   TO = (x + 1) * (NCFG_FRAME+ 1) - NCFG_FRAME + 4;
if ( realtime < (x * 0.1 + 0.000009766 * 5))
   TO = (x + 1) * (NCFG_FRAME+ 1) - NCFG_FRAME + 3;
if ( realtime < (x * 0.1 + 0.000009766 * 4))
   TO = (x + 1) * (NCFG_FRAME+ 1) - NCFG_FRAME + 2;
if ( realtime < (x * 0.1 + 0.000009766 * 3))
   TO = (x + 1) * (NCFG_FRAME+ 1) - NCFG_FRAME + 1;
if ( realtime < (x * 0.1 + 0.000009766 * 2))
   TO = (x + 1) * (NCFG_FRAME+ 1) - NCFG_FRAME + 0;
if ( realtime < (x * 0.1 + 0.000009766 * 1))
   TO = (x + 1) * (NCFG_FRAME+ 1) - NCFG_FRAME - 1;
```

Donde x es la parte entera del tiempo de simulación multiplicada por 10. La utilidad de este código puede verse reflejada con más detalle en el código de C++ del anexo 3.

Además, debido a que se hicieron funciones diferentes fue necesario hacer una función para convertir de las oportunidades de transmisión NCFG a oportunidades de transmisión NENT. Básicamente sucede ya que se consideró que por cada una oportunidad de transmisión de NENT habrá 7 oportunidades de transmisión de NCFG. Estos detalles son necesarios de ver en el código del anexo 3.

5.3 Implementación del modelo de simulación para registro de los nodos.

En el capítulo 4 se analizaron las bases del algoritmo para dar de alta nodos en la red Mesh. Se pudieron observar la forma en que se comportan los distintos parámetros y cómo afectan el tiempo en que se envían los mensajes. Para realizar la configuración Mesh tenemos que partir de un escenario de simulación, en este caso es el que se muestra en la figura 5.3.

Aquí se puede observar que la estación base tiene diez estaciones suscriptoras de alcance. El modelo descrito en el anexo 3 nos brinda los tiempos en que cada nodo intercambia los seis mensajes de control necesarios para dar de alta un nodo en la red con la estación suscriptora. Aquí hay que tomar en cuenta todas las configuraciones posibles y las variables que pueden surgir. Recordemos que las variables son el Holdoff Exponent, NextXmMx así como el número de nodos, que en este caso se mantiene fijo. Una de las grandes ventajas del código de programación en C++ es que estas variables se han definido en el encabezado del

código, brindando mayor facilidad para generar múltiples escenarios, sólo basta con reemplazar por el valor correcto para obtener el resultado deseado, la única consideración necesaria es que hay que introducir los valores que el estándar nos define ya que de lo contrario se obtendrían datos que no corresponden a la realidad.

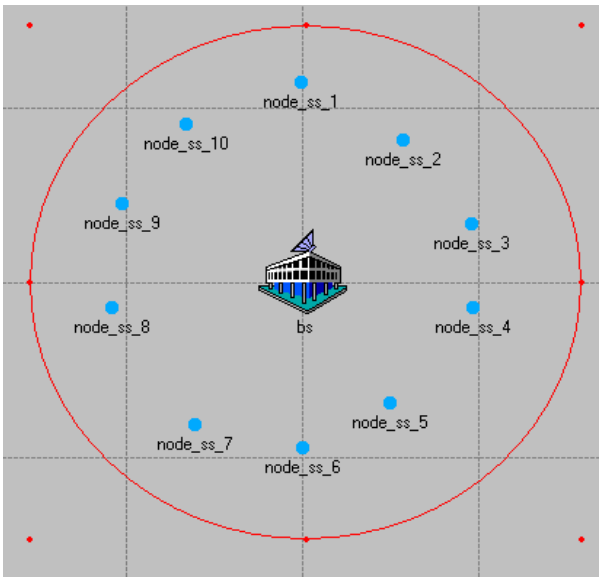


Figura 5.3. Primer nivel jerárquico de la configuración Mesh

Una vez que los diez nodos se han dado de alta es tiempo de que sus nodos hijos se empiecen a dar de alta. Para esto, se guarda el tiempo en que cada nodo se terminó de dar de alta para que sus hijos se empiecen a dar de alta de igual forma que los nodos padres, es decir, intercambiando los seis mensajes de control necesarios para darse de alta, sólo hay que tener en cuenta que los nodos padres tendrán que enviar los mensajes de control a la estación base, por lo tanto se generarán doce mensajes de control que se necesitan intercambiar.

Dentro de este proceso es necesario hacer notar que, además de que el número de nodos se incrementa y sólo uno puede transmitir a la vez, es necesario esperar a que el nodo padre gane su oportunidad de transmisión, por lo que los tiempos se incrementan considerablemente. En otras palabras, se incrementa el número de nodos y se necesitan dos oportunidades de transmisión (para el nodo padre y el hijo) para que un nodo pueda enviar un mensaje de control, si tenemos en cuenta que se tienen que enviar seis mensajes de control se necesitarán doce oportunidades de transmisión. Este análisis se hará más adelante para determinar el tiempo que se eleva el registro de los nodos en un segundo nivel jerárquico.

La figura 5.4 nos indica el esquema del modelo en simulación hasta esta etapa. Hay que tener en cuenta que el diagrama es ilustrativo ya que los nodos en el primer nivel tienen el mismo alcance que la radiobase.

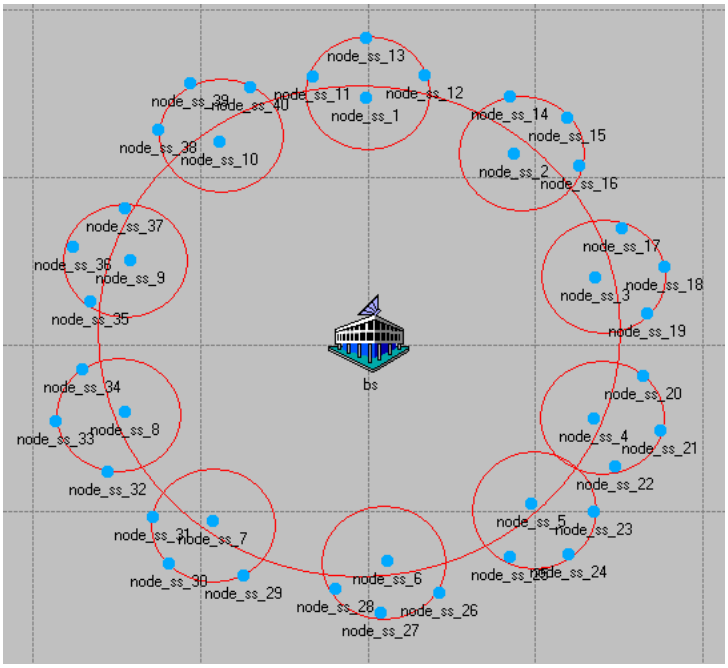


Figura 5.4. Primer y segundo nivel jerárquico de la configuración Mesh

Posteriormente se toma en cuenta el tiempo en que cada nodo se registra para empezar a dar de alta a sus nodos hijos (el tercer nivel jerárquico). En este escenario de simulación es posible cambiar el número de nodos en el primer, segundo y tercer nivel jerárquico, solo hay que tener cuidado en establecer las constantes en el código C++ ya que pueden presentarse resultados inesperados. Finalmente, los nodos del tercer nivel jerárquico tienen que transmitir 18 mensajes de control para poderse dar de alta. Los tiempos como es de esperarse se incrementan ya que se requieren de tres oportunidades de transmisión para que se pueda enviar un mensaje de control de los seis necesarios.

En la figura 5.5 podemos apreciar el esquema de la red Mesh con los tres niveles jerárquicos mencionados.

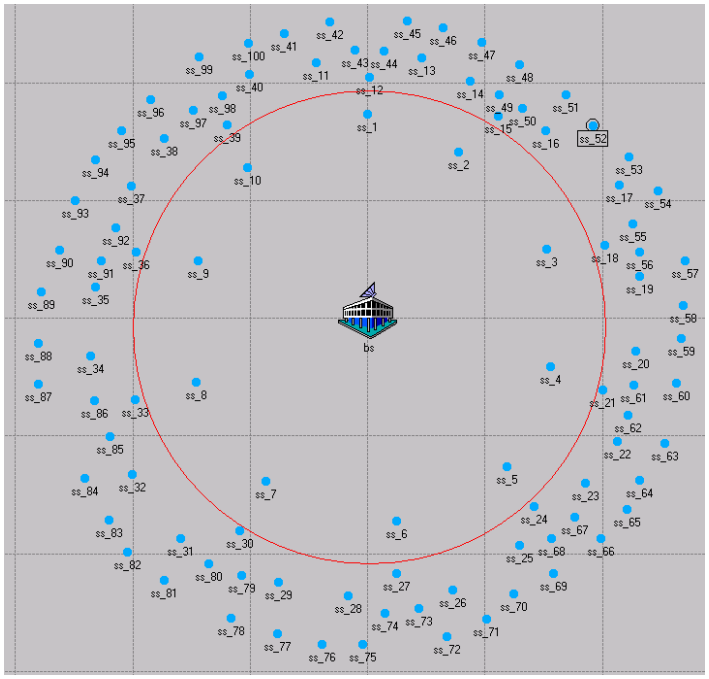


Figura 5.5. Primer, segundo y tercer nivel jerárquico de la configuración Mesh

5.4 Resultados del Modelo para registro de nodos en la red

Tomando en cuenta los tres niveles jerárquicos mencionados en el punto anterior e implementando el código en C++ descrito en el anexo tres podemos obtener los resultados mostrados en la tabla 5.1 utilizando un Holdoff Exponent de cero para el primer nivel, considerando los 10 primeros nodos de la red y con el parámetro Next Xm Time lo más variable posible, tal como se ha manejado. Además es importante notar que se usaron 7 oportunidades de transmisión para los mensajes MSH – NCFG y además se dividió el espacio del NENT en 7 oportunidades tal como se habló en el punto anterior.

	NODO 1	
MESH - NCFG (NetDes)	----------	0.0000000000 [s]
MESH - NENT (Request)	----------	0.2000000000 [s]
MESH - NCFG (OpenIE)	----------	0.2003417950 [s]
MESH - NENT (Ack)	----------	0.4000195320 [s]
MESH - NENT (CloseID)	----------	0.6000195320 [s]
MESH - NCFG (Ack)	----------	0.6904785130 [s]
	NODO 2	
MESH - NCFG (NetDes)	----------	0.0000000000 [s]
MESH - NENT (Request)	----------	0.1000195320 [s]
MESH - NCFG (OpenIE)	----------	0.2904785130 [s]
MESH - NENT (Ack)	----------	0.5000488300 [s]
MESH - NENT (CloseID)	----------	0.7000195320 [s]
MESH - NCFG (Ack)	----------	0.8001367180 [s]
	NODO 3	
MESH - NCFG (NetDes)	----------	0.0000000000 [s]
MESH - NENT (Request)	----------	0.2000000000 [s]
MESH - NCFG (OpenIE)	----------	0.3001367180 [s]
MESH - NENT (Ack)	----------	0.5000292980 [s]
MESH - NENT (CloseID)	----------	0.8000488300 [s]
MESH - NCFG (Ack)	----------	0.9002734360 [s]
	NODO 4	
MESH - NCFG (NetDes)	----------	0.0000000000 [s]
MESH - NENT (Request)	----------	0.3000292980 [s]
MESH - NCFG (OpenIE)	----------	0.3904785130 [s]
MESH - NENT (Ack)	----------	0.5000195320 [s]
MESH - NENT (CloseID)	----------	1.0000097660 [s]
MESH - NCFG (Ack)	----------	1.1904785130 [s]
	NODO 5	
MESH - NCFG (NetDes)	----------	0.0000000000 [s]
MESH - NENT (Request)	----------	0.1000000000 [s]
MESH - NCFG (OpenIE)	----------	0.3004101540 [s]
MESH - NENT (Ack)	----------	0.5000390640 [s]
MESH - NENT (CloseID)	----------	1.1000390640 [s]
MESH - NCFG (Ack)	----------	1.3001367180 [s]
	NODO 6	
MESH - NCFG (NetDes)	----------	0.0000000000 [s]
MESH - NENT (Request)	----------	0.1000488300 [s]
MESH - NCFG (OpenIE)	----------	0.4000683590 [s]
MESH - NENT (Ack)	----------	0.7000488300 [s]
MESH - NENT (CloseID)	----------	0.9000195320 [s]
MESH - NCFG (Ack)	----------	1.1000683590 [s]
	NODO 7	
MESH - NCFG (NetDes)	----------	0.0000000000 [s]
MESH - NENT (Request)	----------	0.5000000000 [s]
MESH - NCFG (OpenIE)	----------	0.6002734360 [s]
MESH - NENT (Ack)	----------	0.8000195320 [s]
MESH - NENT (CloseID)	----------	1.0000195320 [s]
MESH - NCFG (Ack)	----------	1.2002050770 [s]
	NODO 8	
MESH - NCFG (NetDes)	----------	0.0000000000 [s]
MESH - NENT (Request)	----------	0.2000097660 [s]
MESH - NCFG (OpenIE)	----------	0.4003417950 [s]

```
MESH - NENT (Ack)        ----------------   0.9000097660 [s]
MESH - NENT (CloseID)    ----------------   1.1000195320 [s]
MESH - NCFG (Ack)        ----------------   1.3003417950 [s]

                              NODO 9
MESH - NCFG (NetDes)     ----------------   0.0000000000 [s]
MESH - NENT (Request)    ----------------   0.2000390640 [s]
MESH - NCFG (OpenIE)     ----------------   0.4904785130 [s]
MESH - NENT (Ack)        ----------------   1.0000390640 [s]
MESH - NENT (CloseID)    ----------------   1.2000195320 [s]
MESH - NCFG (Ack)        ----------------   1.4904785130 [s]

                             NODO 10
MESH - NCFG (NetDes)     ----------------   0.0000000000 [s]
MESH - NENT (Request)    ----------------   0.3000195320 [s]
MESH - NCFG (OpenIE)     ----------------   0.5001367180 [s]
MESH - NENT (Ack)        ----------------   1.3000000000 [s]
MESH - NENT (CloseID)    ----------------   1.5000195320 [s]
MESH - NCFG (Ack)        ----------------   1.8002734360 [s]
```

Tabla 5.1. Resultados detallados del modelo de registro de nodos para el primer nivel jerárquico.

Posteriormente vienen los tiempos de registro para cada mensaje de control que se tiene que intercambiar, los resultados del segundo nivel los podemos ver en la tabla 5.2. Sólo se ilustra el proceso tal como lo arroja el código para los primeros tres nodos, a fin de comparar los tiempos en que empiezan a registrarse y la forma en que el nodo padre también interviene en el registro. Para los demás nodos sólo se incluye el tiempo final de registro. Hay que destacar que el nodo uno del hijo uno se le denotó 1,1 y así sucesivamente.

```
                                   NODO 1,1 (11)
MESH - NCFG (NetDes)   TO NODE 1 FOR NODE 1,1   ----------------   21.0002734360 [s]
MESH - NCFG (NetDes)   NODE 1,1                 ----------------   24.1904785130 [s]
MESH - NENT (Request)  NODE 1,1                 ----------------   24.4000488300 [s]
MESH - NENT (Request)  TO NODE 1 FOR NODE 1,1   ----------------   24.7000292980 [s]
MESH - NCFG (OpenIE)   TO NODE 1 FOR NODE 1,1   ----------------   28.7001367180 [s]
MESH - NCFG (OpenIE)   NODE 1,1                 ----------------   32.9004101540 [s]
MESH - NENT (Ack)      NODE 1,1                 ----------------   33.2000000000 [s]
MESH - NENT (Ack)      TO NODE 1 FOR NODE 1,1   ----------------   33.4000195320 [s]
MESH - NENT (CloseID)  TO NODE 1 FOR NODE 1,1   ----------------   34.0000390640 [s]
MESH - NENT (CloseID)  NODE 1,1                 ----------------   34.5000097660 [s]
MESH - NCFG (Ack)      NODE 1,1                 ----------------   39.5004101540 [s]
MESH - NCFG (Ack)      TO NODE 1 FOR NODE 1,1   ----------------   45.4003417950 [s]

                                   NODO 1,2 (12)
MESH - NCFG (NetDes)   TO NODE 1 FOR NODE 1,2   ----------------   25.9000683590 [s]
MESH - NCFG (NetDes)   NODE 1,2                 ----------------   29.7003417950 [s]
MESH - NENT (Request)  NODE 1,2                 ----------------   30.2000292980 [s]
MESH - NENT (Request)  TO NODE 1 FOR NODE 1,2   ----------------   30.5000292980 [s]
MESH - NCFG (OpenIE)   TO NODE 1 FOR NODE 1,2   ----------------   34.8003417950 [s]
MESH - NCFG (OpenIE)   NODE 1,2                 ----------------   39.9904785130 [s]
MESH - NENT (Ack)      NODE 1,2                 ----------------   40.3000000000 [s]
MESH - NENT (Ack)      TO NODE 1 FOR NODE 1,2   ----------------   40.6000292980 [s]
MESH - NENT (CloseID)  TO NODE 1 FOR NODE 1,2   ----------------   41.2000390640 [s]
MESH - NENT (CloseID)  NODE 1,2                 ----------------   41.6000000000 [s]
MESH - NCFG (Ack)      NODE 1,2                 ----------------   47.6904785130 [s]
MESH - NCFG (Ack)      TO NODE 1 FOR NODE 1,2   ----------------   73.3000683590 [s]

                                   NODO 1,3 (13)
MESH - NCFG (NetDes)   TO NODE 1 FOR NODE 1,3   ----------------   21.7004101540 [s]
MESH - NCFG (NetDes)   NODE 1,3                 ----------------   25.0003417950 [s]
MESH - NENT (Request)  NODE 1,3                 ----------------   25.5000000000 [s]
MESH - NENT (Request)  TO NODE 1 FOR NODE 1,3   ----------------   26.0000000000 [s]
MESH - NCFG (OpenIE)   TO NODE 1 FOR NODE 1,3   ----------------   29.9003417950 [s]
MESH - NCFG (OpenIE)   NODE 1,3                 ----------------   34.4001367180 [s]
MESH - NENT (Ack)      NODE 1,3                 ----------------   34.9000488300 [s]
MESH - NENT (Ack)      TO NODE 1 FOR NODE 1,3   ----------------   35.0000488300 [s]
MESH - NENT (CloseID)  TO NODE 1 FOR NODE 1,3   ----------------   35.6000390640 [s]
MESH - NENT (CloseID)  NODE 1,3                 ----------------   36.1000488300 [s]
MESH - NCFG (Ack)      NODE 1,3                 ----------------   41.4002050770 [s]
MESH - NCFG (Ack)      TO NODE 1 FOR NODE 1,3   ----------------   47.5904785130 [s]

                                   NODO 2,1 (14)
```

```
MESH - NCFG (Ack)      TO NODE 2 FOR NODE 2,1   -----------------   21.5001367180 [s]
                                  NODO 2,2 (15)
MESH - NCFG (Ack)      TO NODE 2 FOR NODE 2,2   -----------------   25.3001367180 [s]
                                  NODO 2,3 (16)
MESH - NCFG (Ack)      TO NODE 2 FOR NODE 2,3   -----------------   34.5004101540 [s]
                                  NODO 3,1 (17)
 MESH - NCFG (Ack)      TO NODE 3 FOR NODE 3,1   -----------------   7.7002050770 [s]
                                  NODO 3,2 (18)
MESH - NCFG (Ack)      TO NODE 3 FOR NODE 3,2   -----------------   19.2003417950 [s]
                                  NODO 3,3 (19)
MESH - NCFG (Ack)      TO NODE 3 FOR NODE 3,3   -----------------   29.7002050770 [s]
                                  NODO 4,1 (20)
MESH - NCFG (Ack)      TO NODE 4 FOR NODE 4,1   -----------------   91.7004101540 [s]
                                   NODO 4,2 (21)
MESH - NCFG (Ack)      TO NODE 4 FOR NODE 4,2   -----------------   72.2000683590 [s]
                                   NODO 4,3 (22)
MESH - NCFG (Ack)      TO NODE 4 FOR NODE 4,3   -----------------   50.9003417950 [s]
                                   NODO 5,1 (23)
MESH - NCFG (Ack)      TO NODE 5 FOR NODE 5,1   -----------------   54.7904785130 [s]
                                   NODO 5,2 (24)
MESH - NCFG (Ack)      TO NODE 5 FOR NODE 5,2   -----------------   11.6004101540 [s]
                                   NODO 5,3 (25)
MESH - NCFG (Ack)      TO NODE 5 FOR NODE 5,3   -----------------   12.3001367180 [s]
                                    NODO 6,1 (26)
MESH - NCFG (Ack)      TO NODE 6 FOR NODE 6,1   -----------------   13.9001367180 [s]
                                   NODO 6,2 (27)
 MESH - NCFG (Ack)      TO NODE 6 FOR NODE 6,2   -----------------   8.4904785130 [s]
                                   NODO 6,3 (28)
MESH - NCFG (Ack)      TO NODE 6 FOR NODE 6,3   -----------------   12.0004101540 [s]
                                   NODO 7,1 (29)
MESH - NCFG (Ack)      TO NODE 7 FOR NODE 7,1   -----------------   40.5001367180 [s]
                                   NODO 7,2 (30)
MESH - NCFG (Ack)      TO NODE 7 FOR NODE 7,2   -----------------   60.0000683590 [s]
                                   NODO 7,3 (31)
MESH - NCFG (Ack)      TO NODE 7 FOR NODE 7,3   -----------------   32.7004101540 [s]
                                   NODO 8,1 (32)
MESH - NCFG (Ack)      TO NODE 8 FOR NODE 8,1   -----------------   12.5000683590 [s]
                                   NODO 8,2 (33)
MESH - NCFG (Ack)      TO NODE 8 FOR NODE 8,2   -----------------   45.8002734360 [s]
                                  NODO 8,3 (34)
MESH - NCFG (Ack)      TO NODE 8 FOR NODE 8,3   -----------------   40.9000683590 [s]
                                  NODO 9,1 (35)
MESH - NCFG (Ack)      TO NODE 9 FOR NODE 9,1   -----------------   51.4000683590 [s]
                                  NODO 9,2 (36)
MESH - NCFG (Ack)      TO NODE 9 FOR NODE 9,2   -----------------   46.0904785130 [s]
                                  NODO 9,3 (37)
MESH - NCFG (Ack)      TO NODE 9 FOR NODE 9,3   -----------------   46.5002734360 [s]
                                  NODO 10,1 (38)
MESH - NCFG (Ack)     TO NODE 10 FOR NODE 10,1   -----------------   33.7004101540 [s]
                                   NODO 10,2 (39)
MESH - NCFG (Ack)     TO NODE 10 FOR NODE 10,2   -----------------   10.2002734360 [s]
                                    NODO 10,3 (40)
MESH - NCFG (Ack)     TO NODE 10 FOR NODE 10,3   -----------------   15.1002050770 [s]
```

Tabla 5.2. Resultados detallados del modelo de registro de nodos para el segundo nivel jerárquico.

El tercer nivel ya no se incluye, pero es de esperarse que los tiempos aumenten, ya que es necesario transmitir 18 mensajes de control para que se pueda dar de alta el nodo.

Con el fin de analizar un poco más a fondo lo que ocurre durante las transmisiones de los 6 mensajes de control para los primeros 10 nodos debemos de analizar la tabla 5.3. En esta tabla podemos ver el número de oportunidad de transmisión que le corresponde a cada mensaje. Podemos ver los mensajes MSH – NENT con su respectiva numeración, cuentan con 7 oportunidades de transmisión por frame. También podemos ver los mensajes MSH – NCFG., con sus respectivas 7 oportunidades de transmisión por frame.

NENT							NCFG						
2	10	8	7	9	6	4	7	8	9	10			
1	3	5								1			2
		7						3		4		5	
		1			8		6						
		3			2								
	9	1			4								1
		2		10	6								
	5	4			3			2					
	8	6		7	9					3	4		
		5			10								
							6	7	8	9			
							5					10	

Tabla 5.3. Distribución de las oportunidades de transmisión en los frames.

Se usaron los mismos parámetros que para la tabla 5.2. En este caso, podemos observar de manera gráfica la distribución de los mensajes. El número que aparece en la casilla corresponde al nodo. Podemos ver que el resultado es el esperado ya que se transmiten los mensajes MSH – NENT y MSH – NCFG en el orden esperado. Las TO vacías es por la manera en que se comporta el algoritmo y se explica ya que después de que gana el NENT entra en competencia de nuevo para el NCFG.

Además, hay que tener en cuenta que todo esto puede variar dependiendo los valores utilizados (Holdoff Exponent, Next Xm Mx, número de nodos, número de hijos de los nodos) y el carácter aleatorio de la función NENT ya que la función rand() utilizada depende de el valor de la semilla (función srand()) que el valor que se utilizó fue constante y fue de uno.

Si continuáramos implementando una tabla igual para tomar en cuenta gráficamente las oportunidades de transmisión que se requieren para dar de alta los 100 nodos de nuestro modelo de simulación en la red Mesh podríamos abarcar muchísimas hojas. Es por eso, que basándonos en la tabla 5.3 y en los mismos parámetros se agregó el anexo 4 y 5. En estos anexos se modificó la presentación de esta tabla, primero se dividió por el tipo de mensaje de control. Posteriormente podemos ver en la primer columna el nodo que está transmitiendo, en la segunda columna podemos ver el tipo de mensaje que se está transmitiendo y finalmente en la tercer columna podemos ver el tiempo de transmisión que se está utilizando para que los nodos envíen el mensaje de control. Con la presentación que se brinda en estos anexos, no se están tomando en cuenta los espacios vacíos y se ahorra una gran cantidad de espacio. Recordemos la importancia del valor de la semilla que se le está dando en la función srand(). Si el valor de la semilla cambia, no se van a obtener los mimos resultados aunque se tengan los mismos parámetros de simulación.

5.5 Comportamiento del algoritmo cuando es número de nodos vs tiempo de registro

Con el propósito de obtener y visualizar los tiempos en que los nodos hacen el intercambio de sus mensajes MSH - NENT y MSH - NCFG con la estación base y los demás nodos fue necesario realizar las siguientes gráficas.

La gráfica de la figura 5.6 nos muestra el comportamiento que tienen los 100 nodos de la red durante el registro de cada uno. Tiene escala logarítmica en base 10 para el eje vertical (tiempo en segundos) y además se tomó en cuenta el promedio de registro de los nodos para cada nivel. Recordemos que los nodos del primer nivel son 10, los nodos del segundo nivel son 30 y los nodos del tercer nivel son 60 para nuestro escenario de simulación. Los parámetros utilizados fueron los óptimos, tal como se ha comentado en los anteriores capítulos, para los nodos se utiliza un NextXmMx lo más variado posible, del 0 al 31 y repitiéndose para el nodo 32 y así sucesivamente, se usaron distintos HoldoffExponent.

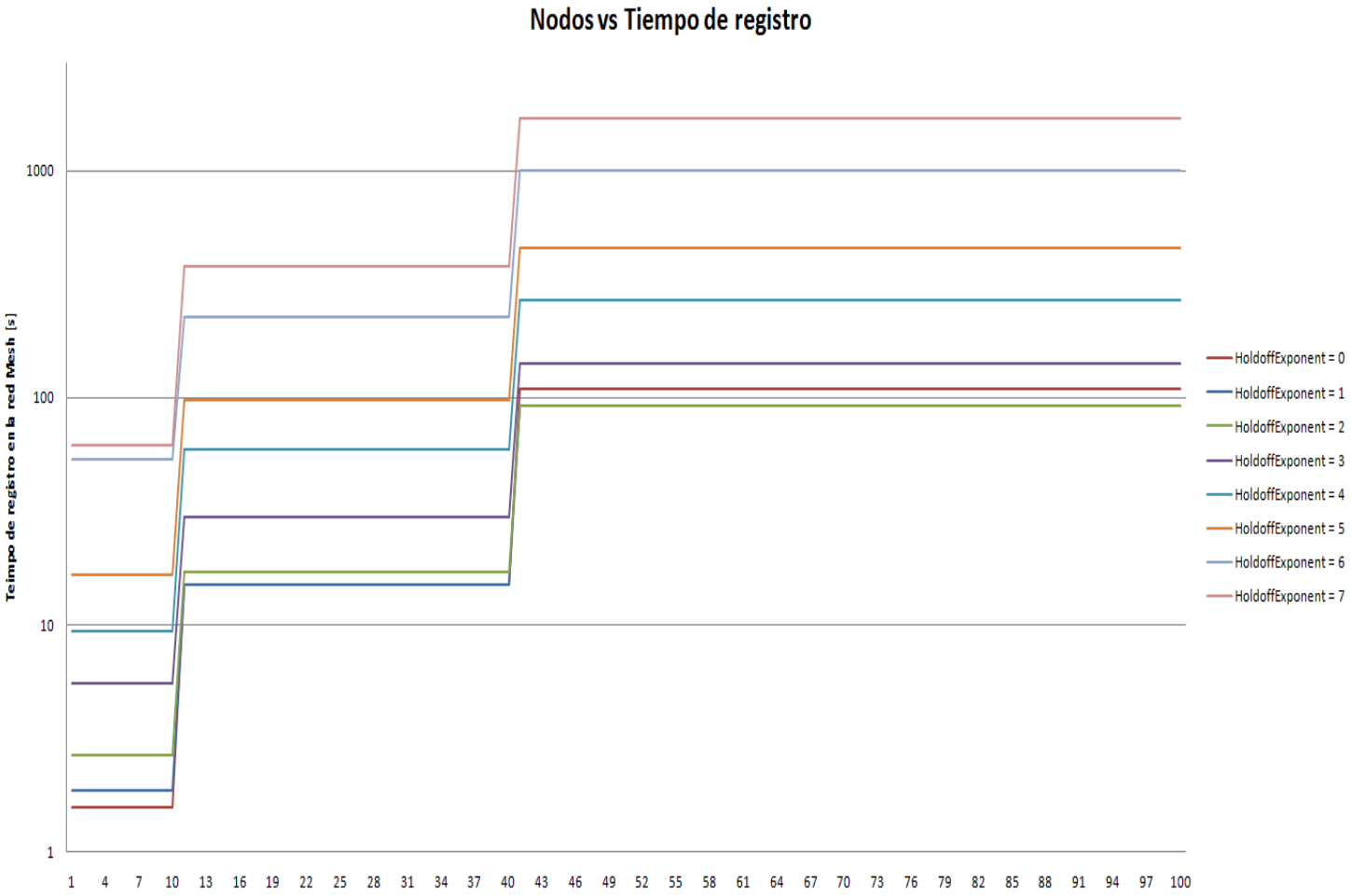


Figura 5.6. Gráfica de 100 de nodos vs tiempo de registro a la red, para los 7 HoldoffExponent NextXmMx variados.

En la figura 5.6 se puede apreciar que los primeros 10 nodos tienen un tiempo de registro corto en comparación a los demás 90 nodos, pero los nodos del 11 al 40 tienen un comportamiento diferente ya que estos nodos son los hijos de los primeros 10 nodos. Sin embargo, los últimos 60 nodos tienen tiempos mayores de registro con respecto a los primeros 40 nodos esto se debe a que por ser hijos de los hijos, éstos emplean tiempos mayores en intercambiar todos sus mensajes NCFG y NENT hacia la estación base y a sus nodos padres. Es decir, los nodos del primer nivel requieren enviar 6 mensajes de control para registrarse, los nodos del segundo nivel requieren de 12 mensajes de control y los nodos del tercer nivel requieren de 18 mensajes de control. Debido a esto, podemos observar en la gráfica tres escalones de tiempo.

De acuerdo a la distribución de los nodos para realizar nuestro modelo los primeros 10 nodos sólo intercambia sus mensajes NCFG y NENT con la estación base pero los siguientes 30 nodos tienen que intercambiar sus mensajes con su nodo padre además de la estación base y lo mismo sucede con los nodos que faltan.

Además el comportamiento de los nodos es afectado directamente por el HoldoffExponent ya que este valor hace que el tiempo de registro de los nodos sea mayor o menor, esto se ve claramente al observar la gráfica en el HoldoffExponent = 7 y es de HoldoffExponent = 2 para el tercer nivel de nodos. Los ejemplos anteriores nos muestran que con un mayor valor de HoldoffExponent el comportamiento es completamente opuesto al del valor menor de HoldoffExponent, en especial para el primer nivel de nodos.

Es aquí en donde intervienen todas las variables para hacer que el tiempo de registro se modifique, por ejemplo, de la figura 5.6 podemos ver que con un HoldoffExponent = 2 se obtiene un valor de registro menor que si usamos HoldoffExponent = 0. Esto sucede por la cantidad de nodos que se quieren registrar, en este caso 100 nodos. Nos conviene tener una ventana más grande para evitar que se generen competencias, aunque no tan grande ya que los tiempos se hacen más grandes.

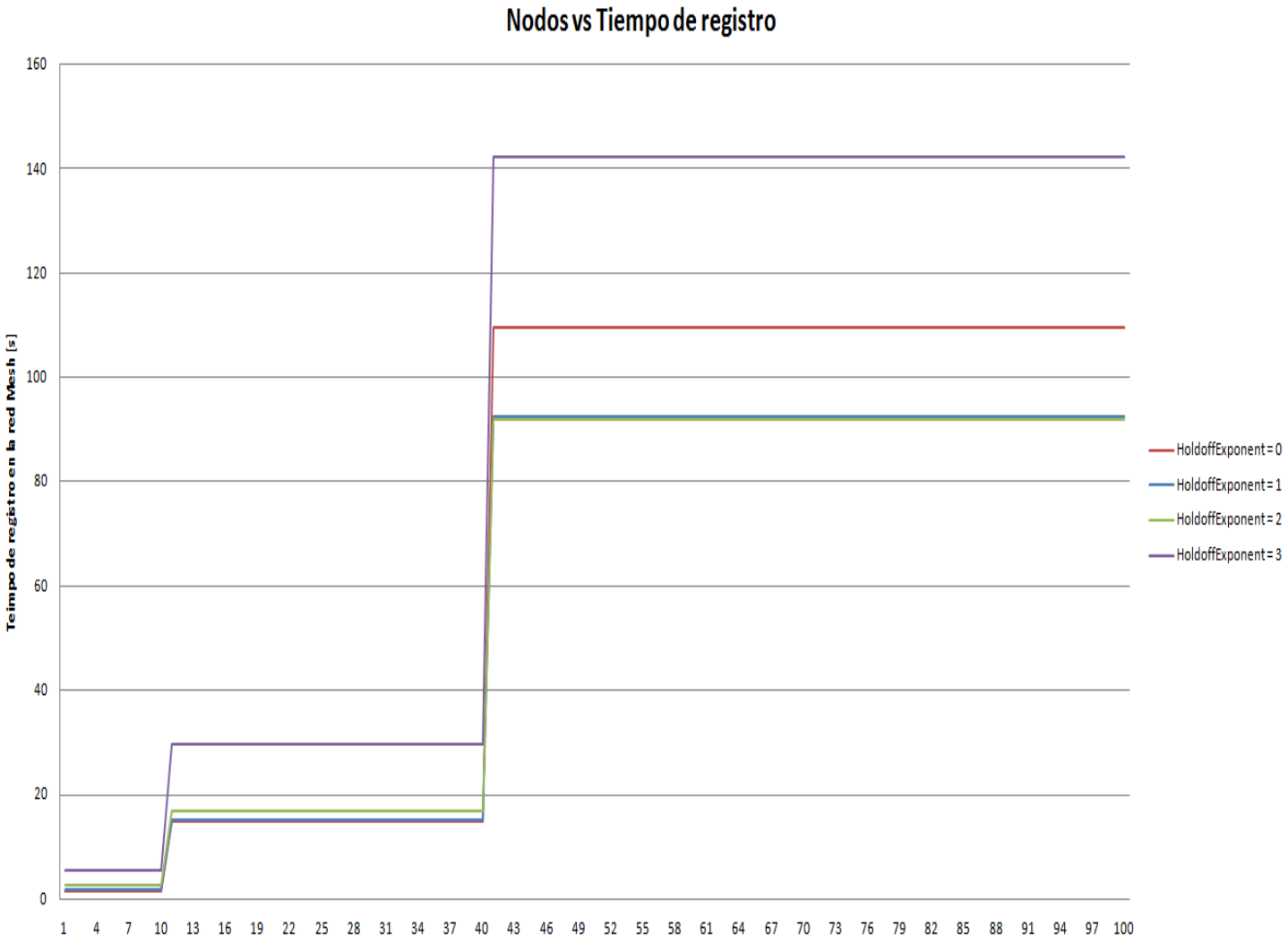


Figura 5.7. Gráfica de 100 de nodos vs tiempo de registro a la red [s], para 4 HoldoffExponent.

En la figura anterior (con la escala logarítmica también) se observa que el comportamiento de los nodos durante el registro en la red es mejor para los primeros 4 valores de HoldoffExponent, ya que el tiempo promedio que se toma para registrarse a la red es 142 [s]. En este caso al valor promedio para el valor cero y uno de HoldoffExponent es muy similar (se distingue mejor en la figura 5.6). La intención de esta gráfica es dar noción de cómo se incrementan los tiempos cada nivel, en este caso, cada vez que se cambia de nivel se aumenta el tiempo de registro en un orden de 10 veces al anterior. Esta conclusión es un poco más fácil de observar en la gráfica anterior debido a la escala.

Haciendo un contraste respecto a la figura 5.6 podemos incluir uno de los casos no más óptimos, que consiste en hacer que todos los nodos tengan el mismo intervalo de transmisión de tal forma que se generen una gran cantidad de competencias y aplicar el algoritmo de selección muchas veces. Para esto el valor de NextXmMx se puso del mismo valor (cero) para todos los nodos de la red. Los resultados los podemos ver en la figura 5.8.

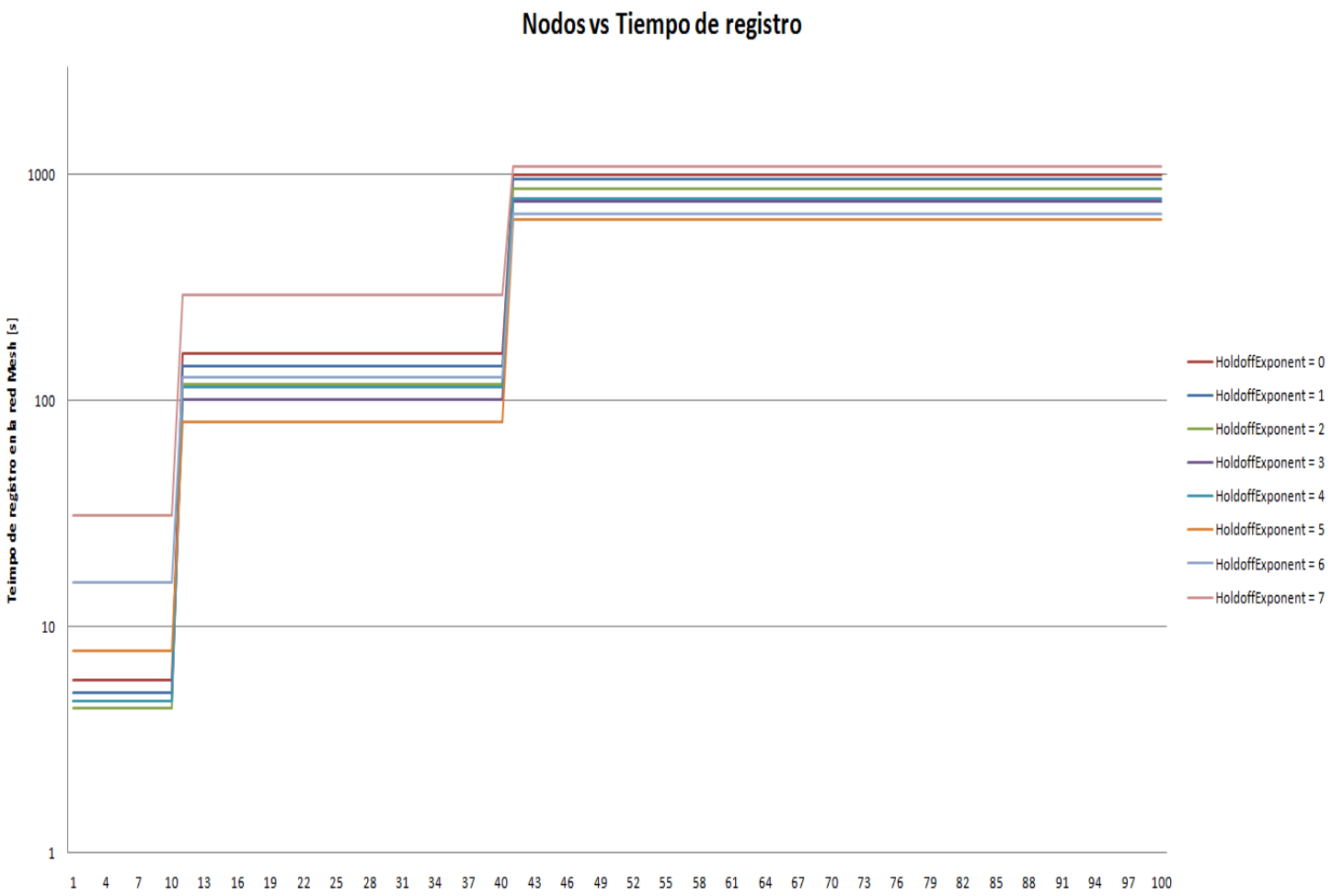


Figura 5.8. Gráfica de 100 de nodos vs tiempo de registro a la red, para los 7 HoldoffExponent NextXmMx iguales.

En este último caso podemos observar que como se tiene un mayor número de competencias debido al mismo valor de NextXmMx se requiere de mayor tiempo para que los nodos se den de alta, especialmente en el primer nivel. Aunque hay que considerar que esto se aprecia en un mayor nivel para los nodos que se encuentran en el nivel tres de la red Mesh. Si comparamos con la figura 5.6 podemos llegar a la conclusión que los tiempos para este tercer nivel se incrementan casi 10 veces para los primeros valores de HoldoffExponent. Por eso, podemos decir que no es uno de los panoramas más óptimos en el tiempo de registro de los nodos asignar el mismo valor de NextXmMx, es necesario asignarlo lo más distribuido posible.

En este punto es necesario notar que para un HoldoffExponent = 7 se obtienen tiempos promedios de registro menores al de la gráfica óptima. Esto es debido a los tamaños de las ventanas (HoldoffExponent), a los tiempos de transmisión (NextXmMx) y al número de nodos. Por eso podemos observar un valor mejor para este caso.

Como podemos observar en las figuras anteriores, existe un gran número de posibilidades y de características de cada red. Es necesario tomar en cuenta todos estos parámetros de diseño al momento de implementar una red Mesh. No es posible indicar cuáles son los parámetros de diseño óptimos ya que existen muchos escenarios de simulación. En nuestro caso, indicamos los contrastes para el modelo establecido en el punto 5.3. Se estableció un determinado número de hijos por nodos y un número de nodos fijos.

Existen tantos escenarios de simulación como combinaciones entre los distintos parámetros de la red. Es por eso, que se brindan las herramientas necesarias para poder implementarlos en los anexos 1, 2 y 3.

Capítulo 6

Conclusiones

6.1 Discusiones Finales

Debido a la implementación que se planeó para realizar la simulación de la transmisión de mensajes MESH – NCFG (Anexo 1 y 2) y principalmente al modelo para la simulación del registro de los nodos en la red (Anexo 3) en donde los mensajes de control NCFG y NENT intervienen directamente en el registro del nodo, es posible determinar el tiempo en que se transmite exactamente cada mensaje de control. Para esto, la ventaja principal que se tiene es que los parámetros que intervienen en el tiempo de registro (Next Xm Mx, Holdoff Exponent y el número de nodos) son fácilmente modificables en el código adjunto, por tanto, es posible obtener distintos escenarios de simulación, tanto como sea posible combinar estos parámetros.

En esta tesis, se muestra el comportamiento principal de estos parámetros y algunos escenarios de simulación que son los óptimos tomando el cuenta los parámetros que intervienen y el tiempo de registro que se busca sea el menor. Los resultados arrojados por cada escenario de simulación son fácilmente manipulables para realizar las gráficas pertinentes y analizar los resultados.

Es importante tomar en cuenta que existen una gran variedad de resultados para cada modelo de simulación descrito, si se varía algún parámetro que interviene directamente en los tiempos de registro se afectará el resultado. Pero hay otros parámetros que no intervienen directamente pero que se mencionan a lo largo de la tesis, por ejemplo, el número de oportunidades de transmisión destinadas a los mensajes de control MESH – NCFG, el número de nodos hijos que va a tener cada nodo en la red Mesh en donde hay que tener especial atención en introducir los valores congruentes ya que ningún tipo de depuración para el usuario.

6.2 Trabajo Futuro

La forma de registro que se analizó en esta tesis contempla la forma en que el estándar IEEE 802.16 envía los mensajes MSH – NCFG. Se planteó una mejora en donde los mensajes de control MESH – NENT se pueden dividir en siete partes para optimizar los tiempos de registro y no desperdiciar nada de espacio e información en un frame. En este caso, la forma en que se analizó la forma en que se envían estos mensajes es una propuesta

basada en el número de nodo ya que se puede optimizar como indica el estándar basándose en la dirección MAC.

Para la propuesta que se indica en esta tesis se puede realizar todavía muchos aspectos que intervienen en la forma de registro, uno de ellos es considerar que el modelo no es perfecto y que se generan colisiones, para esto hay que tomar en cuenta la probabilidad de que dos o más nodos transmitan en el mismo tiempo y asignar un tiempo en el que vuelvan a transmitir.

Con el modelo propuesto y su programación en C++ se sientan las bases para realizar la implementación del algoritmo de transmisión de mensajes y la forma de registro en la red en el software de simulación de redes OPNET. Con este software es posible simular completamente el estándar IEEE 802.16 en sus diferentes versiones que existen, en este caso, la versión 2004. Una vez implementado este algoritmo es posible realizar mejoras como la de considerar los mensajes MSH – NENT por la dirección MAC que está tomando cada nodo.

La red utilizada en este momento soporta usuarios fijos en configuración Mesh. En la extensión *IEEE 802.16*e se contempla el soporte a usuarios móviles. En este caso habrá que realizar las modificaciones pertinentes ya sea en el futuro modelo optimizado para OPNET o bien en el modelo propuesto.

6.3 Contribuciones

Los resultados del modelo de registro de nodos arrojaron muchas contribuciones en el proyecto de redes WiMAX. Gracias a la implementación que se tiene en código C++ es posible implementar función por función en el modelo de OPNET haciendo las modificaciones necesarias en el nombre de las variables.

El modelo descrito sienta las bases para posteriores estudios y optimización del modelo. Se pueden implementar una gran cantidad de escenarios dependiendo de las variables que afectan directamente el modelo de registro de nodos en una red Mesh. Se proponen tres códigos anexos los cuales, analizándolos detenidamente, nos indican cómo es que se llegó a los resultados obtenidos. En cada uno de estos anexos es donde podemos encontrar las funciones que permitirán implementar el código en el simulador de redes Opnet Modeler.

La principal contribución es que el estudio realizado nos brindará facilidades para el análisis del comportamiento de las distintas variables que intervienen en el modelo de registro. Gracias a los resultados arrojados en cada uno de los anexos podemos realizar un gran número de gráficas que nos permitan elegir los valores adecuados para distintas redes Mesh, según nos convenga. En esta tesis sólo se analizaron los casos más relevantes, aunque es conveniente realizar todos los casos prácticos que sean necesarios, sólo es cuestión de hacer las modificaciones correspondientes en la declaración de constantes en el código en cada cabecera.

6.4 Conclusiones Finales

Los resultados mostrados en esta tesis pueden ser analizados para la planeación e implementación de una red que utilice el protocolo para redes inalámbricas de banda ancha, IEEE 802.16. Gracias al análisis que se realizó para los mensajes de control, se puede predecir el comportamiento en que los nodos se darán de alta en la red así como el tiempo necesario.

El algoritmo de elección que se utilizó fue implementado de tal forma que se puedan manipular todas las variables que intervienen en el proceso para registrar un nodo en la red. De esta forma, es posible realizar la implementación del algoritmo de selección para su futuro análisis y nos brinda la oportunidad de trasladarlo a un simulador de redes.

La existencia de este algoritmo nos brinda la oportunidad de que no existan colisiones en la red, aunque gráficamente y a pesar de que se optimizaron los parámetros, quedan algunas oportunidades de transmisión que se desperdician. Además, si hacemos un análisis del número de veces que un nodo transmite también podemos notar que en algunas ocasiones algunos nodos transmiten más veces que otros. Valdría la pena

utilizar este detalle para darle más importancia a un nodo para transmitir, es decir, asignarle una prioridad en base a su NodeID.

Al combinar el envió de mensajes de control tipo NENT y NCFG es cómo se obtiene el comportamiento complejo y el por qué no se aprovechan todas las oportunidades de transmisión. Si utilizáramos la forma en que se usó el NENT se aprovecharían todas las oportunidades y no se dejarían espacios en blanco. Si utilizáramos solo el algoritmo de selección para mensajes de control NCFG se aprovecharían más oportunidades de transmisión. Pero de la forma en que se usó el envío de los dos mensajes de control se aprovechan menos oportunidades de transmisión.

En base a los resultados la principal conclusión que se obtiene es que teniendo pocos nodos nos conviene usar un HoldoffExponent pequeño. Cuando se tienen muchos nodos es necesario un HoldoffExponent intermedio. A su vez, cuando se tienen pocos nodos, nos conviene tener un NextXmMx distribuido, cuando se tienen muchos nodos puede convenir tener un NextXmMx lo más parecido cuando se tienen HoldoffExponent altos. Todos estos parámetros fueron analizados en un escenario de simulación. Una de las ventajas de la implementación es que nos permite generar tantos escenarios de simulación como deseemos.

Acrónimos

3G	Third Generation
3GPP	3rd Generation Partnership Project
ADSL	Asymmetric Digital Subscriber Line
ATM	Asynchronous Transfer Mode
BE	Best Effort
BPSK	Binary Phase Shift Keying
BRAN	Broadband Radio Access Networks
BRAN	Broadband Radio Access Networks
BS	Base Stations
BWA	Broadband Wireless Access
CDMA	Code Division Multiple Access
CID	Connection ID
CRC	Cyclic Redundancy Check
DAMA	Acceso Múltiple por Asignación de Demanda
DCD	Mensaje Descriptor del Canal Descendente
DHCP	Dynamic Host Configuration Protocol
DL-MAP	Mensaje de Mapa del Canal Descendente
DOCSIS	Data Over Cable Service Interface Specification
DSL	Digital Subscriber Line
EDGE	Enhanced Data Rates for GSM Evolution
ETSI	European Telecommunications Standards Institute
FDD	Frecuency Division Duplex
GSM	Global System for Mobile Communications
HSDPA	High Speed Downlink Packet Access
HSUPA	High-Speed Uplink Packet Access
IEEE	Institute of Electrical and Electronic Engineers
ITU	International Telecomunications Union
LAN	Local Area Network
LOS	Línea de vista
MAC	Medium Access Control
MAN	Metropolitan Area Network
MANET	Mobile ad hoc Network
MSH-CSCH	Mesh Mode Schedule with Centraliceted Scheduling
MSH-DSCH	Mesh Mode Schedule with Distributed Scheduling
MSH-NCFG	Mesh Network Configurations
MSH-NENT	Mesh Network Entries
MWS	Multimedia Wireless Systems
nrtPS	Non Real-Time Polling Service
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiplexing Access
PDU	Protocol Data Units
PHY	Physical layer of the OSI model
PKM	Protocolo de Autenticación e intercambio de llaves
PMP	Point MultiPoint
PSTN	Public Switched Telephone Network
QoS	Quality of service
REG-REQ	Mensaje de Solicitud de Registro
REG-RSP	Registration Response

REQ-RSP	Respuesta de Registro
RNG-REQ	Ranging Request
RNG-REQ	Mensaje de Solicitud de Ranging
RNG-RSP	Respuesta de Ranging
rtPS	Real-Time Polling Service
SNMP	Simple Network Management Protocol
SS	Subscriber Stations
SSTG	Subscriber Station Transition Gap
TDD	Time Division Duplex
TDM	Time Division Multiplexing
TDMA	Time Division Multiple Access
TFTP	Trivial File Transfer Protocol
TO	Oportunidades de Transmisión
UCD	Mensaje Descriptor del Canal Ascendente
UGS	Unsolicited Grant Service
UL-MAP	Mensaje Mapa del Canal Ascendente
UMTS	Universal Mobile Telephone Service
VoIP	Voice over IP
WCDMA	Wideband Code Division Multiple Access
Wi-Fi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access Forum
WirelessMAN-OFDM	Wireless Metropolitan Area Networks - OFDM
WMAN	Wireless Metropolitan Area Network
WPAN	Wide Personal Area Network

Anexo 1
Código en C ++ para determinar el nodo ganador y su tiempo correspondiente

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define NUMBERS_NODES 10 //sin incluir el cero
#define TIME_7_SYMB 0.000068359
#define NCFG_FRAME 7 //sin incluir el NENT
#define TIME_SIMULATION 15 //tiempos en los que va a simular


int elected(unsigned long xmttime, unsigned long mynodeid, int nbc, unsigned long nodelistid[]);
int NextXmtMx (unsigned long xmtime, unsigned long mynodeid);
double ConvTime (unsigned long xmttime);
unsigned int xmttime=0;
unsigned long nodelistid[NUMBERS_NODES+1];

int main (void){
   unsigned long XmtHoldoffExponent[NUMBERS_NODES+1], XmtHoldoffTime, NextXmtTime1[NUMBERS_NODES+1], Nex
   tXmtTime2[NUMBERS_NODES+1], NextXmtmx[NUMBERS_NODES+1], EarliestSubsequentXmtTime[NUMBERS_NODES+1],
   node[NUMBERS_NODES+1];
      int i,j;
      printf("Algortimo de seleccion para transmitir los paquetes NCFG \n");
      NextXmtmx[1] = 0;
      NextXmtmx[2] = 0;
      NextXmtmx[3] = 0;
      NextXmtmx[4] = 0;
      NextXmtmx[5] = 0;
      NextXmtmx[6] = 0;
      NextXmtmx[7] = 0;
      NextXmtmx[8] = 0;
      NextXmtmx[9] = 0;
      NextXmtmx[10] = 0;

      for (i = 0; i<= NUMBERS_NODES; i++){
         XmtHoldoffExponent[i] = 0;
      }
      for (i = 1; i<= NUMBERS_NODES; i++){
    node[i] = i;
    XmtHoldoffTime = pow(2,(XmtHoldoffExponent[i]+4));
    NextXmtTime1[i] = pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]) + 1;
    NextXmtTime2[i]= pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]+1);
    EarliestSubsequentXmtTime[i] = NextXmtTime2[i] + XmtHoldoffTime;
    printf("\nNodo: %lu\n",node[i]);
         printf(" XmtHoldoffTime = %lu \n NextXmtMx = %lu \n NextXmtTime1 = %lu \n NextXmtTime2 = %lu      \n
         EarliestSubsequentXmtTime = %lu \n", XmtHoldoffTime, NextXmtmx[i], NextXmtTime1[i], NextXmtTime2[i],
         EarliestSubsequentXmtTime[i]);
    }

    for (xmttime = 0; xmttime<= TIME_SIMULATION; xmttime++){
       j = 1;
       for (i = 1; i<= NUMBERS_NODES; i++){
          nodelistid[i]=0;
       }
       for (i = 1; i<= NUMBERS_NODES; i++){
              //Verificar el EarliestSubsequentXmtTime para ada tiempo y calcular en su caso
          if (EarliestSubsequentXmtTime[i] == xmttime){
             //node[i] = i;
                 NextXmtTime1[i] = pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]) + xmttime + 1;
                 NextXmtTime2[i]= pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]+1) + xmttime;
                 EarliestSubsequentXmtTime[i] = NextXmtTime2[i] + XmtHoldoffTime;
                 printf("\n    Tiempo: %lu  (%10.10lf [s]) el nodo %lu necesita volver a calcular parametros",xmttime,
                 ConvTime (xmttime), node[i]);
                 printf("\nNodo: %lu\n",node[i]);
                 printf(" \n    XmtHoldoffTime = %lu \n    NextXmtMx = %lu \n    NextXmtTime1 = %lu \n    NextXmtTime2
                 = %lu      \n    EarliestSubsequentXmtTime = %lu \n", XmtHoldoffTime, NextXmtmx[i], NextXmtTime1[i],
                 NextXmtTime2[i], EarliestSubsequentXmtTime[i]);
          }

          if((xmttime >= NextXmtTime1[i] && xmttime <= NextXmtTime2[i]) || xmttime == EarliestSubsequentXmtTime[i] ){
             nodelistid[j] = i;
             printf ("\nTiempo %lu nodo %lu",xmttime,nodelistid[j]);
             j++;
          }

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

```
            printf("\nEl NODO %lu GANA, NO HAY COMPETIDORES EN EL TIEMPO %lu (%10.10lf [s])\n",  nodelistid[1],
            xmttime, ConvTime (xmttime));
            NextXmtTime1[nodelistid[1]] = EarliestSubsequentXmtTime[nodelistid[1]]; //estas dos lineas     para
            quitar el nodo hasta su EarliestSubsequentXmtTime mas cercano
            NextXmtTime2[nodelistid[1]] = EarliestSubsequentXmtTime[nodelistid[1]]; //solo hay un nodo en el pri  mer
         renglon de la lista
            continue;
         }
      i = 1;
      while (!elected(xmttime, nodelistid[i], j-1, nodelistid)){
         i++;
      }
      //hay que quitar el nodo hasta su EarliestSubsequentXmtTime mas cercano
      NextXmtTime1[nodelistid[i]] = EarliestSubsequentXmtTime[nodelistid[i]];
      NextXmtTime2[nodelistid[i]] = EarliestSubsequentXmtTime[nodelistid[i]];
   }
   return 0;
}

unsigned long inline_smear(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(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(mynodeid^xmttime);
    smear_val2 = inline_smear(mynodeid + xmttime);

   for (i = 1; i <= nbc; i++) {
      nbr_smear_val = inline_smear(nodelistid[i]^xmttime);
      if (nbr_smear_val > smear_val1) {
         printf("nbr_smear_val[%lu] = %lu > smear_val1[%lu] = %lu. Este nodo (%lu) pierde contra %lu! \n", mynodeid,
         nbr_smear_val, nodelistid[i], smear_val1, mynodeid, nodelistid[i]);
         return 0;
      }

      else if (nbr_smear_val==smear_val1) {
         printf("\nEl primer desempate\n");
         printf("nbr_smear_val[%lu] = %lu == smear_val1[%lu] = %lu!\n", mynodeid, nbr_smear_val, nodelistid[i],
         smear_val1, nodelistid[i]);

         nbr_smear_val=inline_smear(nodelistid[i]+xmttime);
         if (nbr_smear_val > smear_val2) {
            printf("\nnbr_smear_val[%lu] = %lu > smear_val2[%lu] = %lu. Este nodo (%lu) pierde contra %lu!\n",
            mynodeid, nbr_smear_val, nodelistid[i], smear_val2, mynodeid, nodelistid[i]);
            return 0;
         }
         else if (nbr_smear_val==smear_val2) {
            if ((xmttime % 2 == 0 && nodelistid[i] > mynodeid) || (xmttime % 2 == 1 && nodelistid[i] < mynodeid))
            {
               printf("\nEste nodo (%lu) pierde contra %lu!\n", mynodeid, nodelistid[i]);
               return 0;
             }
         }
      }
   }
   //printf("Este nodo %lu es el ganador de todos los competidores\n", nodelistid[i]);
   printf("Fin de la competencia\n");
   printf("\n GANO EL NODO %lu EN EL TIEMPO %d (%10.10lf [s])\n\n",mynodeid, xmttime, ConvTime (xmttime));
   return 1;
}

double ConvTime (unsigned long xmttime){
   unsigned long cociente, temp;
   double realtime;

   cociente = xmttime / (NCFG_FRAME);
   temp = xmttime - (NCFG_FRAME * cociente);
   realtime = cociente * (0.1) + (temp * TIME_7_SYMB);
      if (temp = 0)  ////////////
         realtime = cociente * (0.1) - TIME_7_SYMB; ////////////////////// sin esto existiria colision con un NENT
   return realtime;
}
```

Anexo 2
Código en C ++ para determinar el primer ciclo donde todos los nodos transmiten un mensaje MSH- NCFG por lo menos y el número de transmisiones que realizan.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define NUMBERS_NODES 25 //sin incluir el cero
//#define NUM_OF_PROCCESS 10
#define TIME_7_SYMB 0.000068359
#define NCFG_FRAME 7 //sin incluir el NENT
#define TIME_SIMULATION 1000000000 //tiempos en los que va a simular

int elected(unsigned long xmttime, unsigned long mynodeid, int nbc, unsigned long nodelistid[]);
int NextXmtMx (unsigned long xmtime, unsigned long mynodeid);
double ConvTime (unsigned long xmttime);
int Checar (void);

unsigned int xmttime=0;
unsigned long nodelistid[NUMBERS_NODES+1], Check[NUMBERS_NODES+1], Check2[NUMBERS_NODES+1],
num_trans_nodo[NUMBERS_NODES+1];

int main (void){
    unsigned long XmtHoldoffExponent[NUMBERS_NODES+1], XmtHoldoffTime, NextXmtTime1[NUMBERS_NODES+1],
    NextXmtTime2[NUMBERS_NODES+1], NextXmtmx[NUMBERS_NODES+100], EarliestSubsequentXmtTime[NUMBERS_NODES+1],
    node[NUMBERS_NODES+1];
    int i,j;
    printf("Algortimo de seleccion para transmitir los paquetes NCFG \n");

    for (i = 1; i<= 32; i++){
        NextXmtmx[i] = i-1;
    }
    for (i = 33; i<= 64; i++){
        NextXmtmx[i] = i-33;
    }
    for (i = 65; i<= 96; i++){
        NextXmtmx[i] = i-65;
    }
    for (i = 96; i<= 100; i++){
        NextXmtmx[i] = i-96;
    }
    for (i = 1; i<= NUMBERS_NODES; i++){
        Check[i] = 0;
        Check2[i] = 1;
        num_trans_nodo[i] = 0;
        XmtHoldoffExponent[i]= 7;
    }

    for (i = 1; i<= NUMBERS_NODES; i++){
        node[i] = i;
        XmtHoldoffTime = pow(2,(XmtHoldoffExponent[i]+4));
        NextXmtTime1[i] = pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]) + 1;
        NextXmtTime2[i]= pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]+1);
        EarliestSubsequentXmtTime[i] = NextXmtTime2[i] + XmtHoldoffTime;
        printf("\nNodo: %lu\n",node[i]);
        printf(" XmtHoldoffTime = %lu \n NextXmtMx = %lu \n NextXmtTime1 = %lu \n NextXmtTime2 = %lu       \n
        EarliestSubsequentXmtTime = %lu \n", XmtHoldoffTime, NextXmtmx[i], NextXmtTime1[i], NextXmtTime2[i],
        EarliestSubsequentXmtTime[i]);
    }

    for (xmttime = 0; xmttime<= TIME_SIMULATION; xmttime++){
        j = 1;
        //**********************Aumentar un (Check[n] == 1) segun haya n nodos
        if (Checar() == 1) {
            for (i = 1; i <= NUMBERS_NODES; i++){
                printf("%lu \n",num_trans_nodo[i]);
            }
            xmttime = TIME_SIMULATION;
            continue;
        }
```

```
      for (i = 1; i<= NUMBERS_NODES; i++){
         nodelistid[i]=0;
      }

      for (i = 1; i<= NUMBERS_NODES; i++){
         //Verificar el EarliestSubsequentXmtTime para ada tiempo y calcular en su caso
         if (EarliestSubsequentXmtTime[i] == xmttime){
            //node[i] = i;
            NextXmtTime1[i] = pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]) + xmttime +1;
            NextXmtTime2[i]= pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]+1) + xmttime;
            EarliestSubsequentXmtTime[i] = NextXmtTime2[i] + XmtHoldoffTime;
            //printf("\n    Tiempo: %lu  (%10.10lf [s]) el nodo %lu necesita volver a calcular parametros",xmttime,
            ConvTime (xmttime), node[i]);
            //printf("\nNodo: %lu\n",node[i]);
            //printf(" \n    XmtHoldoffTime = %lu \n    NextXmtMx = %lu \n    NextXmtTime1 = %lu \n    NextXmtTime2 =
            %lu        \n    EarliestSubsequentXmtTime = %lu \n", XmtHoldoffTime, NextXmtmx[i], NextXmtTime1[i],
            NextXmtTime2[i], EarliestSubsequentXmtTime[i]);
         }
         if((xmttime >= NextXmtTime1[i] && xmttime <= NextXmtTime2[i]) || xmttime == EarliestSubsequentXmtTime[i] ){
            nodelistid[j] = i;
            //printf ("\nTiempo %lu nodo %lu",xmttime,nodelistid[j]);
            j++;
         }
      }

      //inicia la competencia
      if (nodelistid[1]==nodelistid[2]) //no hay nodos que compitan
         continue;
      else if (nodelistid[2]==nodelistid[0]){ //solo hay un nodo
         printf("\nEl NODO %lu GANA, NO HAY COMPETIDORES EN EL TIEMPO %lu (%10.10lf [s])\n", nodelistid[1], xmttime,
         ConvTime (xmttime));
         NextXmtTime1[nodelistid[1]] = EarliestSubsequentXmtTime[nodelistid[1]]; //estas dos lineas para quitar el
         nodo hasta su EarliestSubsequentXmtTime mas cercano
         NextXmtTime2[nodelistid[1]] = EarliestSubsequentXmtTime[nodelistid[1]]; //solo hay un nodo en el primer
         renglon de la lista
         //PARA DEPURACION
         Check[nodelistid[1]] = 1;
         num_trans_nodo[nodelistid[1]]++;
         continue;
      }

      i = 1;
      while (!elected(xmttime, nodelistid[i], j-1, nodelistid)){         i++;
      }
      //hay que quitar el nodo hasta su EarliestSubsequentXmtTime mas cercano
      NextXmtTime1[nodelistid[i]] = EarliestSubsequentXmtTime[nodelistid[i]];
      NextXmtTime2[nodelistid[i]] = EarliestSubsequentXmtTime[nodelistid[i]];
      //PARA DEPURACION
      Check[nodelistid[i]] = 1;
      num_trans_nodo[nodelistid[i]]++;
   }
   return 0;
}

unsigned long inline_smear(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(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(mynodeid^xmttime);
```

```
    smear_val2 = inline_smear(mynodeid + xmttime);
    for (i = 1; i <= nbc; i++)
    {
       nbr_smear_val = inline_smear(nodelistid[i]^xmttime);
       if (nbr_smear_val > smear_val1)
       {
                return 0;}
       else if (nbr_smear_val==smear_val1)
       {
          nbr_smear_val=inline_smear(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))
             {
                   return 0;
              }
          }
       }
    }
    printf("\n GANO EL NODO %lu EN EL TIEMPO %d (%10.10lf [s])\n\n",mynodeid, xmttime, ConvTime (xmttime));
    return 1;
}

double ConvTime (unsigned long xmttime){
    unsigned long cociente, temp;
    double realtime;
    cociente = xmttime / (NCFG_FRAME);
    temp = xmttime - (NCFG_FRAME * cociente);
    realtime = cociente * (0.1) + (temp * TIME_7_SYMB);
    if (temp = 0)  ////////////
       realtime = cociente * (0.1) - TIME_7_SYMB; ////////////////////// sin esto existiria colision con un NENT
    return realtime;
}

int Checar (void){
    int k, temp = 0;
    for (k = 1; k <= NUMBERS_NODES; k++){
       if (Check[k] == Check2[k]){
          temp = temp + 1;
       }
       else if (Check[k] != Check2[k]){
          temp = temp;
       }
    }
    if (temp == NUMBERS_NODES){
       //printf ("Nodos: %d Tiempo %d %10.10lf [s]\n", NUMBERS_NODES, xmttime-1, ConvTime (xmttime-1));
       return 1;
    }
    else
    return 0;
}
```

Anexo 3
Código en C ++ para determinar el tiempo en que un nodo se registra en la red Mesh mostrando cada tiempo de cada uno de los mensajes de control.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <ctime>
#include <cstdlib>
#define NUMBERS_NODES 100 // 10 + NUMBERS_MESH_NODES_2_LEVEL + NUMBERS_MESH_NODES_3_LEVEL
#define NUMBERS_MESH_NODES_2_LEVEL 30 // NUMBERS_MESH_CHILD_2_LEVEL * NUMBERS_MESH_NODES_WITH_CHILD_2_LEVEL
#define NUMBERS_MESH_NODES_WITH_CHILD_2_LEVEL 10
#define NUMBERS_MESH_CHILD_2_LEVEL 3
#define NUMBERS_MESH_NODES_3_LEVEL 60 // NUMBERS_MESH_NODES_WITH_CHILD_3_LEVEL + NUMBERS_MESH_CHILD_3_LEVEL * 10
#define NUMBERS_MESH_NODES_WITH_CHILD_3_LEVEL 3 //POR 10
#define NUMBERS_MESH_CHILD_3_LEVEL 2
//#define NUM_OF_PROCCESS 10
#define TIME_7_SYMB 0.000068359//*1E-06
#define NCFG_FRAME 7 //sin incluir el NENT
#define TIME_SIMULATION 100000 //tiempos en los que va a simular //////////////
#define NCFG_TYPE 0
#define NENT_TYPE 1
#define NENT_HOLDOFFTIME 5  //indica cuantos frames se esperara, en vez de usar XmtHoldoffTime


unsigned long apply_selection (unsigned long time_register, int cantidad_de_nodos);
int elected(unsigned long xmttime, unsigned long mynodeid, int nbc, unsigned long nodelistid[]);
int NextXmtMx (unsigned long xmtime, unsigned long mynodeid);
double ConvTime (unsigned long xmttime);
unsigned long ConvTO_NCFG (double realtime); ///////////////////////////////////
int Checar (void);
unsigned long ConvTONCFGaNENT (unsigned long TO);
unsigned long NENT_random (unsigned long TO, int cantidad_de_nodos);

unsigned int xmttime = 0, time_register = 0, check_NENT[NUMBERS_NODES+1]; //se usa esta variable para guaradar el
valor en que se registra, no se puede usar xmttime por que en un ciclo for lo elevamos al maximo valor.
unsigned long nodelistid[NUMBERS_NODES+1], Check[NUMBERS_NODES+1], Check2[NUMBERS_NODES+1],
ConvTO,XmtHoldoffExponent[NUMBERS_NODES+1], XmtHoldoffTime, NextXmtTime1[NUMBERS_NODES+1],
NextXmtTime2[NUMBERS_NODES+1], NextXmtmx[NUMBERS_NODES+1], EarliestSubsequentXmtTime[NUMBERS_NODES+1],
node[NUMBERS_NODES+1], nodelistid_mesh[NUMBERS_MESH_CHILD_2_LEVEL+1], nodelistid_mesh2[NUMBERS_MESH_CHILD_3_LEVEL+1],
checar_trans_NENT[80000];
double realtime; ////////////////////////////////
int type, node_num, NENT_next = 0, i, cantidad_de_nodos; //variable para nodo de interes para saber cuantas veces
transmitira
double highest_register_time[NUMBERS_NODES+1]; //variable que nos ayuda a saber el tiempo mas alto en que la 1a
jerarquia se de de alta


int main (void){
   int k,n,m = 1, p,q, aux = 0;
   printf("Tiempo en que se registra un TODOS los nodos en la red Mesh\n");

   for (i = 0; i<= NUMBERS_NODES; i++){
      XmtHoldoffExponent[i] = 0;
      highest_register_time[i] = 0;
   }
   for (i = 0; i<= 5000; i++){
      checar_trans_NENT[i] = 0;
   }
   for (i = 1; i<= 32; i++){
      NextXmtmx[i] = i-1;
   }
   for (i = 33; i<= 64; i++){
      NextXmtmx[i] = i-33;
   }
   for (i = 65; i<= 96; i++){
      NextXmtmx[i] = i-65;
   }
   for (i = 97; i<= 100; i++){
      NextXmtmx[i] = i-97;
   }
```

```
    cantidad_de_nodos = NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL;

    for (k = 1; k<= NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL; k++){
       xmttime = 1;
       realtime = 0.000009766;

       srand(1); //semilla para crear siempre la misma secuencia d enumeros aleatorios en cada proceso, si no habría
       colisiones.

       for (i = 1; i<= NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL; i++){
          node[i] = i;
          XmtHoldoffTime = pow(2,(XmtHoldoffExponent[i]+4));
          NextXmtTime1[i] = pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]) + xmttime + 1;
          NextXmtTime2[i]= pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]+1) + xmttime;
          EarliestSubsequentXmtTime[i] = NextXmtTime2[i] + XmtHoldoffTime;
         }

       node_num = k;
       type = 1;
       printf("                              NODO %d\n",k);
       printf("MESH - NCFG (NetDes)     -----------------   0.0000000000 [s]\n");
       realtime =ConvTime(NENT_random(1,cantidad_de_nodos));
       realtime = realtime + 0.1;
       printf("MESH - NENT (Request)    -----------------   %10.10lf [s]\n",realtime);

       type = 0;
       ConvTO = ConvTO_NCFG(realtime);
       Realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
       printf("MESH - NCFG (OpenIE)     -----------------   %10.10lf [s]\n" ,realtime);

       ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime)) ; //parche
       type = 1;
       realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
       printf("MESH - NENT (Ack)        -----------------   %10.10lf [s]\n",realtime);

       ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime)) ; //parche
       type = 1;
       realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
       printf("MESH - NENT (CloseID)    -----------------   %10.10lf [s]\n",realtime);

       type = 0;
       ConvTO = ConvTO_NCFG(realtime);
       realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
       printf("MESH - NCFG (Ack)        -----------------   %10.10lf [s]\n\n" ,realtime);

       highest_register_time [k] = realtime;
}
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////

cantidad_de_nodos = NUMBERS_NODES - NUMBERS_MESH_NODES_3_LEVEL;

for (k = 1; k <= NUMBERS_MESH_NODES_WITH_CHILD_2_LEVEL; k++){ //Numero de nodos con hijos

    for (p = 0; p < NUMBERS_MESH_CHILD_2_LEVEL ; p++){
       nodelistid_mesh[p+1] = NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL+ k + ((k-1)*2) +
       p;
    }

    for (n = 1; n <= NUMBERS_MESH_CHILD_2_LEVEL; n++){ //Numero de hijos de    cada nodo en confiuracion MESH
       realtime = highest_register_time[k]; // se inicializa el tiempo mas alto
       xmttime = 0;
       srand(1); //semilla para crear siempre la misma secuencia d enumeros aleatorios en cada proceso, si no habría
       colisiones.

    for (i = 1; i<= NUMBERS_NODES - NUMBERS_MESH_NODES_3_LEVEL; i++){
       node[i] = i;
       XmtHoldoffTime = pow(2,(XmtHoldoffExponent[i]+4));
       NextXmtTime1[i] = pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]) + xmttime + 1;
       NextXmtTime2[i]= pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]+1) + xmttime;
       EarliestSubsequentXmtTime[i] = NextXmtTime2[i] + XmtHoldoffTime;
    }
```

```
    printf("                                                    NODO %d,%d (%d)\n", k,n,NUMBERS_NODES -
    NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL + m);

    node_num = k;
    type = 0;
    ConvTO = ConvTO_NCFG(realtime);
    realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
    while (!elected(ConvTO_NCFG(realtime), NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL + m,
    NUMBERS_MESH_CHILD_2_LEVEL, nodelistid_mesh)){ //elected(TO, node_num, numero de hijos, numero de los hijos))
        realtime =                                  ConvTime(apply_selection(ConvTO,cantidad_de_nodos));    //se usuara
        esto para simular el proceso de seleccion para la segunda jerarquia, la desventaja e sque siempre se considera
        a todos los hijos aunque no tengan nada que transmitir, pero nunca existiran colisiones
    }

    printf("MESH - NCFG (NetDes)   TO NODE %d FOR NODE %d,%d   -----------------   %10.10lf [s]\n" ,k,k,n,realtime);
    node_num = NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL + m;
    ConvTO = ConvTO_NCFG(realtime);
    realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
    printf("MESH - NCFG (NetDes)   NODE %d,%d                -----------------   %10.10lf [s]\n" ,k,n,realtime);


    ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime)) ;
    type = 1;
    realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
    printf("MESH - NENT (Request)  NODE %d,%d                -----------------   %10.10lf [s]\n",k,n,realtime);
    node_num = k;

    ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime)); //parche
    realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
    printf("MESH - NENT (Request)  TO NODE %d FOR NODE %d,%d   -----------------   %10.10lf [s]\n",k,k,n,realtime);

    type = 0;
    ConvTO = ConvTO_NCFG(realtime);
    realtime =ConvTime(apply_selection(ConvTO, cantidad_de_nodos));
    printf("MESH - NCFG (OpenIE)   TO NODE %d FOR NODE %d,%d   -----------------   %10.10lf [s]\n" ,k,k,n,realtime);
    node_num = NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL + m;
    ConvTO = ConvTO_NCFG(realtime);
    realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
    printf("MESH - NCFG (OpenIE)   NODE %d,%d                -----------------   %10.10lf [s]\n" ,k,n,realtime);


    ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime)) ;
    type = 1;
    realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
    printf("MESH - NENT (Ack)      NODE %d,%d                -----------------   %10.10lf [s]\n",k,n,realtime);
    node_num = k;
    ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime));
    realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
    printf("MESH - NENT (Ack)      TO NODE %d FOR NODE %d,%d   -----------------   %10.10lf [s]\n" ,k,k,n,realtime);

    ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime)); //parche
    realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
    printf("MESH - NENT (CloseID)  TO NODE %d FOR NODE %d,%d   -----------------   %10.10lf [s]\n" ,k,k,n,realtime);
    node_num = NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL + m;
    ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime)); //se coloco el parche
    realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
    printf("MESH - NENT (CloseID)  NODE %d,%d                -----------------   %10.10lf [s]\n" ,k,n,realtime);

    type = 0;
    ConvTO = ConvTO_NCFG(realtime);
    realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
    printf("MESH - NCFG (Ack)      NODE %d,%d                -----------------   %10.10lf [s]\n" ,k,n,realtime);
    node_num = k;
    ConvTO = ConvTO_NCFG(realtime);
    realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
    printf("MESH - NCFG (Ack)      TO NODE %d FOR NODE %d,%d   -----------------   %10.10lf [s]\n\n" ,k,k,n,realtime);

    highest_register_time[NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL + m] = realtime;
           m++;
        }
    }

////////////////////////////////////////////////////////////////////////////
```

```
m = 1;
cantidad_de_nodos = NUMBERS_NODES;

for (q = 1; q <= NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL; q++){ //10

    for (k = 1; k <= NUMBERS_MESH_NODES_WITH_CHILD_3_LEVEL; k++){ //Numero de nodos con hijos   //3

        for (p = 0; p < NUMBERS_MESH_CHILD_3_LEVEL; p++){
            nodelistid_mesh2[p+1] = NUMBERS_NODES - NUMBERS_MESH_NODES_3_LEVEL + (aux + 1) + ((aux + 1)-1) + p;
        }

        aux = aux + 1;       //variable para obtener el valor exacto para highest_register_time

        for (n = 1; n <= NUMBERS_MESH_CHILD_3_LEVEL; n++){ //Numero de hijos de cada nodo en confiuracion MESH  //2

            realtime = highest_register_time[NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL +
            aux]; // se inicializa el tiempo mas alto
            xmttime = 0;
            srand(1); //semilla para crear siempre la misma secuencia d enumeros aleatorios en cada proceso, si no
            habría colisiones.

            for (i = 1; i<= NUMBERS_NODES; i++){
                node[i] = i;
                XmtHoldoffTime = pow(2,(XmtHoldoffExponent[i]+4));
                NextXmtTime1[i] = pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]) + xmttime + 1;
                NextXmtTime2[i]= pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]+1) + xmttime;
                EarliestSubsequentXmtTime[i] = NextXmtTime2[i] + XmtHoldoffTime;
            }

            printf("\n                                                    NODO %d,%d,%d (%d)\n",q,k,n,NUMBERS_NODES -
            NUMBERS_MESH_NODES_3_LEVEL + m);
            node_num = q;
            type = 0;
            ConvTO = ConvTO_NCFG(realtime);
            realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
            while (!elected(ConvTO_NCFG(realtime), NUMBERS_NODES - NUMBERS_MESH_NODES_3_LEVEL + m,
            NUMBERS_MESH_CHILD_3_LEVEL, nodelistid_mesh2)){ //elected(TO, node_num, numero de hijos, numero de los
            hijos))
                realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
            }
            printf("MESH - NCFG (NetDes)   TO NODE %d FOR NODE %d,%d       -----------------   %10.10lf [s]\n"
            ,q,q,k,realtime);
            node_num = NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL + m;
            ConvTO = ConvTO_NCFG(realtime);
            realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
            printf("MESH - NCFG (NetDes)   TO NODE %d,%d FOR NODE %d,%d,%d   -----------------   %10.10lf [s]\n"
            ,q,k,q,k,n,realtime);
            node_num = NUMBERS_NODES - NUMBERS_MESH_NODES_3_LEVEL + m;
            ConvTO = ConvTO_NCFG(realtime);
            realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
            printf("MESH - NCFG (NetDes)   NODE %d,%d,%d                   -----------------   %10.10lf [s]\n"
            ,q,k,n,realtime);

            ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime)) ;
            type = 1;
            realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
            printf("MESH - NENT (Request)  NODE %d,%d,%d                   -----------------   %10.10lf [s]\n"
            ,q,k,n,realtime);
            node_num = NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL + m;
            ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime)); //se coloco el parche
            realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
            printf("MESH - NENT (Request)  TO NODE %d,%d FOR NODE %d,%d,%d   -----------------   %10.10lf [s]\n"
            ,q,k,q,k,n,realtime);
            node_num = q;
            ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime)); //se coloco el parche
            realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
            printf("MESH - NENT (Request)  TO NODE %d FOR NODE %d,%d       -----------------   %10.10lf [s]\n"
            ,q,q,k,realtime);

            type = 0;
            ConvTO = ConvTO_NCFG(realtime);
            realtime =ConvTime(apply_selection(ConvTO, cantidad_de_nodos));
```

```
          printf("MESH - NCFG (OpenIE)   TO NODE %d FOR NODE %d,%d       -----------------   %10.10lf [s]\n"
          ,q,q,k,realtime);
          node_num = NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL + m;
          ConvTO = ConvTO_NCFG(realtime);
          realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
          printf("MESH - NCFG (OpenIE)   TO NODE %d,%d FOR NODE %d,%d,%d   -----------------   %10.10lf [s]\n"
          ,q,k,q,k,n,realtime);
          node_num = NUMBERS_NODES - NUMBERS_MESH_NODES_3_LEVEL + m;
          ConvTO = ConvTO_NCFG(realtime);
          realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
          printf("MESH - NCFG (OpenIE)   NODE %d,%d,%d                   -----------------   %10.10lf [s]\n"
          ,q,k,n,realtime);

          ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime)) ;
          type = 1;
          realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
          printf("MESH - NENT (Ack)      NODE %d,%d,%d                   -----------------   %10.10lf [s]\n"
          ,q,k,n,realtime);
          node_num = NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL + m;
          ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime));
          realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
          printf("MESH - NENT (Ack)      TO NODE %d,%d FOR NODE %d,%d,%d   -----------------   %10.10lf [s]\n"
          ,q,k,q,k,n,realtime);
          node_num = q;
          ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime));
          realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
          printf("MESH - NENT (Ack)      TO NODE %d FOR NODE %d,%d       -----------------   %10.10lf [s]\n"
          ,q,q,k,realtime);

          ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime)); //parche
          realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
          printf("MESH - NENT (CloseID)  TO NODE %d FOR NODE %d,%d       -----------------   %10.10lf [s]\n"
          ,q,q,k,realtime);
          node_num = NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL + m;

          ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime)); //parche
          realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
          printf("MESH - NENT (CloseID)  TO NODE %d,%d FOR NODE %d,%d,%d   -----------------   %10.10lf [s]\n"
          ,q,k,q,k,n,realtime);
          node_num = NUMBERS_NODES - NUMBERS_MESH_NODES_3_LEVEL + m;
          ConvTO = ConvTONCFGaNENT (ConvTO_NCFG(realtime));
          realtime =ConvTime(NENT_random(ConvTO,cantidad_de_nodos));
          printf("MESH - NENT (CloseID)  NODE %d,%d,%d                   -----------------   %10.10lf [s]\n"
          ,q,k,n,realtime);

          type = 0;
          ConvTO = ConvTO_NCFG(realtime);
          realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
          printf("MESH - NCFG (Ack)      NODE %d,%d,%d                   -----------------   %10.10lf [s]\n"
          ,q,k,n,realtime);
          node_num = NUMBERS_NODES - NUMBERS_MESH_NODES_2_LEVEL - NUMBERS_MESH_NODES_3_LEVEL + m;
          ConvTO = ConvTO_NCFG(realtime);
          realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
          printf("MESH - NCFG (Ack)      TO NODE %d,%d FOR NODE %d,%d,%d   -----------------   %10.10lf [s]\n"
          ,q,k,q,k,n,realtime);
          node_num = q;
          ConvTO = ConvTO_NCFG(realtime);
          realtime =ConvTime(apply_selection(ConvTO,cantidad_de_nodos));
          printf("MESH - NCFG (Ack)      TO NODE %d FOR NODE %d,%d       -----------------   %10.10lf [s]\n"
          ,q,q,k,realtime);
       m++;
       }
    }


}
////////////////////////////////////////////////////////////////////////////////
    return 0;
}

unsigned long apply_selection (unsigned long time_register, int cantidad_de_nodos){
    int j;
    xmttime = time_register;
```

```
    for (i = 0; i<= cantidad_de_nodos; i++){
        Check[i] = 0;
        Check2[i] = 1;
    }

    for (xmttime; xmttime<= TIME_SIMULATION; xmttime++){
        j = 1;

        if (Check[node_num] == 1){
            time_register = xmttime -1 ; // variable para guaradar el valor correcto y se resta uno por el ciclo
            siguiente
            xmttime = TIME_SIMULATION;
            continue;
        }

        for (i = 1; i<= cantidad_de_nodos; i++){
            nodelistid[i]=0;
        }

        for (i = 1; i<= cantidad_de_nodos; i++){
            if (EarliestSubsequentXmtTime[i] <= xmttime){
                NextXmtTime1[i] = pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]) + xmttime + 1;
                NextXmtTime2[i]= pow(2,XmtHoldoffExponent[i])*(NextXmtmx[i]+1) + xmttime;
                EarliestSubsequentXmtTime[i] = NextXmtTime2[i] + XmtHoldoffTime;
            }

            if((xmttime >= NextXmtTime1[i] && xmttime <= NextXmtTime2[i]) || xmttime == EarliestSubsequentXmtTime[i] ){
                nodelistid[j] = i;
                j++;
            }
        }
        //inicia la competencia
        if (nodelistid[1]==nodelistid[2]) //no hay nodos que compitan
            continue;
        else if (nodelistid[2]==nodelistid[0]){ //solo hay un nodo
            //printf("\nEl NODO %lu GANA, NO HAY COMPETIDORES EN EL TIEMPO %lu (%10.10lf [s])\n", nodelistid[1],
            xmttime, ConvTime (xmttime));
            NextXmtTime1[nodelistid[1]] = EarliestSubsequentXmtTime[nodelistid[1]]; //estas dos lineas para quitar el
            nodo hasta su EarliestSubsequentXmtTime mas cercano
            NextXmtTime2[nodelistid[1]] = EarliestSubsequentXmtTime[nodelistid[1]]; //solo hay un nodo en el primer
            renglon de la lista
            Check[nodelistid[1]] = 1;
            continue;
        }
        i = 1;
        while (!elected(xmttime, nodelistid[i], j-1, nodelistid)){
            i++;
        }
        //hay que quitar el nodo hasta su EarliestSubsequentXmtTime mas cercano
        NextXmtTime1[nodelistid[i]] = EarliestSubsequentXmtTime[nodelistid[i]];
        NextXmtTime2[nodelistid[i]] = EarliestSubsequentXmtTime[nodelistid[i]];
        Check[nodelistid[i]] = 1; ////////////////////////////////
    }
    return time_register;
}

unsigned long inline_smear(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(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(mynodeid^xmttime);
    smear_val2 = inline_smear(mynodeid + xmttime);

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

double ConvTime (unsigned long xmttime){
   unsigned long cociente, temp;

   if(type == NCFG_TYPE){
      cociente = (xmttime ) / (NCFG_FRAME);
      temp = (xmttime ) - (NCFG_FRAME * cociente);

      if (cociente == 0){
         realtime = cociente * (0.1) + ((temp - cociente)  * TIME_7_SYMB);
      }
      else{
         realtime = cociente * (0.1) + (temp * TIME_7_SYMB);
         if (temp == 0)
            realtime = cociente * (0.1) - TIME_7_SYMB - 0.009453128;
      }
      return realtime;
   }else{
      cociente = xmttime / (7); //se usa 7 como constante porque es en lo que se divide el NENT
      temp = xmttime - (7 * cociente);
      if (cociente == 0){
         realtime = cociente * (0.1) + (temp  * 0.000009766)- 0.000009766;
      }
      else{
         realtime = cociente * (0.1) + ((temp - 1)  * 0.000009766);
            if (temp == 0)
               realtime = cociente * 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

   if(type == NCFG_TYPE){
      realtime = realtime * 10;
      x = realtime;
      realtime = realtime / 10;
      prueba = x * 0.1 + 0.000546872;
      TO= NCFG_FRAME *x + (realtime + TIME_7_SYMB * x - x * 0.1)/ 0.000068359; //se hizo sistema 3 ecuaciones y se
      despejo, para el procedimiento
```

```
        xmttime = TO ;
    }
    else if (type == NENT_TYPE){
        realtime = realtime * 10;
        x = realtime;
        realtime = realtime / 10;
        prueba = x * 0.1 + 0.000546872;

        if ( realtime < (x * 0.1 + 0.000009766 * 7))
            TO = (x + 1) * (NCFG_FRAME+ 1) - NCFG_FRAME + 5;
        if ( realtime < (x * 0.1 + 0.000009766 * 6))
            TO = (x + 1) * (NCFG_FRAME+ 1) - NCFG_FRAME + 4;
        if ( realtime < (x * 0.1 + 0.000009766 * 5))
            TO = (x + 1) * (NCFG_FRAME+ 1) - NCFG_FRAME + 3;
        if ( realtime < (x * 0.1 + 0.000009766 * 4))
            TO = (x + 1) * (NCFG_FRAME+ 1) - NCFG_FRAME + 2;
        if ( realtime < (x * 0.1 + 0.000009766 * 3))
            TO = (x + 1) * (NCFG_FRAME+ 1) - NCFG_FRAME + 1;
        if ( realtime < (x * 0.1 + 0.000009766 * 2))
            TO = (x + 1) * (NCFG_FRAME+ 1) - NCFG_FRAME + 0;
        if ( realtime < (x * 0.1 + 0.000009766 * 1))
            TO = (x + 1) * (NCFG_FRAME+ 1) - NCFG_FRAME - 1;
        xmttime = TO + 1;
    }
    return (xmttime);
}

unsigned long ConvTONCFGaNENT (unsigned long TO){ //parche para llamar al algoritmo en base a los TO de NCFG ( 7
TONCFG es 1TONENT) mas o menos es la idea
    unsigned long x, cociente, temp;
    realtime = realtime * 10;
    x = realtime;
    realtime = realtime / 10;
    TO = (x + 1) * 7 +1;
    NENT_next = 1;
    return TO;
}

unsigned long NENT_random (unsigned long TO, int cantidad_de_nodos){
    int random, k,m,j= 1, contador = 0,aux;
    unsigned int lista_competidores[NUMBERS_NODES+1];
    unsigned long x;
    realtime = realtime * 10;
    x = realtime;
    realtime = realtime / 10;
    for (m = 1; m <= 1000; m++){
        if (Checar() == 1){
            for (k =0; k<= cantidad_de_nodos; k++){
                check_NENT[k] = 0;
            }
            srand(1);
        }

        j = 1;
        contador = 0;

        for (k = 1; k <= cantidad_de_nodos; k++){
                lista_competidores[k] = 0;
        }

        for (k = 1; k <= cantidad_de_nodos; k++){
            if (check_NENT[k] == 0){
                lista_competidores[j] = k;
                j++;
            }

        }

        for (k = 1; k <= cantidad_de_nodos; k++){
            if (lista_competidores[k] != 0)
                contador = contador + 1;
        }
```

```
        random = (rand() % contador) + 1;
        if (lista_competidores[random] == node_num && check_NENT[lista_competidores[random]] == 0 &&
        checar_trans_NENT[TO] == 0){
            check_NENT[lista_competidores[random]] = 1;
            checar_trans_NENT[TO] = 1;
            break;
        }

        else{
            check_NENT[lista_competidores[random]] = 1;
            TO++;
        }
    }
    return TO;
}

int Checar (void){
    int k, temp = 0;
    for (k = 1; k <= cantidad_de_nodos; k++){
        if (check_NENT[k] == 1){
            temp = temp + 1;
        }
        else if (check_NENT[k] != 1){
            temp = temp;
        }
    }
    if (temp == cantidad_de_nodos){
        return 1;
    }
    else
        return 0;
}
```

Anexo 4
Tabla de envio de mensajes de control MESH – NENT para dar de alta los nodos en la red Mesh.

NODO	TIPO DE MENSAJE	TO	NODO	TIPO DE MENSAJE	TO	NODO	TIPO DE MENSAJE	TO	NODO	TIPO DE MENSAJE	TO
2	NENT (Request)	8	5	NENT (CloseID)	887	9	NENT (Ack)	2797	70	NENT (Ack)	4142
10	NENT (Request)	9	33	NENT (CloseID)	894	1	NENT (Ack)	2801	2	NENT (CloseID)	4146
8	NENT (Request)	10	25	NENT (CloseID)	910	35	NENT (Request)	2809	85	NENT (Ack)	4148
7	NENT (Request)	11	42	NENT (Request)	949	41	NENT (Request)	2814	94	NENT (CloseID)	4158
9	NENT (Request)	12	12	NENT (Request)	971	10	NENT (Ack)	2815	7	NENT (Ack)	4163
6	NENT (Request)	13	22	NENT (Ack)	975	58	NENT (Request)	2816	40	NENT (Ack)	4166
4	NENT (Request)	14	4	NENT (Ack)	1018	36	NENT (CloseID)	2821	86	NENT (Request)	4169
1	NENT (Request)	15	14	NENT (Ack)	1053	95	NENT (Ack)	2825	5	NENT (Ack)	4182
3	NENT (Request)	16	4	NENT (CloseID)	1062	34	NENT (Request)	2838	69	NENT (CloseID)	4183
5	NENT (Request)	17	22	NENT (CloseID)	1066	97	NENT (Ack)	2839	8	NENT (CloseID)	4199
7	NENT (Ack)	24	2	NENT (Ack)	1073	4	NENT (Request)	2849	57	NENT (CloseID)	4207
1	NENT (Ack)	31	49	NENT (Request)	1087	1	NENT (Ack)	2851	21	NENT (CloseID)	4216
8	NENT (Ack)	34	2	NENT (CloseID)	1118	6	NENT (Request)	2856	51	NENT (CloseID)	4235
3	NENT (Ack)	38	14	NENT (CloseID)	1144	65	NENT (Ack)	2868	55	NENT (Ack)	4238
2	NENT (Ack)	41	74	NENT (Request)	1152	5	NENT (Request)	2869	87	NENT (CloseID)	4246
9	NENT (Ack)	44	44	NENT (Request)	1163	68	NENT (Ack)	2875	56	NENT (Request)	4258
1	NENT (CloseID)	45	19	NENT (Request)	1179	66	NENT (CloseID)	2879	7	NENT (CloseID)	4265
4	NENT (Ack)	48	6	NENT (Request)	1198	28	NENT (Request)	2885	5	NENT (CloseID)	4286
2	NENT (CloseID)	52	53	NENT (Request)	1209	63	NENT (Request)	2893	51	NENT (CloseID)	4309
10	NENT (Ack)	54	46	NENT (Request)	1212	68	NENT (Request)	2896	8	NENT (Ack)	4323
6	NENT (Ack)	55	1	NENT (Request)	1218	3	NENT (CloseID)	2900	8	NENT (Request)	4350
5	NENT (Ack)	58	23	NENT (Request)	1219	1	NENT (CloseID)	2907	81	NENT (CloseID)	4359
4	NENT (CloseID)	59	20	NENT (Request)	1222	67	NENT (Ack)	2908	40	NENT (CloseID)	4381
3	NENT (CloseID)	62	4	NENT (Request)	1260	89	NENT (Ack)	2912	65	NENT (Ack)	4392
8	NENT (CloseID)	65	3	NENT (Request)	1279	15	NENT (CloseID)	2913	8	NENT (CloseID)	4426
6	NENT (CloseID)	66	21	NENT (Request)	1309	10	NENT (Ack)	2921	55	NENT (CloseID)	4450
7	NENT (CloseID)	68	16	NENT (Request)	1311	45	NENT (CloseID)	2934	70	NENT (CloseID)	4460
9	NENT (CloseID)	69	2	NENT (Request)	1312	3	NENT (Request)	2937	85	NENT (CloseID)	4465
5	NENT (CloseID)	73	43	NENT (Request)	1317	27	NENT (CloseID)	2939	35	NENT (Ack)	4475
10	NENT (CloseID)	76	1	NENT (Request)	1329	59	NENT (Ack)	2942	71	NENT (Ack)	4491
15	NENT (Request)	87	78	NENT (Request)	1357	77	NENT (Ack)	2951	41	NENT (Ack)	4527
2	NENT (Request)	104	4	NENT (Request)	1363	10	NENT (Ack)	2953	5	NENT (Ack)	4535
17	NENT (Request)	107	13	NENT (Request)	1394	1	NENT (CloseID)	2956	6	NENT (Ack)	4564
27	NENT (Request)	143	45	NENT (Request)	1401	33	NENT (Request)	2960	5	NENT (CloseID)	4636
3	NENT (Request)	148	1	NENT (Request)	1415	57	NENT (CloseID)	2973	59	NENT (Request)	4641
6	NENT (Request)	155	90	NENT (Request)	1455	83	NENT (Request)	2980	29	NENT (Request)	4658
37	NENT (Request)	165	66	NENT (Request)	1472	10	NENT (Ack)	2993	6	NENT (CloseID)	4671
38	NENT (Request)	169	41	NENT (Request)	1494	13	NENT (CloseID)	2996	4	NENT (Request)	4674
11	NENT (Request)	174	15	NENT (Request)	1504	9	NENT (CloseID)	3000	35	NENT (CloseID)	4686
23	NENT (Request)	189	57	NENT (Request)	1515	9	NENT (Ack)	3003	62	NENT (Request)	4694
1	NENT (Request)	193	36	NENT (Request)	1516	43	NENT (CloseID)	3017	65	NENT (CloseID)	4711
10	NENT (Request)	199	91	NENT (Request)	1520	10	NENT (CloseID)	3021	41	NENT (CloseID)	4741
29	NENT (Request)	203	5	NENT (Request)	1528	10	NENT (CloseID)	3026	61	NENT (Request)	4773
40	NENT (Request)	206	54	NENT (Request)	1530	47	NENT (Ack)	3030	32	NENT (Request)	4783
35	NENT (Request)	210	11	NENT (Request)	1537	100	NENT (Request)	3032	4	NENT (Request)	4796
7	NENT (Request)	217	95	NENT (Request)	1539	38	NENT (Request)	3049	71	NENT (CloseID)	4813
9	NENT (Request)	218	60	NENT (Request)	1544	61	NENT (CloseID)	3050	31	NENT (Request)	4820
5	NENT (Request)	221	24	NENT (Request)	1549	48	NENT (CloseID)	3051	58	NENT (Ack)	4871
13	NENT (Request)	225	61	NENT (Request)	1553	10	NENT (CloseID)	3054	4	NENT (Request)	4875
39	NENT (Request)	231	65	NENT (Request)	1580	4	NENT (Request)	3055	63	NENT (Ack)	4923
9	NENT (Request)	232	27	NENT (Request)	1586	56	NENT (Request)	3060	68	NENT (Ack)	4927
31	NENT (Request)	244	9	NENT (Request)	1590	41	NENT (Ack)	3072	50	NENT (Ack)	4935
10	NENT (Request)	248	1	NENT (Request)	1598	68	NENT (CloseID)	3083	48	NENT (Request)	4936
17	NENT (Ack)	253	3	NENT (Request)	1603	53	NENT (Request)	3087	18	NENT (Request)	4948
26	NENT (Request)	256	1	NENT (Request)	1604	98	NENT (CloseID)	3091	20	NENT (Ack)	4949
10	NENT (Request)	257	10	NENT (Request)	1605	78	NENT (CloseID)	3095	38	NENT (Ack)	4951
1	NENT (Request)	260	9	NENT (Request)	1607	10	NENT (CloseID)	3096	88	NENT (Ack)	4970
12	NENT (Request)	263	99	NENT (Request)	1609	69	NENT (CloseID)	3097	2	NENT (Request)	4976
19	NENT (Request)	272	48	NENT (Request)	1615	26	NENT (Request)	3102	33	NENT (Ack)	4990
11	NENT (Ack)	273	97	NENT (Request)	1616	8	NENT (Request)	3107	5	NENT (Ack)	5020
6	NENT (Request)	281	98	NENT (Request)	1625	49	NENT (Ack)	3108	2	NENT (Ack)	5021
1	NENT (Request)	284	7	NENT (Request)	1638	9	NENT (CloseID)	3110	28	NENT (Ack)	5036
15	NENT (Ack)	285	44	NENT (Request)	1641	5	NENT (Request)	3116	84	NENT (Request)	5048
27	NENT (Ack)	288	3	NENT (Request)	1655	3	NENT (Request)	3119	58	NENT (Ack)	5058
1	NENT (Ack)	290	14	NENT (Request)	1663	67	NENT (CloseID)	3122	64	NENT (Ack)	5081
3	NENT (Request)	292	89	NENT (Request)	1669	91	NENT (CloseID)	3125	54	NENT (Request)	5083
28	NENT (Request)	293	77	NENT (Request)	1680	11	NENT (Ack)	3139	4	NENT (Ack)	5085
2	NENT (Ack)	294	67	NENT (Request)	1683	59	NENT (CloseID)	3159	3	NENT (Ack)	5086
3	NENT (Ack)	298	1	NENT (Request)	1691	99	NENT (CloseID)	3162	83	NENT (Ack)	5103

6	NENT (Ack)	302	59	NENT (Request)	1696	97	NENT (CloseID)	3163	53	NENT (Ack)	5113
36	NENT (Request)	312	68	NENT (Request)	1701	65	NENT (CloseID)	3178	8	NENT (Request)	5114
7	NENT (Request)	313	69	NENT (Request)	1714	55	NENT (Ack)	3180	2	NENT (CloseID)	5126
6	NENT (Request)	315	51	NENT (Request)	1738	25	NENT (Ack)	3195	8	NENT (Ack)	5130
1	NENT (CloseID)	334	10	NENT (Request)	1742	1	NENT (Ack)	3201	8	NENT (Ack)	5131
9	NENT (Request)	336	9	NENT (Request)	1757	19	NENT (Ack)	3202	34	NENT (Ack)	5134
30	NENT (Request)	338	47	NENT (Request)	1763	90	NENT (Ack)	3206	47	NENT (Ack)	5137
2	NENT (CloseID)	341	96	NENT (Request)	1766	3	NENT (Ack)	3210	17	NENT (Ack)	5144
6	NENT (CloseID)	348	10	NENT (Request)	1784	70	NENT (Request)	3219	4	NENT (Ack)	5145
7	NENT (Request)	352	10	NENT (Request)	1796	2	NENT (Ack)	3235	20	NENT (CloseID)	5166
11	NENT (CloseID)	366	66	NENT (Request)	1806	89	NENT (CloseID)	3241	2	NENT (Ack)	5170
37	NENT (Ack)	374	21	NENT (Request)	1829	95	NENT (CloseID)	3245	3	NENT (CloseID)	5189
35	NENT (Ack)	375	7	NENT (Request)	1835	47	NENT (Request)	3257	5	NENT (CloseID)	5226
15	NENT (CloseID)	376	10	NENT (Request)	1839	17	NENT (Request)	3268	8	NENT (CloseID)	5231
38	NENT (Ack)	378	93	NENT (Request)	1848	96	NENT (Ack)	3278	8	NENT (CloseID)	5238
3	NENT (CloseID)	381	55	NENT (Request)	1857	60	NENT (Ack)	3279	28	NENT (CloseID)	5248
27	NENT (CloseID)	382	52	NENT (Request)	1864	2	NENT (Request)	3294	4	NENT (CloseID)	5252
29	NENT (Ack)	383	2	NENT (Request)	1869	7	NENT (Ack)	3302	38	NENT (CloseID)	5265
9	NENT (Ack)	385	25	NENT (Request)	1872	1	NENT (CloseID)	3306	50	NENT (CloseID)	5266
17	NENT (CloseID)	386	63	NENT (Request)	1881	3	NENT (CloseID)	3313	2	NENT (CloseID)	5273
7	NENT (Ack)	392	3	NENT (Request)	1887	9	NENT (Ack)	3326	58	NENT (CloseID)	5286
9	NENT (Ack)	393	22	NENT (Request)	1893	72	NENT (Request)	3340	4	NENT (CloseID)	5289
18	NENT (Request)	397	9	NENT (Request)	1894	2	NENT (CloseID)	3341	33	NENT (CloseID)	5298
12	NENT (Ack)	400	21	NENT (Ack)	1895	11	NENT (CloseID)	3353	53	NENT (CloseID)	5323
40	NENT (Ack)	404	4	NENT (Ack)	1911	41	NENT (CloseID)	3394	63	NENT (CloseID)	5337
13	NENT (Ack)	405	2	NENT (Request)	1936	80	NENT (Ack)	3395	92	NENT (Ack)	5338
24	NENT (Request)	408	67	NENT (Request)	1942	7	NENT (CloseID)	3404	68	NENT (CloseID)	5348
10	NENT (Ack)	409	80	NENT (Request)	1946	25	NENT (CloseID)	3407	34	NENT (CloseID)	5350
1	NENT (Ack)	412	4	NENT (CloseID)	1958	82	NENT (Request)	3413	17	NENT (CloseID)	5353
10	NENT (Ack)	413	73	NENT (Request)	1982	19	NENT (CloseID)	3417	58	NENT (CloseID)	5373
34	NENT (Request)	416	20	NENT (Ack)	1986	42	NENT (Request)	3424	88	NENT (CloseID)	5395
16	NENT (Request)	418	87	NENT (Request)	2011	9	NENT (CloseID)	3432	64	NENT (CloseID)	5403
5	NENT (Request)	419	37	NENT (Request)	2015	49	NENT (CloseID)	3436	62	NENT (Ack)	5414
19	NENT (Ack)	420	4	NENT (Ack)	2023	47	NENT (CloseID)	3438	9	NENT (Ack)	5422
1	NENT (Ack)	424	85	NENT (Request)	2025	10	NENT (Request)	3441	83	NENT (CloseID)	5424
9	NENT (CloseID)	432	21	NENT (CloseID)	2026	77	NENT (CloseID)	3468	72	NENT (Ack)	5429
2	NENT (Request)	434	81	NENT (Request)	2027	50	NENT (Ack)	3483	47	NENT (CloseID)	5449
7	NENT (CloseID)	439	50	NENT (Request)	2034	60	NENT (CloseID)	3494	42	NENT (Ack)	5469
8	NENT (Request)	443	7	NENT (Request)	2049	52	NENT (Request)	3497	9	NENT (CloseID)	5525
3	NENT (Ack)	446	5	NENT (Request)	2067	7	NENT (Ack)	3498	100	NENT (Ack)	5544
23	NENT (Ack)	450	4	NENT (CloseID)	2070	55	NENT (CloseID)	3499	6	NENT (Ack)	5549
3	NENT (Request)	452	20	NENT (CloseID)	2077	7	NENT (Request)	3503	62	NENT (CloseID)	5626
1	NENT (CloseID)	453	51	NENT (Request)	2083	6	NENT (Request)	3505	70	NENT (Ack)	5632
5	NENT (Ack)	458	42	NENT (Ack)	2094	52	NENT (Ack)	3530	6	NENT (CloseID)	5651
10	NENT (CloseID)	460	55	NENT (Request)	2117	90	NENT (CloseID)	3534	92	NENT (CloseID)	5655
32	NENT (Request)	461	12	NENT (Ack)	2119	73	NENT (Ack)	3540	10	NENT (Ack)	5656
1	NENT (CloseID)	467	43	NENT (Request)	2123	22	NENT (Ack)	3559	42	NENT (CloseID)	5681
29	NENT (CloseID)	474	46	NENT (Ack)	2124	43	NENT (Ack)	3584	72	NENT (CloseID)	5747
9	NENT (CloseID)	479	69	NENT (Request)	2143	66	NENT (Ack)	3594	60	NENT (Request)	5748
3	NENT (CloseID)	488	7	NENT (Request)	2147	7	NENT (CloseID)	3600	30	NENT (Request)	5763
13	NENT (CloseID)	489	53	NENT (Ack)	2148	44	NENT (Ack)	3601	4	NENT (Request)	5809
12	NENT (CloseID)	491	1	NENT (Ack)	2166	2	NENT (Ack)	3602	10	NENT (CloseID)	5863
10	NENT (CloseID)	493	6	NENT (Request)	2190	76	NENT (Request)	3624	82	NENT (Ack)	5910
28	NENT (Ack)	495	39	NENT (Request)	2202	10	NENT (Ack)	3645	70	NENT (CloseID)	5945
8	NENT (Request)	497	23	NENT (Ack)	2212	14	NENT (Ack)	3651	100	NENT (CloseID)	5969
39	NENT (Ack)	498	57	NENT (Request)	2217	6	NENT (Ack)	3653	52	NENT (Ack)	5996
5	NENT (CloseID)	502	16	NENT (Ack)	2221	54	NENT (Ack)	3668	7	NENT (Ack)	6002
37	NENT (CloseID)	507	94	NENT (Request)	2223	24	NENT (Ack)	3691	79	NENT (Ack)	6085
35	NENT (CloseID)	508	5	NENT (Request)	2231	50	NENT (CloseID)	3693	49	NENT (Ack)	6097
10	NENT (Ack)	509	1	NENT (Ack)	2239	46	NENT (Request)	3696	7	NENT (CloseID)	6106
19	NENT (CloseID)	511	70	NENT (Request)	2248	2	NENT (CloseID)	3705	7	NENT (Ack)	6160
38	NENT (CloseID)	513	1	NENT (CloseID)	2270	80	NENT (CloseID)	3715	52	NENT (CloseID)	6208
6	NENT (Ack)	518	74	NENT (Ack)	2275	67	NENT (Ack)	3720	82	NENT (CloseID)	6228
36	NENT (Ack)	535	40	NENT (Request)	2276	1	NENT (Ack)	3726	7	NENT (CloseID)	6267
23	NENT (CloseID)	541	44	NENT (Ack)	2290	87	NENT (Ack)	3728	49	NENT (CloseID)	6316
26	NENT (Ack)	543	5	NENT (Request)	2292	93	NENT (Ack)	3731	76	NENT (Ack)	6402
33	NENT (Request)	549	8	NENT (Request)	2302	94	NENT (Ack)	3736	79	NENT (CloseID)	6414
30	NENT (Ack)	551	3	NENT (Ack)	2309	75	NENT (Ack)	3741	56	NENT (Ack)	6446
25	NENT (Request)	555	8	NENT (Request)	2312	64	NENT (Ack)	3742	46	NENT (Ack)	6493
31	NENT (Ack)	558	64	NENT (Request)	2317	3	NENT (Ack)	3745	6	NENT (Ack)	6513
9	NENT (Ack)	560	6	NENT (Ack)	2325	10	NENT (CloseID)	3747	26	NENT (Ack)	6529
8	NENT (Request)	564	88	NENT (Request)	2326	6	NENT (CloseID)	3754	3	NENT (Ack)	6576
5	NENT (Request)	565	12	NENT (CloseID)	2330	9	NENT (Ack)	3759	6	NENT (CloseID)	6617
6	NENT (Ack)	568	9	NENT (Request)	2338	63	NENT (Ack)	3764	3	NENT (CloseID)	6680
40	NENT (CloseID)	575	1	NENT (CloseID)	2340	45	NENT (Ack)	3770	46	NENT (CloseID)	6705
7	NENT (Ack)	588	58	NENT (Request)	2364	22	NENT (CloseID)	3771	76	NENT (CloseID)	6718
10	NENT (CloseID)	591	75	NENT (Request)	2386	9	NENT (Ack)	3777	26	NENT (CloseID)	6740
22	NENT (Request)	597	45	NENT (Request)	2391	37	NENT (Ack)	3793	56	NENT (CloseID)	6759

6	NENT (CloseID)	605	6	NENT (Request)	2407	43	NENT (CloseID)	3797	86	NENT (Ack)	7202
9	NENT (CloseID)	607	3	NENT (CloseID)	2410	57	NENT (Ack)	3799	56	NENT (Ack)	7289
7	NENT (Ack)	608	42	NENT (CloseID)	2412	66	NENT (CloseID)	3805	59	NENT (Ack)	7380
6	NENT (CloseID)	614	23	NENT (CloseID)	2425	6	NENT (Request)	3820	8	NENT (Ack)	7381
28	NENT (CloseID)	628	6	NENT (CloseID)	2431	1	NENT (CloseID)	3831	29	NENT (Ack)	7418
39	NENT (CloseID)	630	16	NENT (CloseID)	2432	5	NENT (Ack)	3845	8	NENT (CloseID)	7485
26	NENT (CloseID)	634	8	NENT (Request)	2436	52	NENT (CloseID)	3849	4	NENT (Ack)	7488
16	NENT (Ack)	635	46	NENT (CloseID)	2443	3	NENT (CloseID)	3852	56	NENT (CloseID)	7503
4	NENT (Request)	640	53	NENT (CloseID)	2466	79	NENT (Request)	3853	86	NENT (CloseID)	7518
7	NENT (CloseID)	649	50	NENT (Request)	2468	73	NENT (CloseID)	3859	4	NENT (CloseID)	7590
2	NENT (Ack)	651	20	NENT (Request)	2476	14	NENT (CloseID)	3864	29	NENT (CloseID)	7627
34	NENT (Ack)	654	92	NENT (Request)	2490	9	NENT (CloseID)	3866	62	NENT (Ack)	7667
18	NENT (Ack)	657	44	NENT (CloseID)	2507	6	NENT (Ack)	3868	59	NENT (CloseID)	7697
36	NENT (CloseID)	668	2	NENT (Request)	2543	69	NENT (Ack)	3870	32	NENT (Ack)	7713
3	NENT (Ack)	671	66	NENT (Ack)	2548	9	NENT (CloseID)	3880	4	NENT (Ack)	7722
7	NENT (CloseID)	675	78	NENT (Ack)	2570	39	NENT (Ack)	3883	61	NENT (Ack)	7758
30	NENT (CloseID)	676	36	NENT (Ack)	2604	8	NENT (Ack)	3894	31	NENT (Ack)	7809
8	NENT (Ack)	681	74	NENT (CloseID)	2605	96	NENT (CloseID)	3897	4	NENT (CloseID)	7828
31	NENT (CloseID)	691	45	NENT (Ack)	2606	51	NENT (Ack)	3908	4	NENT (Ack)	7864
2	NENT (CloseID)	698	62	NENT (Request)	2607	5	NENT (Ack)	3911	32	NENT (CloseID)	7925
24	NENT (Ack)	711	9	NENT (Request)	2615	44	NENT (CloseID)	3925	4	NENT (CloseID)	7968
3	NENT (CloseID)	712	5	NENT (Ack)	2620	5	NENT (CloseID)	3950	62	NENT (CloseID)	7990
5	NENT (Ack)	720	48	NENT (Ack)	2633	64	NENT (CloseID)	3957	31	NENT (CloseID)	8021
16	NENT (CloseID)	726	99	NENT (Ack)	2645	49	NENT (Request)	3964	61	NENT (CloseID)	8076
32	NENT (Ack)	738	57	NENT (Ack)	2657	6	NENT (CloseID)	3971	84	NENT (Ack)	8754
18	NENT (CloseID)	741	7	NENT (Ack)	2660	63	NENT (CloseID)	3974	54	NENT (Ack)	8786
14	NENT (Request)	752	69	NENT (Ack)	2685	45	NENT (CloseID)	3981	8	NENT (Ack)	8817
5	NENT (CloseID)	761	43	NENT (Ack)	2693	21	NENT (Ack)	3999	8	NENT (CloseID)	8920
2	NENT (Request)	762	15	NENT (Ack)	2696	37	NENT (CloseID)	4004	54	NENT (CloseID)	9000
8	NENT (CloseID)	766	91	NENT (Ack)	2715	24	NENT (CloseID)	4008	84	NENT (CloseID)	9073
8	NENT (Ack)	777	5	NENT (CloseID)	2725	5	NENT (CloseID)	4013	48	NENT (Ack)	9503
34	NENT (CloseID)	787	27	NENT (Ack)	2727	7	NENT (Request)	4025	18	NENT (Ack)	9510
24	NENT (CloseID)	797	65	NENT (Request)	2728	2	NENT (Ack)	4039	2	NENT (Ack)	9590
33	NENT (Ack)	801	61	NENT (Ack)	2743	81	NENT (Ack)	4041	60	NENT (Ack)	9645
25	NENT (Ack)	823	71	NENT (Request)	2761	67	NENT (CloseID)	4042	30	NENT (Ack)	9676
8	NENT (CloseID)	824	7	NENT (CloseID)	2767	93	NENT (CloseID)	4050	2	NENT (CloseID)	9697
8	NENT (Ack)	826	98	NENT (Ack)	2772	75	NENT (CloseID)	4058	18	NENT (CloseID)	9725
32	NENT (CloseID)	831	13	NENT (Ack)	2781	39	NENT (CloseID)	4092	4	NENT (Ack)	9739
5	NENT (Ack)	845	64	NENT (Request)	2785	54	NENT (CloseID)	4095	48	NENT (CloseID)	9826
8	NENT (CloseID)	873	3	NENT (Ack)	2796	51	NENT (Ack)	4099	4	NENT (CloseID)	9844

Anexo 5
Tabla de envio de mensajes de control MESH – NCFG para dar de alta los nodos en la red Mesh.

NODO	TIPO DE MENSAJE	TO	NODO	TIPO DE MENSAJE	TO	NODO	TIPO DE MENSAJE	TO	NODO	TIPO DE MENSAJE	TO
7	NCFG (OpenIE)	8	7	NCFG (NetDes)	889	41	NCFG (NetDes)	2293	5	NCFG (Ack)	4464
8	NCFG (OpenIE)	9	1	NCFG (NetDes)	908	34	NCFG (NetDes)	2325	11	NCFG (Ack)	4531
9	NCFG (OpenIE)	10	7	NCFG (NetDes)	912	21	NCFG (Ack)	2333	46	NCFG (Ack)	4545
10	NCFG (OpenIE)	11	13	NCFG (NetDes)	953	19	NCFG (OpenIE)	2340	44	NCFG (Ack)	4548
1	NCFG (OpenIE)	19	2	NCFG (Ack)	961	7	NCFG (OpenIE)	2343	59	NCFG (NetDes)	4575
2	NCFG (OpenIE)	21	7	NCFG (Ack)	961	67	NCFG (OpenIE)	2354	7	NCFG (OpenIE)	4606
3	NCFG (OpenIE)	23	7	NCFG (Ack)	964	68	NCFG (OpenIE)	2366	30	NCFG (NetDes)	4614
4	NCFG (OpenIE)	25	1	NCFG (NetDes)	968	6	NCFG (NetDes)	2374	42	NCFG (OpenIE)	4664
5	NCFG (OpenIE)	27	22	NCFG (OpenIE)	971	75	NCFG (NetDes)	2376	61	NCFG (NetDes)	4679
6	NCFG (OpenIE)	29	6	NCFG (Ack)	975	20	NCFG (Ack)	2388	62	NCFG (NetDes)	4688
1	NCFG (Ack)	49	16	NCFG (NetDes)	975	26	NCFG (NetDes)	2396	70	NCFG (OpenIE)	4726
2	NCFG (Ack)	58	10	NCFG (NetDes)	979	2	NCFG (NetDes)	2407	67	NCFG (Ack)	4733
3	NCFG (Ack)	67	3	NCFG (Ack)	983	2	NCFG (OpenIE)	2408	25	NCFG (Ack)	4746
4	NCFG (Ack)	68	10	NCFG (NetDes)	983	50	NCFG (NetDes)	2420	58	NCFG (OpenIE)	4777
2	NCFG (NetDes)	68	34	NCFG (Ack)	988	33	NCFG (NetDes)	2423	47	NCFG (Ack)	4782
6	NCFG (Ack)	78	23	NCFG (NetDes)	990	92	NCFG (NetDes)	2424	3	NCFG (Ack)	4806
7	NCFG (Ack)	79	9	NCFG (NetDes)	1015	47	NCFG (OpenIE)	2447	54	NCFG (Ack)	4816
3	NCFG (NetDes)	79	49	NCFG (NetDes)	1016	5	NCFG (OpenIE)	2449	1	NCFG (Ack)	4823
8	NCFG (Ack)	80	10	NCFG (NetDes)	1020	70	NCFG (NetDes)	2466	96	NCFG (Ack)	4864
9	NCFG (Ack)	81	14	NCFG (OpenIE)	1024	7	NCFG (OpenIE)	2501	69	NCFG (Ack)	4865
15	NCFG (NetDes)	81	9	NCFG (NetDes)	1024	6	NCFG (OpenIE)	2502	63	NCFG (OpenIE)	4879
5	NCFG (Ack)	85	3	NCFG (NetDes)	1044	59	NCFG (OpenIE)	2504	84	NCFG (NetDes)	4880
10	NCFG (Ack)	90	10	NCFG (NetDes)	1046	38	NCFG (NetDes)	2505	75	NCFG (Ack)	4886
17	NCFG (NetDes)	93	36	NCFG (NetDes)	1050	66	NCFG (OpenIE)	2506	51	NCFG (Ack)	4894
6	NCFG (NetDes)	94	33	NCFG (Ack)	1051	78	NCFG (OpenIE)	2545	50	NCFG (OpenIE)	4905
9	NCFG (NetDes)	101	25	NCFG (Ack)	1065	45	NCFG (OpenIE)	2588	9	NCFG (Ack)	4906
10	NCFG (NetDes)	112	74	NCFG (NetDes)	1081	57	NCFG (OpenIE)	2589	68	NCFG (OpenIE)	4910
27	NCFG (NetDes)	115	24	NCFG (Ack)	1088	99	NCFG (OpenIE)	2611	48	NCFG (NetDes)	4917
6	NCFG (NetDes)	116	48	NCFG (NetDes)	1089	11	NCFG (OpenIE)	2614	9	NCFG (Ack)	4918
7	NCFG (NetDes)	120	4	NCFG (NetDes)	1100	5	NCFG (OpenIE)	2621	88	NCFG (OpenIE)	4920
5	NCFG (NetDes)	122	2	NCFG (NetDes)	1100	2	NCFG (OpenIE)	2641	10	NCFG (Ack)	4986
1	NCFG (NetDes)	124	1	NCFG (NetDes)	1105	8	NCFG (OpenIE)	2648	94	NCFG (Ack)	5011
9	NCFG (NetDes)	126	65	NCFG (NetDes)	1124	25	NCFG (OpenIE)	2655	83	NCFG (OpenIE)	5015
38	NCFG (NetDes)	134	46	NCFG (NetDes)	1128	43	NCFG (OpenIE)	2661	1	NCFG (Ack)	5035
10	NCFG (NetDes)	138	8	NCFG (Ack)	1137	60	NCFG (OpenIE)	2663	64	NCFG (OpenIE)	5036
37	NCFG (NetDes)	148	15	NCFG (NetDes)	1144	4	NCFG (Ack)	2669	8	NCFG (OpenIE)	5048
11	NCFG (NetDes)	152	7	NCFG (NetDes)	1151	71	NCFG (NetDes)	2673	10	NCFG (Ack)	5063
1	NCFG (NetDes)	158	63	NCFG (NetDes)	1154	95	NCFG (OpenIE)	2685	47	NCFG (OpenIE)	5064
23	NCFG (NetDes)	161	10	NCFG (NetDes)	1165	64	NCFG (NetDes)	2688	52	NCFG (OpenIE)	5080
10	NCFG (NetDes)	164	69	NCFG (NetDes)	1165	91	NCFG (OpenIE)	2695	10	NCFG (Ack)	5092
40	NCFG (NetDes)	164	14	NCFG (NetDes)	1180	65	NCFG (NetDes)	2705	43	NCFG (Ack)	5114
29	NCFG (NetDes)	165	20	NCFG (NetDes)	1186	9	NCFG (OpenIE)	2706	81	NCFG (Ack)	5125
3	NCFG (OpenIE)	171	53	NCFG (NetDes)	1189	66	NCFG (OpenIE)	2714	10	NCFG (Ack)	5134
35	NCFG (NetDes)	174	7	NCFG (NetDes)	1200	7	NCFG (NetDes)	2718	1	NCFG (Ack)	5178
7	NCFG (NetDes)	189	60	NCFG (NetDes)	1205	4	NCFG (Ack)	2732	14	NCFG (Ack)	5210
1	NCFG (NetDes)	192	27	NCFG (NetDes)	1219	5	NCFG (OpenIE)	2738	92	NCFG (OpenIE)	5214
13	NCFG (NetDes)	193	7	NCFG (NetDes)	1220	98	NCFG (OpenIE)	2742	87	NCFG (Ack)	5222
3	NCFG (NetDes)	193	5	NCFG (Ack)	1222	17	NCFG (NetDes)	2766	90	NCFG (Ack)	5234
9	NCFG (NetDes)	201	2	NCFG (NetDes)	1226	6	NCFG (OpenIE)	2772	7	NCFG (Ack)	5236
2	NCFG (OpenIE)	204	8	NCFG (Ack)	1232	42	NCFG (Ack)	2788	16	NCFG (Ack)	5241
6	NCFG (OpenIE)	204	6	NCFG (NetDes)	1235	58	NCFG (NetDes)	2792	49	NCFG (OpenIE)	5280
6	NCFG (NetDes)	204	5	NCFG (Ack)	1248	63	NCFG (NetDes)	2799	72	NCFG (OpenIE)	5337
31	NCFG (NetDes)	215	11	NCFG (NetDes)	1273	97	NCFG (OpenIE)	2810	4	NCFG (OpenIE)	5360
39	NCFG (NetDes)	216	21	NCFG (NetDes)	1278	52	NCFG (NetDes)	2854	22	NCFG (Ack)	5408
1	NCFG (OpenIE)	217	47	NCFG (NetDes)	1278	50	NCFG (OpenIE)	2864	85	NCFG (Ack)	5413
8	NCFG (NetDes)	219	61	NCFG (NetDes)	1279	53	NCFG (Ack)	2874	70	NCFG (Ack)	5432
12	NCFG (NetDes)	230	78	NCFG (NetDes)	1288	42	NCFG (NetDes)	2878	100	NCFG (OpenIE)	5445
36	NCFG (NetDes)	233	28	NCFG (Ack)	1312	89	NCFG (OpenIE)	2885	37	NCFG (Ack)	5455

10	NCFG (OpenIE)	234	32	NCFG (Ack)	1312	68	NCFG (NetDes)	2886	7	NCFG (Ack)	5472
19	NCFG (NetDes)	238	67	NCFG (NetDes)	1312	77	NCFG (OpenIE)	2895	46	NCFG (OpenIE)	5482
26	NCFG (NetDes)	242	43	NCFG (NetDes)	1315	8	NCFG (OpenIE)	2912	65	NCFG (Ack)	5486
15	NCFG (OpenIE)	248	3	NCFG (NetDes)	1316	2	NCFG (OpenIE)	2924	19	NCFG (Ack)	5497
17	NCFG (OpenIE)	251	14	NCFG (Ack)	1318	83	NCFG (NetDes)	2938	4	NCFG (OpenIE)	5523
9	NCFG (OpenIE)	256	59	NCFG (NetDes)	1325	8	NCFG (OpenIE)	2944	71	NCFG (Ack)	5526
7	NCFG (NetDes)	258	68	NCFG (NetDes)	1348	100	NCFG (NetDes)	2962	24	NCFG (Ack)	5528
11	NCFG (OpenIE)	259	51	NCFG (NetDes)	1369	46	NCFG (NetDes)	2982	45	NCFG (Ack)	5626
27	NCFG (OpenIE)	260	6	NCFG (OpenIE)	1374	9	NCFG (OpenIE)	2993	39	NCFG (Ack)	5635
28	NCFG (NetDes)	261	45	NCFG (NetDes)	1378	41	NCFG (OpenIE)	2995	4	NCFG (OpenIE)	5672
8	NCFG (NetDes)	267	8	NCFG (NetDes)	1378	43	NCFG (OpenIE)	3004	21	NCFG (Ack)	5687
5	NCFG (NetDes)	269	66	NCFG (NetDes)	1382	49	NCFG (OpenIE)	3021	2	NCFG (OpenIE)	5718
9	NCFG (OpenIE)	272	66	NCFG (NetDes)	1387	74	NCFG (Ack)	3037	60	NCFG (NetDes)	5740
7	NCFG (OpenIE)	277	90	NCFG (NetDes)	1403	22	NCFG (OpenIE)	3039	3	NCFG (Ack)	5768
10	NCFG (OpenIE)	290	8	NCFG (NetDes)	1416	56	NCFG (NetDes)	3042	8	NCFG (OpenIE)	5848
37	NCFG (OpenIE)	297	57	NCFG (NetDes)	1418	14	NCFG (OpenIE)	3054	6	NCFG (Ack)	5850
10	NCFG (OpenIE)	298	21	NCFG (NetDes)	1426	4	NCFG (NetDes)	3054	82	NCFG (OpenIE)	5891
3	NCFG (NetDes)	307	54	NCFG (NetDes)	1439	55	NCFG (OpenIE)	3096	51	NCFG (Ack)	5910
1	NCFG (OpenIE)	313	43	NCFG (NetDes)	1449	63	NCFG (OpenIE)	3101	64	NCFG (Ack)	5949
38	NCFG (OpenIE)	316	5	NCFG (NetDes)	1452	90	NCFG (OpenIE)	3110	1	NCFG (Ack)	5989
2	NCFG (NetDes)	320	95	NCFG (NetDes)	1455	37	NCFG (OpenIE)	3117	93	NCFG (Ack)	6003
4	NCFG (NetDes)	320	41	NCFG (NetDes)	1462	24	NCFG (OpenIE)	3123	1	NCFG (Ack)	6039
1	NCFG (OpenIE)	321	4	NCFG (OpenIE)	1463	4	NCFG (NetDes)	3126	79	NCFG (OpenIE)	6048
30	NCFG (NetDes)	324	3	NCFG (OpenIE)	1478	96	NCFG (OpenIE)	3132	57	NCFG (Ack)	6075
3	NCFG (OpenIE)	331	3	NCFG (NetDes)	1487	8	NCFG (NetDes)	3144	60	NCFG (Ack)	6140
5	NCFG (OpenIE)	337	91	NCFG (NetDes)	1487	64	NCFG (OpenIE)	3172	66	NCFG (Ack)	6153
39	NCFG (OpenIE)	338	1	NCFG (OpenIE)	1495	57	NCFG (OpenIE)	3173	68	NCFG (Ack)	6155
29	NCFG (OpenIE)	345	22	NCFG (NetDes)	1499	47	NCFG (NetDes)	3175	63	NCFG (Ack)	6174
35	NCFG (OpenIE)	350	22	NCFG (Ack)	1504	45	NCFG (OpenIE)	3181	56	NCFG (OpenIE)	6193
18	NCFG (NetDes)	368	8	NCFG (Ack)	1506	12	NCFG (Ack)	3226	83	NCFG (Ack)	6211
13	NCFG (OpenIE)	370	2	NCFG (Ack)	1507	49	NCFG (NetDes)	3255	5	NCFG (Ack)	6238
24	NCFG (NetDes)	370	5	NCFG (NetDes)	1527	6	NCFG (OpenIE)	3270	29	NCFG (OpenIE)	6243
40	NCFG (OpenIE)	371	25	NCFG (NetDes)	1529	39	NCFG (OpenIE)	3273	64	NCFG (Ack)	6248
12	NCFG (OpenIE)	377	1	NCFG (OpenIE)	1530	5	NCFG (OpenIE)	3277	47	NCFG (Ack)	6270
34	NCFG (NetDes)	378	97	NCFG (NetDes)	1533	82	NCFG (NetDes)	3278	55	NCFG (Ack)	6286
16	NCFG (NetDes)	380	8	NCFG (NetDes)	1546	51	NCFG (OpenIE)	3297	2	NCFG (Ack)	6290
6	NCFG (OpenIE)	387	99	NCFG (NetDes)	1562	66	NCFG (Ack)	3308	76	NCFG (OpenIE)	6305
9	NCFG (OpenIE)	392	5	NCFG (NetDes)	1569	72	NCFG (NetDes)	3320	35	NCFG (Ack)	6310
19	NCFG (OpenIE)	397	6	NCFG (Ack)	1570	21	NCFG (OpenIE)	3335	40	NCFG (Ack)	6312
7	NCFG (OpenIE)	407	89	NCFG (NetDes)	1579	80	NCFG (OpenIE)	3384	56	NCFG (OpenIE)	6314
23	NCFG (OpenIE)	408	44	NCFG (OpenIE)	1582	56	NCFG (NetDes)	3413	41	NCFG (Ack)	6324
32	NCFG (NetDes)	425	77	NCFG (NetDes)	1588	4	NCFG (NetDes)	3426	58	NCFG (Ack)	6360
11	NCFG (Ack)	427	98	NCFG (NetDes)	1596	4	NCFG (NetDes)	3434	32	NCFG (OpenIE)	6392
6	NCFG (OpenIE)	435	57	NCFG (NetDes)	1599	73	NCFG (OpenIE)	3441	2	NCFG (Ack)	6427
15	NCFG (Ack)	439	44	NCFG (NetDes)	1612	23	NCFG (Ack)	3463	3	NCFG (Ack)	6509
7	NCFG (OpenIE)	451	9	NCFG (NetDes)	1615	40	NCFG (OpenIE)	3482	88	NCFG (Ack)	6544
17	NCFG (Ack)	456	4	NCFG (OpenIE)	1616	44	NCFG (Ack)	3482	31	NCFG (OpenIE)	6559
5	NCFG (NetDes)	458	50	NCFG (NetDes)	1627	52	NCFG (OpenIE)	3492	72	NCFG (Ack)	6576
27	NCFG (Ack)	459	55	NCFG (NetDes)	1641	44	NCFG (OpenIE)	3529	50	NCFG (Ack)	6578
8	NCFG (NetDes)	459	1	NCFG (OpenIE)	1651	4	NCFG (OpenIE)	3532	2	NCFG (Ack)	6609
28	NCFG (OpenIE)	470	37	NCFG (NetDes)	1663	3	NCFG (OpenIE)	3545	5	NCFG (Ack)	6613
36	NCFG (OpenIE)	472	51	NCFG (NetDes)	1683	78	NCFG (Ack)	3550	6	NCFG (Ack)	6654
30	NCFG (OpenIE)	495	93	NCFG (NetDes)	1707	4	NCFG (OpenIE)	3559	50	NCFG (Ack)	6724
5	NCFG (OpenIE)	498	12	NCFG (OpenIE)	1719	43	NCFG (Ack)	3567	92	NCFG (Ack)	6748
1	NCFG (Ack)	505	4	NCFG (Ack)	1721	57	NCFG (Ack)	3581	54	NCFG (OpenIE)	6781
2	NCFG (OpenIE)	515	64	NCFG (NetDes)	1725	55	NCFG (OpenIE)	3581	7	NCFG (Ack)	6806
26	NCFG (OpenIE)	523	40	NCFG (NetDes)	1752	76	NCFG (NetDes)	3588	4	NCFG (OpenIE)	6816
33	NCFG (NetDes)	524	96	NCFG (NetDes)	1761	3	NCFG (OpenIE)	3601	9	NCFG (Ack)	6832
3	NCFG (OpenIE)	534	16	NCFG (OpenIE)	1764	54	NCFG (OpenIE)	3628	100	NCFG (Ack)	6840
8	NCFG (OpenIE)	536	3	NCFG (NetDes)	1766	91	NCFG (Ack)	3638	63	NCFG (Ack)	6937
3	NCFG (Ack)	542	5	NCFG (OpenIE)	1770	67	NCFG (OpenIE)	3640	8	NCFG (Ack)	6997
31	NCFG (OpenIE)	546	8	NCFG (NetDes)	1776	97	NCFG (Ack)	3643	9	NCFG (Ack)	7025
25	NCFG (NetDes)	548	2	NCFG (OpenIE)	1786	75	NCFG (OpenIE)	3645	10	NCFG (Ack)	7042
13	NCFG (Ack)	565	23	NCFG (OpenIE)	1789	94	NCFG (OpenIE)	3654	38	NCFG (Ack)	7083

29	NCFG (Ack)	565	6	NCFG (NetDes)	1800	93	NCFG (OpenIE)	3663	53	NCFG (Ack)	7155
12	NCFG (Ack)	572	6	NCFG (NetDes)	1802	87	NCFG (OpenIE)	3687	86	NCFG (OpenIE)	7174
2	NCFG (Ack)	574	52	NCFG (NetDes)	1804	45	NCFG (Ack)	3713	17	NCFG (Ack)	7182
8	NCFG (OpenIE)	575	55	NCFG (NetDes)	1809	89	NCFG (Ack)	3728	82	NCFG (Ack)	7198
22	NCFG (NetDes)	576	39	NCFG (NetDes)	1823	79	NCFG (NetDes)	3735	33	NCFG (Ack)	7277
35	NCFG (Ack)	579	9	NCFG (OpenIE)	1841	8	NCFG (OpenIE)	3744	5	NCFG (Ack)	7279
37	NCFG (Ack)	581	10	NCFG (OpenIE)	1842	5	NCFG (OpenIE)	3753	6	NCFG (Ack)	7321
6	NCFG (Ack)	595	81	NCFG (NetDes)	1845	31	NCFG (NetDes)	3755	59	NCFG (OpenIE)	7332
19	NCFG (Ack)	602	21	NCFG (OpenIE)	1867	99	NCFG (Ack)	3763	79	NCFG (Ack)	7374
16	NCFG (OpenIE)	603	3	NCFG (OpenIE)	1873	41	NCFG (OpenIE)	3770	8	NCFG (Ack)	7408
1	NCFG (NetDes)	612	80	NCFG (NetDes)	1875	35	NCFG (OpenIE)	3786	34	NCFG (Ack)	7484
2	NCFG (NetDes)	626	1	NCFG (OpenIE)	1884	69	NCFG (OpenIE)	3808	5	NCFG (Ack)	7574
18	NCFG (OpenIE)	627	5	NCFG (NetDes)	1892	51	NCFG (OpenIE)	3829	42	NCFG (Ack)	7603
34	NCFG (OpenIE)	631	67	NCFG (NetDes)	1902	95	NCFG (Ack)	3829	62	NCFG (OpenIE)	7610
24	NCFG (OpenIE)	633	7	NCFG (OpenIE)	1902	1	NCFG (Ack)	3840	76	NCFG (Ack)	7712
23	NCFG (Ack)	639	3	NCFG (OpenIE)	1910	2	NCFG (OpenIE)	3852	20	NCFG (Ack)	7716
3	NCFG (NetDes)	641	87	NCFG (NetDes)	1928	98	NCFG (Ack)	3890	62	NCFG (Ack)	7742
5	NCFG (OpenIE)	645	85	NCFG (NetDes)	1933	58	NCFG (OpenIE)	3900	61	NCFG (OpenIE)	7750
1	NCFG (Ack)	646	8	NCFG (NetDes)	1944	36	NCFG (Ack)	3903	7	NCFG (Ack)	7782
8	NCFG (OpenIE)	648	20	NCFG (NetDes)	1954	49	NCFG (Ack)	3937	70	NCFG (Ack)	7844
7	NCFG (Ack)	652	58	NCFG (NetDes)	1960	29	NCFG (NetDes)	3943	56	NCFG (Ack)	7904
1	NCFG (Ack)	654	73	NCFG (NetDes)	1965	81	NCFG (OpenIE)	3950	9	NCFG (Ack)	7962
2	NCFG (NetDes)	658	4	NCFG (NetDes)	1970	32	NCFG (NetDes)	3955	18	NCFG (OpenIE)	8183
9	NCFG (Ack)	670	20	NCFG (OpenIE)	1979	41	NCFG (Ack)	3956	2	NCFG (Ack)	8264
9	NCFG (Ack)	672	10	NCFG (OpenIE)	1994	10	NCFG (OpenIE)	4016	28	NCFG (Ack)	8308
38	NCFG (Ack)	678	1	NCFG (OpenIE)	1995	33	NCFG (OpenIE)	4037	5	NCFG (Ack)	8308
32	NCFG (OpenIE)	689	9	NCFG (OpenIE)	2017	6	NCFG (OpenIE)	4050	4	NCFG (Ack)	8319
39	NCFG (Ack)	726	10	NCFG (NetDes)	2019	6	NCFG (Ack)	4050	30	NCFG (OpenIE)	8324
3	NCFG (Ack)	728	6	NCFG (NetDes)	2020	18	NCFG (NetDes)	4066	58	NCFG (Ack)	8343
14	NCFG (NetDes)	729	46	NCFG (OpenIE)	2030	70	NCFG (OpenIE)	4072	49	NCFG (Ack)	8576
4	NCFG (OpenIE)	732	1	NCFG (OpenIE)	2031	3	NCFG (Ack)	4073	8	NCFG (Ack)	8616
5	NCFG (Ack)	734	42	NCFG (OpenIE)	2052	86	NCFG (NetDes)	4074	4	NCFG (Ack)	8677
40	NCFG (Ack)	735	10	NCFG (OpenIE)	2058	34	NCFG (OpenIE)	4086	86	NCFG (Ack)	8690
33	NCFG (OpenIE)	740	62	NCFG (NetDes)	2059	85	NCFG (OpenIE)	4112	84	NCFG (OpenIE)	8704
2	NCFG (NetDes)	748	53	NCFG (OpenIE)	2065	77	NCFG (Ack)	4118	2	NCFG (Ack)	8820
6	NCFG (NetDes)	751	36	NCFG (OpenIE)	2066	28	NCFG (OpenIE)	4122	46	NCFG (Ack)	8826
25	NCFG (OpenIE)	761	10	NCFG (OpenIE)	2068	55	NCFG (Ack)	4132	6	NCFG (Ack)	8827
36	NCFG (Ack)	764	45	NCFG (NetDes)	2070	26	NCFG (OpenIE)	4140	52	NCFG (Ack)	8857
12	NCFG (NetDes)	767	9	NCFG (OpenIE)	2075	7	NCFG (OpenIE)	4145	59	NCFG (Ack)	8905
9	NCFG (NetDes)	777	28	NCFG (NetDes)	2090	13	NCFG (Ack)	4146	9	NCFG (Ack)	8907
10	NCFG (Ack)	784	4	NCFG (NetDes)	2090	61	NCFG (Ack)	4186	10	NCFG (Ack)	8974
30	NCFG (Ack)	798	7	NCFG (OpenIE)	2102	20	NCFG (OpenIE)	4188	61	NCFG (Ack)	9253
1	NCFG (NetDes)	798	10	NCFG (OpenIE)	2106	27	NCFG (Ack)	4204	26	NCFG (Ack)	9311
9	NCFG (NetDes)	799	69	NCFG (NetDes)	2129	54	NCFG (NetDes)	4222	48	NCFG (OpenIE)	9399
4	NCFG (NetDes)	800	13	NCFG (OpenIE)	2130	15	NCFG (Ack)	4257	3	NCFG (Ack)	9496
3	NCFG (NetDes)	812	70	NCFG (NetDes)	2140	59	NCFG (Ack)	4286	8	NCFG (Ack)	9541
31	NCFG (Ack)	815	94	NCFG (NetDes)	2185	38	NCFG (OpenIE)	4294	60	NCFG (OpenIE)	9585
26	NCFG (Ack)	830	5	NCFG (NetDes)	2187	69	NCFG (Ack)	4325	62	NCFG (Ack)	9618
10	NCFG (Ack)	839	48	NCFG (OpenIE)	2189	65	NCFG (OpenIE)	4326	7	NCFG (Ack)	9807
16	NCFG (Ack)	840	9	NCFG (OpenIE)	2194	80	NCFG (Ack)	4352	6	NCFG (Ack)	10223
1	NCFG (NetDes)	840	74	NCFG (OpenIE)	2208	67	NCFG (Ack)	4355	7	NCFG (Ack)	10243
10	NCFG (Ack)	849	27	NCFG (OpenIE)	2209	6	NCFG (OpenIE)	4366	29	NCFG (Ack)	10385
9	NCFG (NetDes)	852	61	NCFG (OpenIE)	2222	53	NCFG (OpenIE)	4372	56	NCFG (Ack)	10595
18	NCFG (Ack)	858	15	NCFG (OpenIE)	2229	65	NCFG (Ack)	4377	31	NCFG (Ack)	10652
5	NCFG (NetDes)	863	3	NCFG (OpenIE)	2231	17	NCFG (OpenIE)	4419	84	NCFG (Ack)	10855
44	NCFG (NetDes)	869	65	NCFG (OpenIE)	2241	73	NCFG (Ack)	4442	3	NCFG (Ack)	10967
19	NCFG (NetDes)	875	2	NCFG (NetDes)	2263	52	NCFG (Ack)	4448	48	NCFG (Ack)	11272
24	NCFG (NetDes)	875	88	NCFG (NetDes)	2263	68	NCFG (Ack)	4448	60	NCFG (Ack)	11420
9	NCFG (Ack)	881	35	NCFG (NetDes)	2278	71	NCFG (OpenIE)	4453	4	NCFG (Ack)	11891
2	NCFG (OpenIE)	884	53	NCFG (NetDes)	2279	48	NCFG (Ack)	4456	4	NCFG (Ack)	12317
42	NCFG (NetDes)	886	69	NCFG (OpenIE)	2283	62	NCFG (OpenIE)	4462	32	NCFG (Ack)	12367

Referencias

[1] EEE Standard for Local and metropolitan area networks. - Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs(tm)), IEEE Std 802.15.

[2] IEEE Standard for Local and Metropolitan Area Network. - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11-1999.

[3] IEEE Standard for Local and metropolitan area networks. -Part 16: *Air Interface for Fixed Broadband Wireless Access Systems, IEEE Std 802.16-2001 TM* 8 April 2002.

[4] C. Eklund, R. B. Marks, K. L. Stanwood, "IEEE Standard 802.16: A Technical Overview of the WirelessMANTM Air Interface for Broadband Wireless Access", *IEEE Communications Magazine*, 0163-6804, pp. 98-107, Jun. 2002.

[5] IEEE Standard for Local and metropolitan area networks. - Part 16: *Air Interface for Fixed Broadband Wireless Access Systems, IEEE Std 802.16-2004, Octubre 2004.*

[6] IEEE Standard for Local and metropolitan area networks. - Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, IEEE 802.16e-2005, Febrero 2006

[7] Min Cao, Vivek Raghunathan, P. R. Kumar. "A Tractable Algorith for Fair and Efficient Uplink Scheduling of Multi-hop WiMax Mesh Networks"

[8] Bustamante Roberto, Hincapié Roberto Carlos Hincapié. Análisis, modelamiento y simulación de redes enmalladas basadas en el estándar 802.16-2004, Universidad ICESI, Noviembre de 2005

[9] V. Rangel Licea, J. E. Cota Guajardo, J. Gómez Castellanos y J. Reyes García; *Diseño de procedimientos handoff en redes inalámbricas de banda ancha basado en el protocolo IEEE 802.16*.

[10] Min Cao, Wenchao Ma, Qian Zhang , Xiaodong Wang , Wenwu Zhu. "Modelling and Performance Analysis of the Distributed Scheduler in IEEE 802.16 Mesh Mode". MobiHoc'05, May 25–27, 2005

[11] Wang Shie, Lin Chih, Fang Ku. Facilitating the Network Entry and Link Establishment Processes of IEEE 802.16 Mesh Networks. IEEE, 2007.