IMPLEMENTACIÓN DEL MECANISMO HCCA PROPUESTO EN IEEE 802.11E / D 8.0 USANDO OMNET++

ISMAEL EDGARDO MELÉNDEZ RINCÓN

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
BOGOTÁ
2005

IMPLEMENTACIÓN DEL MECANISMO HCCA PROPUESTO EN IEEE 802.11E / D 8.0 USANDO OMNET++

ISMAEL EDGARDO MELÉNDEZ RINCÓN

Proyecto de grado para optar por el título de
Magíster en Ingeniería Electrónica y de Computadores

Asesor
Ph.D. ROBERTO BUSTAMANTE MILLER

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
BOGOTÁ
2005

A mis padres y mi hermana,
por su apoyo incondicional

AGRADECIMIENTOS

El autor expresa su agradecimiento a:

Ph.D. Roberto Bustamante Miller, Director del departamento de Ingeniería Eléctrica y Electrónica de la Universidad de Los Andes, por su participación en el desarrollo de este proyecto y sus invaluables aportes.

RESUMEN

En este documento se define e implementa un modelo eficiente y flexible del mecanismo HCCA propuesto en IEEE 802.11e / D8.0, el programador y la unidad de control de admisión (ACU) utilizados son los sugeridos en dicho documento. Se desarrollan generadores de tráfico apropiados con el fin de verificar el comportamiento del modelo construido. Las pruebas realizadas permitieron comprobar el correcto funcionamiento del modelo, así como la importancia de implementar nuevos algoritmos para el programador y ACU.

TABLA DE CONTENIDO

Lista de Figuras

Lista de Tablas

1. INTRODUCCIÓN

Actualmente se pueden encontrar redes inalámbricas de área local (WLAN) en lugares como universidades, cafés o aeropuertos. Algunas características que han ayudado a la popularidad de dichas redes son: bajos costos, facilidad de implementación (no es necesario construir redes cableadas), flexibilidad, entre otras. Desde su creación se ha buscado que las WLAN tengan características similares (velocidad, seguridad, calidad de servicio (QoS)) a las redes alambradas de área local (LAN). En [1] se describe el primer estándar para WLAN (capa de control de acceso al medio (MAC) y capa física (PHY)). Se han desarrollado estándares enfocados a mejorar la velocidad [2], [3], [4] y seguridad [5] de las WLAN. Actualmente se esta trabajando en un estándar que ayude a mejorar el nivel de QoS ofrecido por dichas redes [6], puesto que la implementación inicial no garantiza la transmisión adecuada de tráfico de tiempo real (Video, VoIP, Multimedia, entre otros).

Hoy por hoy existen diversos estudios enfocados tanto a evaluar el desempeño de [6] (algunos de ellos son: [7], [8], [9], [10]) como a proponer mejoras al mecanismo HCCA ([11], [12]). Sin embargo, la mayoría de las investigaciones han sido desarrolladas utilizando paquetes comerciales (como OPNET por ejemplo) lo cuál obliga a los grupos de investigación a invertir dinero en software y resta flexibilidad a los modelos desarrollados.

Es muy importante evaluar el desempeño del futuro estándar y proponer algunas mejoras; para lograrlo se esta desarrollando el proyecto "Implementación y evaluación del desempeño de IEEE 802.11e/D 8.0", en éste trabajo se busca obtener una completa representación de [6] usando el simulador OMNeT++ [13], para luego hacer pruebas de desempeño y posteriormente proponer alternativas para su optimización. Esta investigación hace parte de dicho proyecto y busca la implementación del mecanismo HCCA de una forma eficiente, flexible y que permita la fácil interacción con los demás modelos a desarrollar en el futuro.

2. BASES TEÓRICAS

2.1. WLAN 802.11

En este documento se asume que el lector conoce detalladamente el funcionamiento de [1], de no ser así, se recomienda leer [16] para obtener una idea general del funcionamiento de dicha norma.

2.2. Mejoras propuestas en IEEE 802.11e

El objetivo principal de [6] es definir los procedimientos para soportar aplicaciones de red de área local con requerimientos de calidad de servicio (QoS), incluyendo el transporte de voz, audio y vídeo en WLANs.
Es importante mencionar que los cambios propuestos buscan mejorar la QoS ofrecida (a los usuarios) por este tipo de redes, particularmente se solucionan problemas como: retraso en la transmisión de las tramas baliza (Beacon), tiempo ilimitado de transmisión de las estaciones (dichos problemas fueron detectados inicialmente en las WLAN), entre otros. A continuación se hace una descripción de las principales mejoras propuestas en [6][1].

2.2.1. Función de Coordinación Híbrida (HCF)

La función de coordinación híbrida combina características tanto de la función de coordinación distribuida (DCF) como de la función de coordinación puntual (PCF), además incluye algunas mejoras. La HCF utiliza dos métodos para acceder al canal. El primero es un método basado en contienda, llamado EDCA; el segundo es un método libre de contienda, llamado HCCA. La unidad básica de asignación (del derecho a transmitir información a través del medio inalámbrico) es la TXOP. Cada TXOP esta definida por un tiempo de inicio y una duración máxima especificada. Las QSTAs pueden obtener TXOPs utilizando cualquiera de los métodos descritos anteriormente. Si la TXOP es obtenida

[1] En el Anexo A se definen algunos de los términos y abreviaciones comúnmente utilizados en [6].

luchando por el canal se conoce como EDCA TXOP, de lo contrario se conoce como HCCA TXOP o TXOP asignada.

Una de las principales modificaciones introducidas es la posibilidad de que el HC acceda al canal tanto en CP como en CFP, esto le permite asignar TXOPs a las QSTAs cuando lo necesiten.

2.2.1.1. Acceso al canal distribuido mejorado (EDCA)

EDCA ofrece acceso distribuido y diferenciado (al medio inalámbrico) a las QSTAs utilizando diferentes categorías de acceso (AC). Cada AC utiliza una versión mejorada de DCF para luchar por las TXOPs asignadas a la QSTA. El QAP informa los parámetros EDCA que deben ser utilizados para cada AC en algunas tramas baliza (Beacon).

Como se mencionó anteriormente el protocolo de acceso al canal propuesto para esta función es el mismo utilizado en DCF, sin embargo, el algoritmo interno ejecutado en cada QSTA para decidir cuál es la próxima trama a transmitir es diferente. Con el fin de otorgar prioridades distintas según el tipo de tráfico a transmitir, se crean 4 categorías de acceso (AC): Voz, Vídeo, Mejor Esfuerzo y Tráfico de Fondo. En la Tabla 1 se observa la clasificación propuesta. Todas las estaciones deben implementar una cola independiente para cada AC. En la Figura 1 se observa un diagrama de bloques que ilustra un modelo sugerido.

	(UP – Same as 802.1D User Priority)	Designation	Category (AC)	(Informative)
lowest	1	BK	AC_BK	Background
	2	-	AC_BK	Background
	0	BE	AC_BE	Best Effort
	3	EE	AC_BE	Best Effort
	4	CL	AC_VI	Video
	5	VI	AC_VI	Video
	6	VO	AC_VO	Voice
highest	7	NC	AC_VO	Voice

Tabla 1. Categorías de Acceso definidas en [6]. [2]

[2] Tomado de [6]. Páginas 12-13.

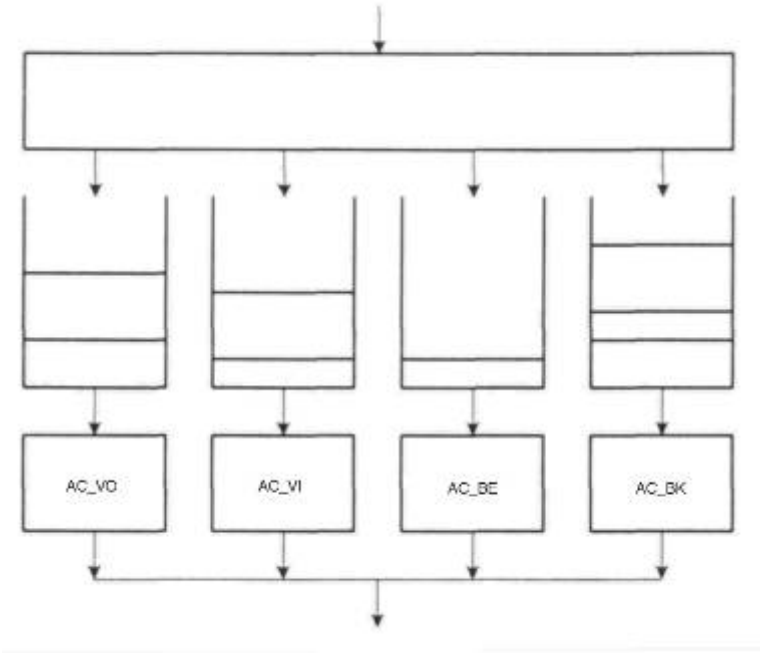

Figura 1. Modelo de Implementación de Referencia.[3]

En EDCA existen dos modos de TXOP: Iniciación y continuación. Una iniciación de una EDCA TXOP ocurre cuando las reglas establecidas en dicho algoritmo permiten a la QSTA acceder al medio. Una continuación ocurre cuando la QSTA conserva el derecho de acceder al canal al terminar una secuencia de intercambio de tramas.

Las QSTAs deben asegurarse de no usar el canal por un tiempo mayor al límite de duración de la TXOP, el cuál es especificado por el QAP en algunas tramas Baliza.

Algunos de los mecanismos utilizados para dar prioridades diferentes a cada AC son: asignación de espacios arbitrarios entre tramas (AIFS), ventanas de contención específicas por AC, entre otros. El HC es el encargado de definir el valor asignado a cada uno de los parámetros EDCA. Se define un AIFS para cada AC y su valor esta dado por:

$$AIFS[AC] = AIFSN[AC] \times SlotTime + SIFS \quad (2.2.1.1)$$

[3] Tomado de [6]. Página 71.

Como se puede observar en la ecuación (2.2.1.1), a través de la asignación de valores diferentes al parámetro *AIFSN [AC]* es posible controlar la prioridad con que cada AC hace uso del medio inalámbrico. En la Figura 2 se ilustra un ejemplo del proceso descrito anteriormente. Se observa que el AIFS establecido para las AC con mayor prioridad (Voz o AC_VO y Vídeo o AC_VI) es menor al asignado a las demás AC (Tráfico de Mejor esfuerzo o AC_BE y de fondo o AC_BK). Otra diferencia en los parámetros de cada AC radica en el tamaño de las ventanas de contención, por ejemplo el tamaño de la ventana de contención asignada a AC_VO es de tres unidades de tiempo (Slot Time) mientras que el asignado para AC_BK es de ocho. Es importante aclarar que la ventana de contención (CW) es el rango (en unidades de tiempo) del número aleatorio seleccionado por cada AC al hacer el procedimiento de "*Backoff*". La CW aumenta cuando ocurre una colisión en el medio y tiene valores máximo y mínimo (por AC) definidos por el HC.

En el ejemplo mostrado en la Figura 2 se observa que la AC que ganó el derecho a usar el medio inalámbrico fue AC_VO, dando inicio inmediatamente al intercambio de tramas RTS/CTS.

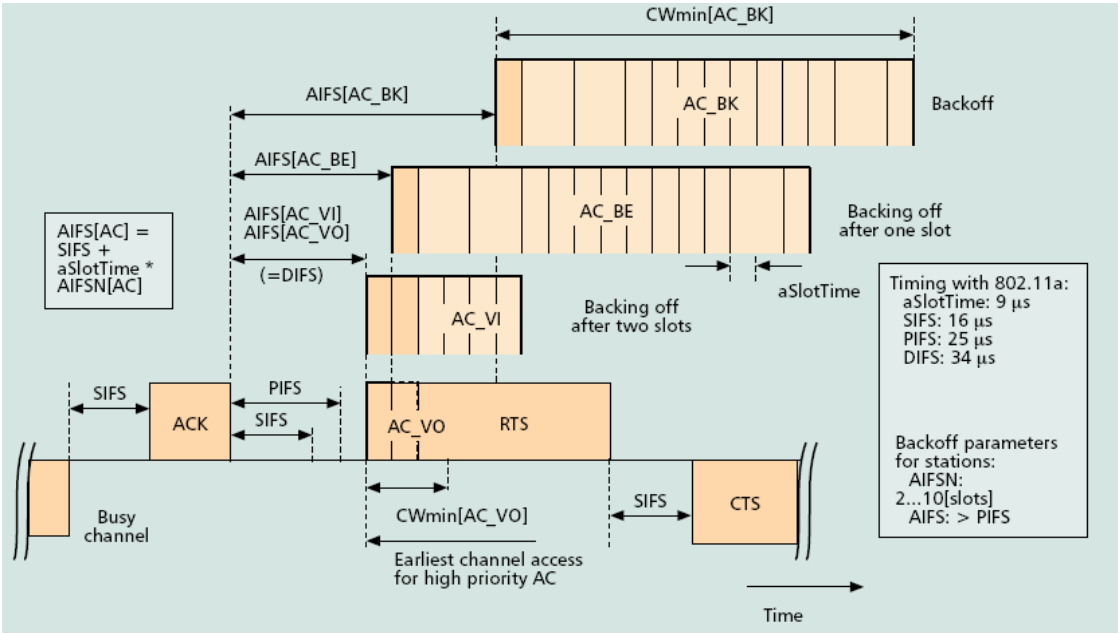

Figura 2. Ejemplo del uso de AIFS[AC] y CW para fijar prioridades específicas a cada AC[4].

[4] Tomado de [15].

Todas las QSTAs deben utilizar los mismos parámetros por AC, dichos parámetros son informados por el QAP en las tramas baliza.
Adicionalmente a los parámetros *AIFSN[AC]*, *CWmin[AC]* y *CWmax[AC]* se definen también Límites de duración de las TXOP (o TXOPLimit) por AC, un mayor valor de TXOP permite a la AC en cuestión mayor capacidad, puesto que puede transmitir más de un MSDU en cada TXOP.

2.2.1.2. Acceso al canal controlado por HCF (HCCA)

El mecanismo HCCA administra el acceso al canal inalámbrico usando un coordinador híbrido con mayor prioridad (de acceso al medio) que cualquier otra QSTA, permitiéndole transmitir MSDUS y asignar TXOP a las estaciones pertenecientes al QBSS.
El HC es de cierta forma un coordinador centralizado pero difiere del PC en muchos aspectos. La diferencia más importante es que el HC puede hacer uso del canal tanto en CP como en CFP. Otra diferencia significativa es que las TXOP asignadas a las QSTAs tiene una duración máxima especificada (TXOPLimit) y no pueden excederla.

- Procedimiento HCCA

El HC gana control del medio inalámbrico esperando un tiempo más corto entre transmisiones que las QSTAs que utilizan el procedimiento EDCA.
En la Figura 3 se ilustra un ejemplo de la relación existente entre los periodos de CP, CFP y CAP. Durante el CFP el HC puede entregar MSDUS y asignar TXOP a las QSTAs, sin embargo no es obligatorio que el HC genere CFPs puesto que también puede asignar el canal a las estaciones durante CP (dicho periodo se llama CAP).
Un CAP se genera cuando el HC necesita acceder al medio y las QSTAs están utilizando EDCA como mecanismo de acceso al medio (es decir, el QBSS esta en CP). El HC debe garantizar que ninguna TXOP vaya a retrasar la transmisión de las tramas baliza.

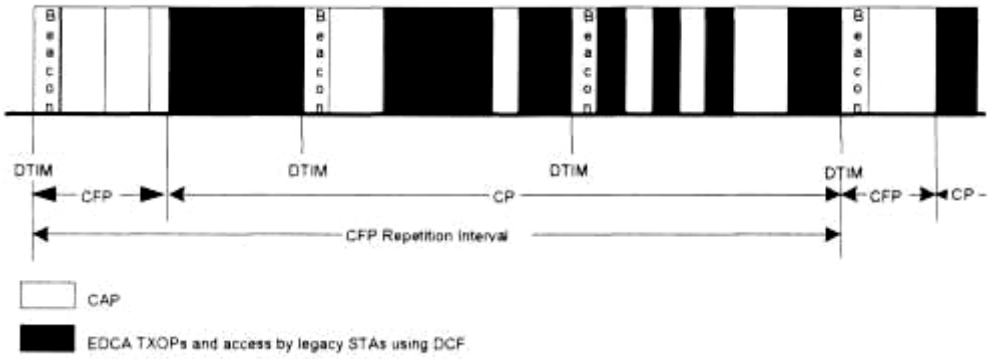

Figura 3. Ejemplo de la interacción entre periodos CFP, CP y CAP.[5]

Después de asegurarse de que el canal ha estado libre durante un PIFS el HC puede iniciar la transmisión de las tramas, asignando a la duración de la primera trama transmitida un valor suficiente para cubrir toda la TXOP, en la Figura 4 se observa una ilustración de dicho proceso.

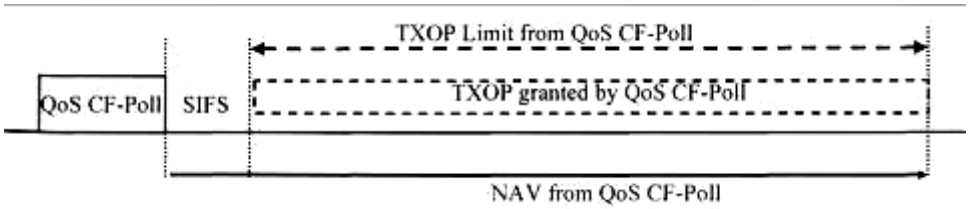

Figura 4. Protección de la duración de la TXOP a través del NAV[6]**.**

Las QSTAs deben reiniciar el valor del NAV (entrar en contienda) si reciben una trama QoS CF-POLL de duración cero, dicha trama es transmitida por ejemplo cuando el HC no tiene más QSTAs en lista para asignarles TXOPs ni MSDUs para ser entregados.

Las QSTAs pueden solicitar al HC la asignación de TXOPs utilizando el campo QoS Data, el cuál hace parte de todas las tramas de datos transmitidas en redes de tipo 802.11e. En dicho campo la QSTA informa al QAP la duración de TXOP solicitada así como la cadena de tráfico a la cuál se desea asignar dicha TXOP.

[5] Tomado de [6]. Página 77.
[6] Tomado de [6]. Página 79.

- Cadenas de Tráfico (TS)

Una cadena de tráfico describe un flujo de información de una fuente a un destino, dicho flujo es descrito especificando sus requerimientos de QoS. Cada QSTA puede tener como máximo ocho cadenas de tráfico activas (simultáneamente) en cada dirección (QSTA ► QAP y QAP ► QSTA). La fuente debe enviar al QAP la solicitud de creación de una TS (haciendo uso de la primitiva ADDTS descrita en [6]), para ello la QSTA debe construir un elemento TSPEC en el cuál incluya las características de la TS a crear, algunas de éstas son: tamaño nominal de MSDU, tamaño máximo de MSDU, Mínimo SI, Máximo SI, Tasa media de datos, entre otros.

La aceptación de la nueva TS depende de las condiciones actuales del QBSS (Número de TS actualmente activas, porcentaje del canal asignado a HCCA, estado del canal, entre otras).

Cuando una TS es aceptada es identificada dentro de la QSTA utilizando el TSID asignado (dicho número varia entre 1 y 8), mientras que el QAP la identifica utilizando adicionalmente la dirección de la QSTA que la solicitó. La transmisión de los datos correspondientes a la TS se hace utilizando HCCA a menos de que se habilite adicionalmente el uso de EDCA al momento de crear la TS.

En la Figura 5 se muestra el ciclo de vida de una cadena de tráfico. En el diagrama se observan los diferentes estados por los que puede atravesar una TS, como se puede ver es posible que una cadena de tráfico sea desactivada por solicitud de la QSTA que la creo (TS Deletion) o por inactividad (TS Timeout).

En la Figura 6 se muestra (de forma general) el intercambio de información que debe existir para crear una TS. Como se aprecia la estación origen envía una solicitud de creación de TS (ADDTS.request), dicha solicitud es transmitida y un contador (ADDTS Timer) es activado. La solicitud es analizada y respondida por el HC.

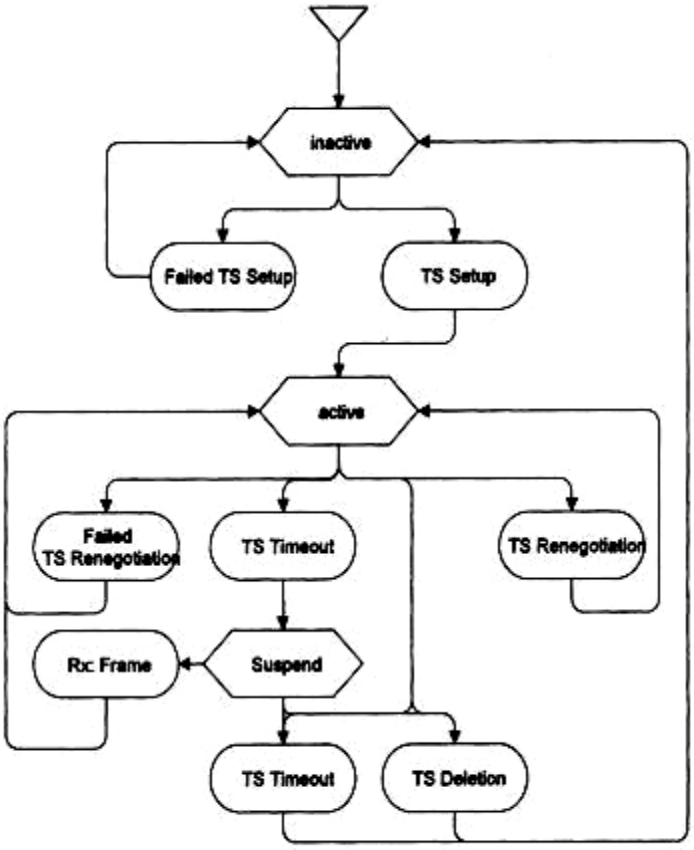

Figura 5. Ciclo de Vida de una Cadena de Tráfico.[7]

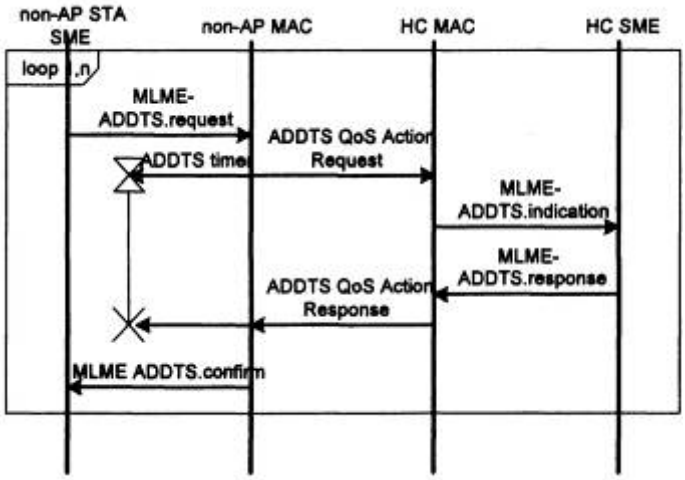

Figura 6. Configuración de una TS[8]

[7] Tomado de [6]. Página 133.
[8] Tomado de [6]. Página 134.

Ahora que ya se tiene un concepto claro de lo que es una cadena de tráfico se pueden discutir algunas características de HCCA adicionales.

- Recuperación de la ausencia de una recepción esperada:

Es importante aclarar que las reglas de recuperación en HCCA son muy diferentes de las usadas en EDCA, ya que en este modo de operación el NAV de las QSTAs es fijado para que no intenten transmitir antes del final de la TXOP.

El QAP debe retransmitir la última trama (después de un PIFS) si no detecta que la QSTA destino respondió(es decir, que el canal ha sido ocupado un SIFS después de la última transmisión). Lo mismo vale para una QSTA siempre y cuando la duración de la TXOP sea suficiente para completar el intercambio de tramas adecuadamente.

- Reglas de transferencia en HCCA:

Cuando el QAP asigna una TXOP a una estación específica, éste sugiere la TS hacia el cuál va dirigida la TXOP, sin embargo, la estación puede utilizar el medio para transmitir información de cualquier TS activa, esto se hace considerando que la información en la QSTA es más reciente (que la contenida en el QAP), por lo tanto la estación puede hacer una mejor elección para el uso del canal.

Cuando una estación transmite un mensaje debe informar al QAP (a través del campo "*QoS Control*") el tiempo necesario para transmitir el tráfico en cola correspondiente a la TS a la cuál éste pertenece, de tal forma que el QAP analice dicha información y la utilice para determinar la duración de la próxima TXOP.

- Control de Admisión en el HC:

El HC debe decidir (de acuerdo a las condiciones actuales de la red) si acepta o rechaza una solicitud de admisión de una nueva cadena de tráfico, sin embargo, si la TS es aceptada el HC debe garantizarle el tiempo (y la frecuencia) de acceso al medio inalámbrico necesario(s) para cumplir con los requerimientos de QoS especificados en el TSPEC correspondiente a dicha TS. Cómo se observó en la Figura 5, el HC puede cancelar una TS únicamente si permanece

demasiado tiempo inactiva (el valor límite para cada TS es especificado en el TSPEC).

2.2.2. Respuesta en Bloque (BA)

El mecanismo de respuesta en bloque permite transmitir un bloque de MSDUs separados por un SIFS. Fue diseñado para mejorar la eficiencia del canal al unir varias respuestas en una sola trama. Existen dos clases de respuestas en bloque: inmediata y retardada.

La estación fuente inicia el proceso enviando una trama llamada solicitud de adición de respuesta en bloque (ADDBA request), la estación receptora debe contestar dicha solicitud utilizando una trama de respuesta (ADDBA response), informando si la solicitud fue aceptada o no. Después de establecer la conexión es posible transmitir mensajes en bloque. El número de mensajes en bloque que se pueden transmitir es limitado, al igual que el número de confirmaciones que se le permite mantener a la estación receptora.

Es importante mencionar que el mecanismo de respuesta en bloque no requiere la configuración de una cadena de tráfico, sin embargo, las estaciones que tengan cadenas de tráfico activas pueden decidir (a través del programador) utilizar el mecanismo de respuesta en bloque para transmitir paquetes pertenecientes a dichas cadenas.

En la Figura 7 se ilustra la secuencia de mensajes necesaria para configurar, transmitir datos y terminar el mecanismo de respuesta en bloque.

2.2.3. Protocolo de Enlace Directo (DLP)

En general no se permite a las estaciones transmitir tramas directamente, siempre deben hacerlo a través del punto de acceso. DLP fue creado para eliminar esta limitación. Antes de que dos estaciones puedan hacer transferencias de información directamente es necesario ejecutar el intercambio de tramas DLP iniciales a través del QAP, esta restricción es impuesta ya que la estación receptora puede estar en modo de ahorro de energía y la única forma de despertarla es a través del QAP.

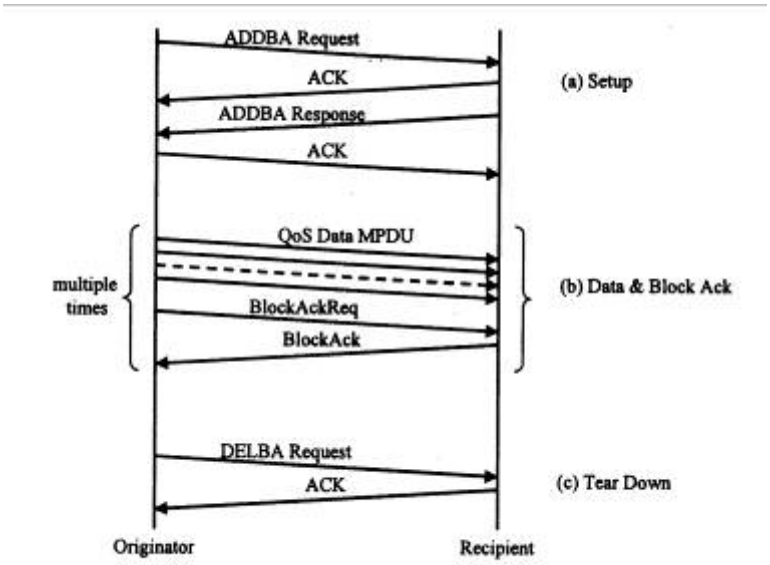

Figura 7. Secuencia de mensajes necesaria para: a) Configurar b) Transmitir y c) Terminar el mecanismo de respuesta en bloque.[9]

Como consecuencia de lo discutido anteriormente el mecanismo DLP impide a las estaciones entrar en modo de ahorro de energía mientras tengan alguna conexión DLP activa. En la Figura 8 se observan los pasos necesarios para establecer una conexión directa utilizando DLP.

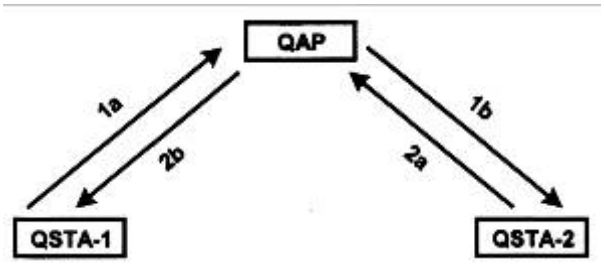

Figura 8. Pasos necesarios para establecer una conexión directa usando DLP.[10]

a) La estación 1 envía una trama de solicitud de creación de enlace directo (DLP request) a la Estación 2 a través del QAP.
b) El QAP verifica que la estación 2 tenga el mecanismo DLP implementado, de ser así reenvía la solicitud.
c) La estación 2 envía una respuesta a la solicitud (DLP response), dicha respuesta es enviada al QAP (pero esta dirigida a la estación 1).

[9] Tomado [6]. Página 87
[10] Tomado [6]. Página 143

d) El QAP reenvía la trama recibida a la estación 1. Después de que la estación 1 recibe dicha trama se puede iniciar la transmisión de información directamente entre las estaciones 1 y 2.

2.3. OMNeT++

A continuación se hace una breve descripción de algunas de las características más importante del simulador OMNeT++ (para obtener una descripción más detallada se recomienda consultar [13]), los temas discutidos son los siguientes: Descripción general, Lenguaje NED, Módulos Simples y Compuestos, Librería de Simulación, entre otros.

2.3.1. Descripción General

OMNeT++ es un simulador de redes orientado a objetos, basado en simulación por eventos discretos. El simulador puede ser utilizado para investigar en diferentes áreas del conocimiento entre las que se cuentan:

- Modelos de tráfico de redes de telecomunicaciones.
- Modelos de Protocolos
- Modelaje de colas
- Modelamiento de sistemas distribuidos
- Validación de arquitecturas de hardware
- Evaluación del desempeño de sistemas complejos
- Simulación de redes de comunicaciones.

Una de las mayores ventajas ofrecidas por el simulador es su uso libre para fines académicos.

Un modelo en OMNeT++ esta compuesto de módulos anidados (un (varios) módulo(s) dentro de otro(s)). La profundidad de los niveles jerárquicos construidos no tiene límites, esto permite al usuario implementar la estructura real del sistema a modelar. Los módulos se comunican a través de mensajes

que pueden contener cualquier tipo de estructura compleja. Es posible crear parámetros diferentes para cada módulo definido, dichos parámetros pueden ser variados en cada simulación, permitiendo hacer un análisis detallado de su influencia en el sistema.

El programa cuenta con diferentes ambientes de simulación disponibles para el usuario (dependiendo del propósito de la simulación), entre ellos están: ambiente para depurar el código, ambiente para demostración del funcionamiento del modelo, ambiente especial para realizar numerosas simulaciones modificando el valor de un parámetro y observar su influencia en el desempeño del sistema, otros.

Se pueden asignar tres propiedades diferentes a las conexiones: retardo de propagación, tasa de error de bit (BER), y tasa de transmisión.
Es posible programar el comportamiento de cada uno de los módulos definidos en el sistema, cada módulo es descrito por un archivo en C++, teniendo así disponible toda la capacidad de programación que brinda dicho lenguaje. Además es posible utilizar la librería de OMNeT++, dicha librería contiene clases que son muy útiles para la simulación de redes de comunicaciones, algunas de ellas son:

- Módulos (cModule)
- Mensajes (cMessage)
- Parámetros (cParameter)
- Colas, Arreglos (cQueue, cArray)
- Vectores para almacenar estadísticas (cVector)

El contenido de la librería de OMNeT++ es analizado en detalle en 2.3.4.
En la Figura 9 se muestra un ejemplo del ambiente de simulación que brinda OMNeT++ (en su ambiente gráfico).

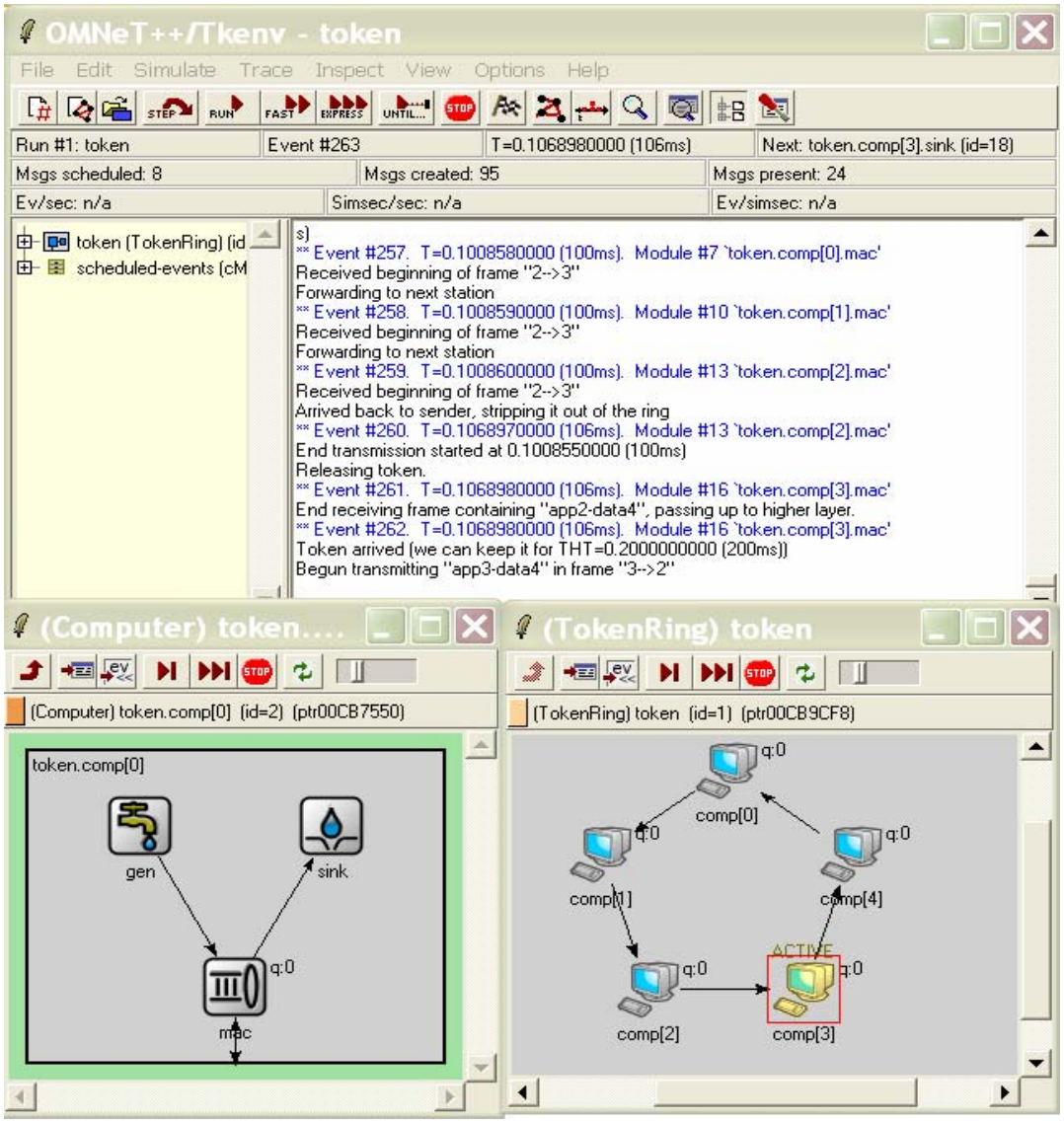

Figura 9. Ambiente de simulación gráfico de OMNeT++[11].

2.3.2. Lenguaje NED

La topología de un modelo se especifica haciendo uso del lenguaje NED, ya que éste facilita la descripción modular de una red. El lenguaje admite naturalmente la utilización de un modelo creado (por ejemplo un módulo simple o compuesto) en otro sistema.

[11] El ejemplo ejecutado es una simulación sencilla del protocolo *"Token Ring"* disponible en OMNeT++ V3.0

La descripción de un modelo en lenguaje NED puede contener los siguientes componentes:

- Lista de archivos a importar: Utilizada para incluir modelos declarados en otros archivos NED. La adición de un archivo NED funciona como la utilización de una librería en el lenguaje C++, permitiendo utilizar los modelos allí creados.
- Definición de canales: La definición de un canal crea una conexión con unas características específicas que puede ser usada una o varias veces para interconectar dos o más módulos.
- Definición de módulos simples y compuestos: En la Figura 10 se ilustran los posibles componentes de la definición tanto para un módulo simple como para un módulo compuesto.

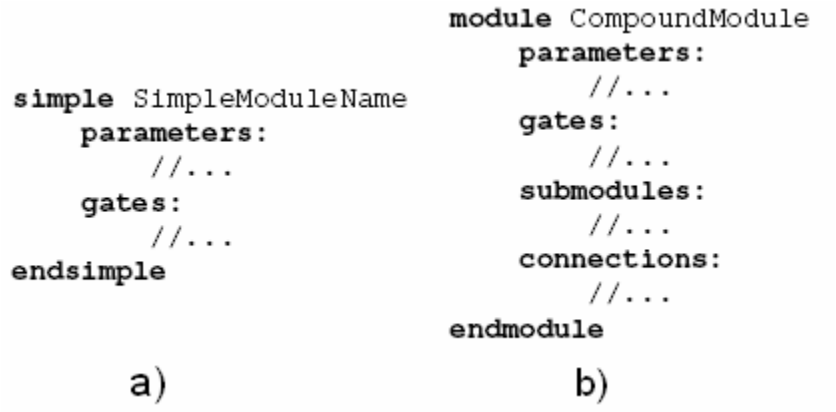

Figura 10. Descripción General de la estructura NED de un módulo: a) Simple y b) Compuesto.[12]

Como se puede observar en la figura anterior, existen dos componentes comunes a las dos clases de módulos: parámetros (los cuáles se utilizan por ejemplo para especificar el comportamiento de un módulo) y compuertas(o *gates*, las cuales permiten la salida o entrada de mensajes al módulo). Los módulos compuestos tienen dos componentes adicionales: submódulos (su función es adicionar módulos simples al módulo compuesto) y conexiones (cuya función es permitir el intercambio de mensajes entre submódulos).

- Definición de redes: La definición de una red ("*network"*) crea una instancia de los modelos definidos previamente.

[12] Tomado de [13]

Para finalizar la descripción del lenguaje NED, vale la pena mencionar que éste permite cierto nivel de programación, por ejemplo, es posible incluir ciclos en su código, utilizar las funciones disponibles en la librería "*math.h*", generar números aleatorios, entre otras; facilitando así la implementación de modelos que dependan de algún parámetro de entrada particular, igualmente se pueden crear conexiones y compuertas con características configurables a través de ciertos parámetros.

2.3.3. Librería de Simulación

OMNeT++ incluye una librería llamada "*omnetpp.h*". Dicha librería incluye diversas clases muy importantes y útiles para el desarrollo e implementación de diversos modelos en el simulador. A continuación se mencionan las clases más importantes, incluyendo una breve descripción se sus métodos más usados.

2.3.3.1. Clase cSimpleModule

El uso de esta clase es fundamental en el desarrollo de cualquier modelo ya que permite especificar detalladamente el comportamiento de los módulos simples en el sistema.

Algunos de sus métodos más importantes son:

- Método *Initialize()*: Invocado en la etapa de inicialización de la simulación, allí se deben establecer los valores iniciales de cada una de las variables a utilizar, así como asignar memoria a las estructuras declaradas.
- Método *HandleMessage()*: Es un método llamado durante el procesamiento de los eventos en el simulador. OMNeT++ recurre a este método cada vez que el evento actual es un mensaje dirigido hacia éste módulo.
- Método *Finish()*: Es ejecutado al terminar la simulación. Su principal función es almacenar (en archivos) las estadísticas recolectadas durante la simulación, para su posterior análisis.

Existe otro método disponible en la clase cSimpleModule, llamado *activity*, dicho método puede reemplazar al método *HandleMessage*, sin embargo, no se

recomienda utilizarlo ya que consume demasiada memoria y es eficiente únicamente cuando es usado para modelar sistemas muy particulares.

Otros métodos útiles disponibles en esta clase son:

- Método *ParentModule()*: Devuelve un apuntador al módulo padre del módulo actual. Útil para acceder a parámetros globales desconocidos por el módulo actual, pero conocidos por módulos de mayor jerarquía.
- Método *Send()*: Utilizado para enviar un mensaje hacia otro módulo a través de una compuerta de salida.
- Método *scheduleAt()*: Utilizado por el módulo para enviarse un auto-mensaje dentro de un tiempo determinado, particularmente útil para modelar contadores y temporizadores.

Para finalizar es importante mencionar que el nombre del módulo declarado en el archivo NED y la clase creada en lenguaje C++ debe ser el mismo. En la Figura 11 se ilustra un ejemplo del concepto anterior. Adicionalmente se puede observar el uso de la macro *Define_Module()*, dicha macro es usada para informarle a OMNeT++ que un nuevo módulo ha sido creado.

```
// file: swp.ned
simple SlidingWindowProtocol
    parameters:
        windowSize: numeric const;
    gates:
        in: fromNetw, fromHigherLayer;
        out: toNetw, toHigherLayer;
endsimple
```

a)

```
// module class declaration:
class SlidingWindowProtocol : public cSimpleModule
{
    Module_Class_Members(SlidingWindowProtocol,cSimpleModule,0)
    virtual void handleMessage(cMessage *msg);
};

// module type registration:
Define_Module( SlidingWindowProtocol );

// implementation of the module class:
void SlidingWindowProtocol::handleMessage(cMessage *msg)
{
   ...
}
```

b)

Figura 11. Ejemplo de uso de la clase *cSimpleModule* a) Archivo Ned b) Archivo C++.[13]

2.3.3.2. Clase cMessage

Esta clase es muy importante, ya que permite modelar la transmisión de paquetes entre módulos. Algunos de sus atributos más importantes son introducidos a continuación:

[13] Tomado de [13]

- *Name()*: Puede ser utilizado libremente por el usuario, con el fin de identificar el tipo de mensaje que esta llegando a un módulo particular. El nombre es almacenado en una cadena de caracteres y puede ser modificado a través del método *setName*.
- *Kind()*: Se utiliza para asignar al mensaje creado un tipo de mensaje especifico, desde el punto de vista del programador este atributo es muy útil en las estructuras *switch()*. El tipo de mensaje es almacenado en un entero y puede ser modificado a través del método *setKind*.
- *Length()*: Variable entera que contiene la longitud del mensaje (en bits). Este atributo es utilizado cuando el mensaje va a ser transmitido a través de un medio físico. La longitud del mensaje puede ser fijada usando el método *setLength()*.
- *hasBitError()*: Variable binaria (verdadera o falsa) utilizada para indicar que el mensaje sufrió errores durante la transmisión. El modelo de la capa física es el encargado de activar o no dicha variable utilizando el método *setBitError()*.
- *contextPointer()*: Apuntador que puede ser utilizado por el usuario para agregar información extra al mensaje enviado o identificar características adicionales del mensaje recibido. El método que permite utilizar dicho apuntador es *setContextPointer()*.

Existen otros atributos (de solo lectura) que permiten saber entre otras cosas: el tiempo de recepción y envío del mensaje, módulo fuente/destino y compuerta fuente/destino, entre otros.

Es posible también hacer una copia idéntica de un mensaje a través del método *dup()*.

La clase *cMessage* permite ser extendida fácilmente, es decir, se pueden crear nuevas clases derivadas de la clase *cMessage* con atributos y métodos adicionales creados por el usuario, según las necesidades específicas del modelo a desarrollar.

En la Figura 12 se observa un ejemplo de la construcción de una clase nueva llamada *MyPacket* derivada de la clase *cMessage*. El proceso es el siguiente: inicialmente debe crearse un archivo (con el nombre de la nueva clase y

extensión msg) que incluya los nuevos campos (Ver Figura 12a), luego se declara la clase y sus métodos (Ver Figura12b). Finalmente cuando se desee utilizar la clase creada debe incluirse el nombre de su archivo de encabezado (que en el caso del ejemplo es “mypacket_m.h”).

```
message MyPacket
{
    fields:
        int srcAddress;
        int destAddress;
        int hops = 32;
};
```

a)

```
class MyPacket : public cMessage {
    ...
    virtual int getSrcAddress() const;
    virtual void setSrcAddress(int srcAddress);
    ...
};
```

b)

Figura 12. Ejemplo de la construcción de una nueva clase a partir de la clase cMessage. a) Archivo MyPacket.msg b) Declaración de la clase MyPacket.[14]

2.3.3.3. Clase cQueue

Esta clase facilita la implementación de colas en los modelos a desarrollar, además permite almacenar la mayoría de las clases incluidas en la librería de OMNeT++.

En la Figura 13 se observa una ilustración de la estructura de dicha clase.

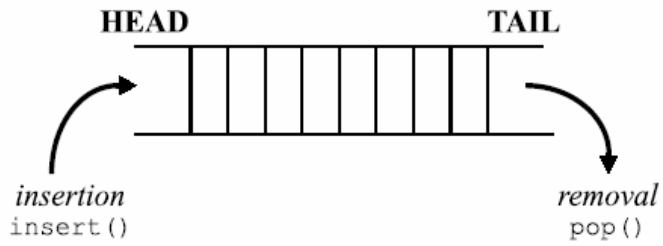

Figura 13. Estructura general de la clase cQueue.[15]

Los métodos más importantes de cQueue son:

- Método *Length()*: Devuelve el tamaño de la cola (número de estructuras en cola).
- Método *empty()*: Variable binaria (Verdadera o Falsa) que indica si la cola esta vacía o no.

[14] Tomado de [13]
[15] Tomado de [13]

- Método *tail()*: Devuelve un apuntador al primer objeto en la cola, sin afectar su contenido.
- Método *head()*:Devuelve un apuntador al último objeto en la cola, sin afectar su contenido.
- Método *pop()*: Extrae el primer objeto de la cola.

Utilizando los métodos descritos anteriormente es posible manipular la clase *cQueue* según las necesidades del modelo.

2.3.3.4. Clase cArray

Esta clase es utilizada para almacenamiento de objetos, funciona como un arreglo de C++ pero aumenta su tamaño automáticamente cuando se llena.

Algunos de sus métodos más usados son:

- Método *Add()*: Agrega un nuevo objeto al arreglo en la posición actual.
- Método *AddAt()*: Agrega un nuevo objeto al arreglo en la posición especificada.
- Método *find()*: Regresa el índice del arreglo en el cuál se encuentra almacenado el objeto.
- Método *clear()*: Vacía el arreglo.
- Método *items():* Devuelve el índice en el cuál se guardó el último objeto en el arreglo (que es el tamaño actual del arreglo si éste ha sido llenado secuencialmente).

Esta clase puede ser muy útil a la hora de almacenar objetos en estructuras creadas para modelar un sistema específico.

2.3.4. Compilando en Windows XP® un modelo desarrollado en OMNeT++.

Antes de discutir los pasos a seguir para compilar un modelo es importante hablar de las herramientas que tiene OMNeT++ disponibles para efectuar dicha compilación.

- Comando *nedtool:* se encarga de convertir los archivos descritos en lenguaje NED a archivos en C++, sin embargo, si el usuario lo necesita *nedtool* puede convertir los archivos NED a otros formatos como XML.
- Comando *opp_nmakemake*: se encarga de recolectar todos los archivos del modelo e incluirlos en un archivo llamado *Makefile.vc* el cuál será utilizado por *nmake* para crear el archivo ejecutable. Es importante mencionar que esta herramienta permite al usuario decidir si el programa va a ser ejecutado en ambiente de texto (cmdenv) o en ambiente gráfico (tkenv). Más adelante se muestra un ejemplo detallado aclarando el uso de este comando.
- Comando *nmake*: Es el encargado de hacer la compilación final, éste comando crea un archivo ejecutable. Si existe algún error en el modelo, *nmake* informa el nombre y la línea del archivo con error.

La figura 14 muestra los comandos que se deben ejecutar para generar un archivo ejecutable en modo texto (Figura 14 a) y en modo gráfico (Figura 14 b). El archivo que contiene todos los módulos del sistema se llama *nim.ned*. La opción –s es utilizada para especificar la extensión de los archivos que genera *nedtool* (que puede ser .cc o .cpp). La opción –f indica a los comandos *opp_nmakemake* y *nmake* sobrescribir los archivos previamente existentes.

```
nedtool -s _n.cpp nim.ned
opp_nmakemake -f -u cmdenv
nmake -f Makefile.VC
```

a)

```
nedtool -s _n.cpp nim.ned
opp_nmakemake -f
nmake -f Makefile.VC
```

b)

Figura 14. Ejemplo de comandos a ejecutar para crear un archivo ejecutable en a) ambiente texto y b) en ambiente gráfico.

2.3.5. Configurando y ejecutando las simulaciones

En la etapa de depuración de código y verificación del comportamiento del modelo se recomienda utilizar el ambiente gráfico, ya que éste permite hacer un

seguimiento detallado del estado y valores de las diferentes variables del modelo así como de la agenda de eventos.

Si después de verificar el correcto funcionamiento del modelo se desean ejecutar pruebas para diferentes valores de los parámetros del sistema, se recomienda utilizar el ambiente de texto (cmdenv), puesto que permite la ejecución simultánea de varias simulaciones y es mucho más veloz por que no tiene gráficos.

Toda simulación necesita un archivo de inicialización (generalmente *omnetpp.ini*) que especifique los valores de los diferentes parámetros del modelo. Este archivo tiene diferentes secciones: general, parámetros, cmdenv, tkenv y corridas. En cada sección se pueden especificar diversos parámetros. Para obtener mayores detalles al respecto se recomienda consultar [13].

3. MODELOS PLANTEADOS E IMPLEMENTACIÓN

Se implementaron dos modelos: Punto de acceso (QAP) y Estación (QSTA). En la Figura 15 se observa la arquitectura de los modelos implementados. Como se puede observar dichos modelos tienen algunos módulos en común (TrafficGen, CBR, VBR), sin embargo, difieren en la implementación de los módulos HCCA y ME (Entidad de Gestión). A continuación se describen las funciones de cada uno de los módulos implementados.

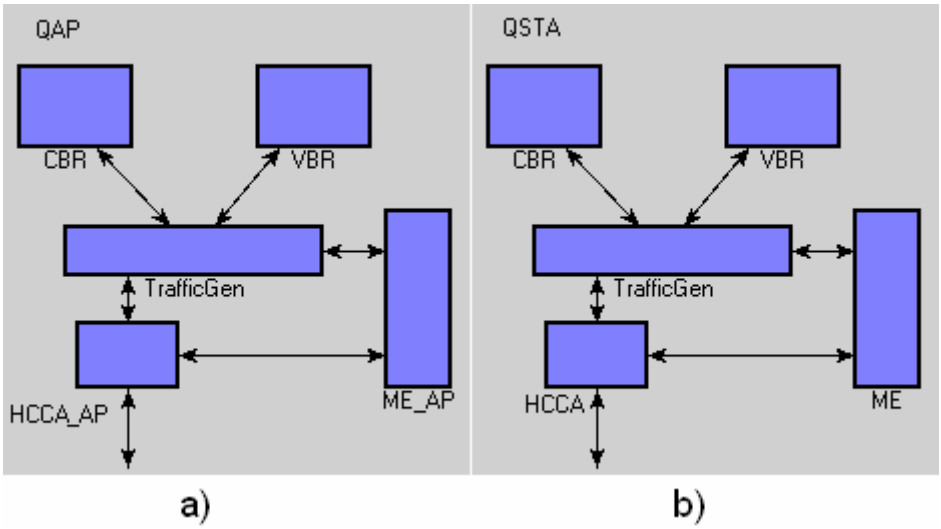

Figura 15. Arquitectura Modelos Implementados. a) Punto de Acceso (QAP). b) Estación (QSTA)

3.1. Módulo HCCA (QSTA)

Su función principal es garantizar que la estación haga un uso correcto del canal, es decir, que su comportamiento sea acorde al especificado en [6]. Algunas de sus funciones concretas son:

- Actualizar variables como el contador del mecanismo virtual de monitoreo del canal (NAV) y programar el tiempo de expiración de los espacios entre tramas (SIFS, PIFS).
- Transmitir las tramas de datos y gestión (que sean posibles) cada vez que el QAP le asigne el canal a la QSTA, así como verificar si hubo o no error en la

transmisión de las tramas de datos. Si hay un error ejecuta el algoritmo de recuperación de error descrito en [6].

- Actualizar el contador de retransmisiones cada vez que ocurra un error y eliminar la trama que esta siendo transmitida cuando dicha variable supere el límite establecido.
- Remover la trama (de la cola correspondiente) cuando esta ha sido recibida con éxito por la estación destino.
- Almacenar en la cola adecuada las tramas enviados por el generador de tráfico. De la misma forma debe guardar en una cola independiente los mensajes de gestión despachados por el módulo ME.

3.1.1. Programador implementado en la Estación

En la Figura 16 se muestra un diagrama de flujo que busca ilustrar el comportamiento del programador implementado en el modelo QSTA.

El macroalgoritmo ilustrado en la figura anterior puede ser analizado de la siguiente manera: Inicialmente la estación recibe una TXOP asignada por el QAP, enseguida se procede a verificar si hay algún paquete en la cola correspondiente a dicha TS, esto es posible puesto que el QAP indica (a través del campo TSID del elemento TSPEC) la TS hacia la cuál va dirigida la TXOP, si hay un paquete en cola y el tiempo asignado es suficiente, dicho paquete es transmitido, de lo contrario el programador ordena las TS activas, poniendo en primer lugar la cadena con mayor número de paquetes en cola.

Luego se ejecuta un ciclo buscando transmitir el primer paquete (en cola) posible da cada una de las cadenas de tráfico activas. Si no es posible transmitir ninguno de lo paquetes en cola (debido a restricciones de tiempo) se transmite una trama QoS Null para devolver al QAP el canal.

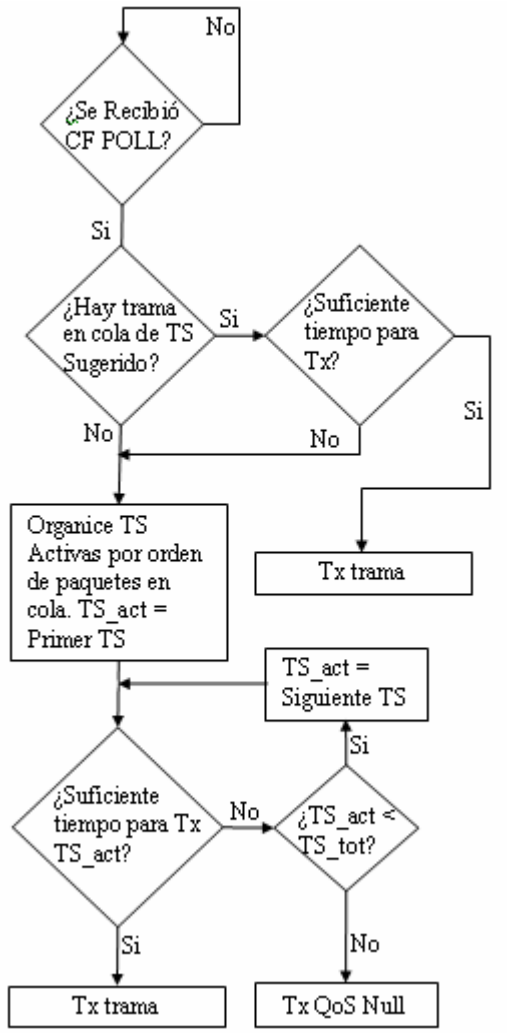

Figura 16. Diagrama de Flujo del Macroalgoritmo del programador implementado en la estación

3.2. Módulo Entidad de Gestión o ME (QSTA)

Este módulo se encarga de los temas relacionados con gestión de la red. Algunas de sus tareas específicas son:

- Construir las especificaciones de tráfico (TSPEC) para cada tipo de tráfico particular (como CBR, VBR por ejemplo).
- Construir las tramas de solicitud de adición (ADDTS) y de cancelación (DELTS) de una cadena de tráfico cuando sea necesario.

- Recibir e interpretar las respuestas (a las tramas de gestión) enviadas por la entidad equivalente en el QAP (módulo ME_AP)
- Mantener un listado actualizado de los TS activos y libres (de tal manera que pueda asignar apropiadamente un TS al recibir una solicitud nueva).
- Controlar que no se supere el número máximo de TS activos simultáneos permitidos para una QSTA (que según [6] es de 8).

3.3. Módulo HCCA_AP (QAP)

Este módulo cumple con las funciones del bloque HCCA (implementado en el modelo QSTA), pero además tiene otras responsabilidades adicionales. Su principal objetivo es administrar el uso del canal, por lo tanto (a través del algoritmo del programador) debe decidir a que QSTA y por cuánto tiempo se le asigna el medio inalámbrico. Otras labores adicionales ejecutadas por este bloque son:

- Transmitir las tramas baliza (BEACON).
- Asegurarse que el control del canal haya sido transferido adecuadamente (garantizando así el uso eficiente del canal).
- Recuperar el control del medio cuando una QSTA lo entrega antes de lo previsto y decidir a que estación se le asigna.
- Monitorear que ninguna QSTA use el canal mas tiempo del autorizado.
- Mantener un listado actualizado de la dirección de todas las QSTAs que hacen parte de la red (QBSS) así como del número de TS activos de cada una.

Los algoritmos utilizados tanto por el programador como por la unidad de control de admisión (del QAP) son los propuestos en el apéndice H de [6] y se describen brevemente a continuación:

3.3.1. Programador QAP

Como se dijo anteriormente el programador implementado en el modelo del punto de acceso es el propuesto en [6]. Es un programador sencillo que está lejos de ser el más eficiente, sin embargo, es una buena aproximación teniendo en cuenta que éste es el primer paso dado hacia la implementación de [6] usando OMNeT++.

El programador implementado ejecuta el siguiente procedimiento para cada una de las cadenas de tráfico activas:

Inicialmente se estima el número de paquetes generados por la cadena de tráfico (específica) durante un Intervalo de Servicio (SI). Dicha estimación se hace utilizando (3.3.1)

$$N_i = \left\lceil \frac{SI \times \rho_i}{L_i} \right\rceil \quad (3.3.1)$$

Con:

SI Intervalo de Servicio Actual.

ρ_i Tasa de Datos Promedio (de la cadena de tráfico en cuestión)

L_i Longitud promedio de los paquetes generados por la cadena de tráfico

Luego se utiliza (3.3.2) para decidir la duración de la Oportunidad de Transmisión asignada a dicha cadena de tráfico.

$$TXOP_i = \max\left\lceil \frac{N_i \times L_i}{R}, \frac{M}{R} \right\rceil + O \quad (3.3.2)$$

Con:

R Tasa de Transmisión

M Longitud Máxima permitida para un MSDU

O Incluye tiempo necesario para transmitir ACKs, SIFS, etc.

La ecuación (3.3.2) puede ser interpretada como: Escoger el valor máximo entre el tiempo necesario para transmitir los paquetes generados o transmitir un paquete de longitud máxima.

3.3.2. Unidad de Control de Admisión o ACU QAP

La unidad de control de admisión implementada es la propuesta en el apéndice H de [6]. Su función principal es decidir si se acepta o no una solicitud de adición de una cadena de tráfico. El criterio de decisión utilizado es la desigualdad (3.3.3).

$$\frac{TXOP_{K+1}}{SI} + \sum_{i=1}^{k} \frac{TXOP_i}{SI} \leq \frac{T - T_{CP}}{T} \quad (3.3.3)$$

La parte derecha de (3.3.3) representa el porcentaje de tiempo que el sistema funciona en HCCA. La parte izquierda es el porcentaje de tiempo del SI ocupado por las oportunidades de transmisión, si dicho porcentaje es menor que el porcentaje de funcionamiento del sistema en HCCA, la cadena de tráfico es aceptada, de lo contrario es rechazada. La figura 17 ilustra un ejemplo en el cuál hay tres cadenas de tráfico activas y el intervalo de servicio es de 50 ms.

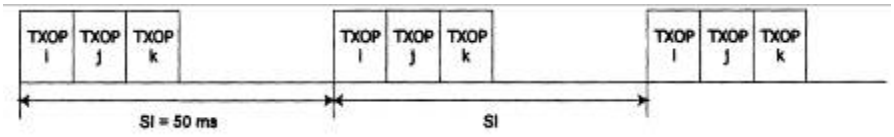

Figura 17. Ejemplo con tres cadenas de tráfico activas y SI = 50 ms.[16]

3.4. Módulo ME_AP (QAP)

Al igual que el bloque descrito anteriormente (HCCA_AP), el módulo ME_AP cumple con las funciones tanto de la Entidad de Gestión de una estación así como con otras funciones adicionales. Su tarea principal es responder a todas las tramas de gestión enviadas por las estaciones (QSTAs). Otras tareas a cargo de este bloque son:

- Implementar la unidad de control de admisión (ACU) que se encarga de decidir (basada en el estado actual de la red) si se puede aceptar (y por lo tanto garantizar QoS) a una nueva cadena de tráfico.

[16] Tomado de [6]. Página 169

- Calcular la duración de la oportunidad de transmisión de una cadena de tráfico específica, basándose en las especificaciones de tráfico suministradas por la estación.
- Definir la duración del intervalo de servicio (SI), dicha duración puede ser modifica cada vez que se acepta una nueva cadena de tráfico.

3.5. Módulo Generador de Tráfico (TrafficGen)

Su función principal es permitir la comunicación de uno o varios generadores de tráfico especializados (CBR o VBR por ejemplo) con los módulos HCCA / HCCA_AP y ME / ME_AP. La construcción de este módulo permite adicionar fácilmente al modelo cuántos generadores de tráfico se desee. Otra tarea importante desarrollada por este bloque es extraer la información contenida en clases específicas de cada generador y guardarla en clases que puedan ser interpretadas por los demás módulos del modelo.

3.6. Módulo Generador de Tasa de Bits Constante (CBR)

Este módulo genera paquetes de tráfico de tamaño constante a un intervalo de tiempo fijo. El tamaño del paquete, tiempo entre llegadas y número de mensajes a generar son parámetros del modelo y se pueden especificar en el archivo de inicialización de la simulación. Haciendo uso de este bloque se pueden modelar diversos tipos de tráfico, entre ellos: Audio, VoIP, tráfico de video de tasa constante, entre otros.

3.7. Módulo Generador de Tasa de Bits Variable (VBR)

Su función principal es generar paquetes de tamaño variable con tiempo de llegada entre paquetes también variable (la distribución de los dos parámetros mencionados anteriormente depende del tipo de tráfico que se desee modelar). Después de analizar varias alternativas de programación del módulo en

cuestión, se concluyó que la forma más amplia y flexible de implementación de éste era leer la información del tráfico a modelar de un archivo de texto. Esta decisión fue tomada por que se observó que en la mayoría de los estudios encaminados a modelar correctamente el tráfico VBR se obtiene como resultado uno o varios archivos de texto en los cuáles se especifican las trazas (tiempo de creación del paquete y tamaño) que caracterizan dicho tráfico. Con este bloque se puede modelar tráfico de video de tasa de bits variable principalmente.

3.8. Módulo Canal Inalámbrico (WM)

La implementación de éste módulo es muy sencilla y lo único que busca es interconectar las estaciones entre sí. Se modela únicamente el tiempo de transmisión de los paquetes.

4. Pruebas

En esta sección se describen las pruebas hechas al modelo implementado. Vale la pena aclarar que las pruebas desarrolladas simplemente buscan comprobar el correcto funcionamiento del modelo (no buscan evaluar el desempeño de [6]).Teniendo en cuenta dicho objetivo se hizo una investigación en artículos relacionados con el tema buscando experimentos practicados y resultados obtenidos en condiciones similares a las cuales se desea probar el modelo, con el fin de tener un punto de referencia a cerca de su funcionamiento.

Se realizaron dos tipos de experimentos: Prueba con tráfico VoIP y Prueba con Tráfico de Video. Las condiciones del sistema en cada una de ellas y los resultados obtenidos se describen en detalle en las secciones 4.1 y 4.2. En la Figura 18 se ilustra la arquitectura utilizada para realizar los experimentos, allí se pueden observar un punto de acceso (QAP) y un vector de estaciones (de tipo QSTA) de tamaño Num_Sta (éste es un parámetro configurable para cada simulación ejecutada utilizando el modelo).

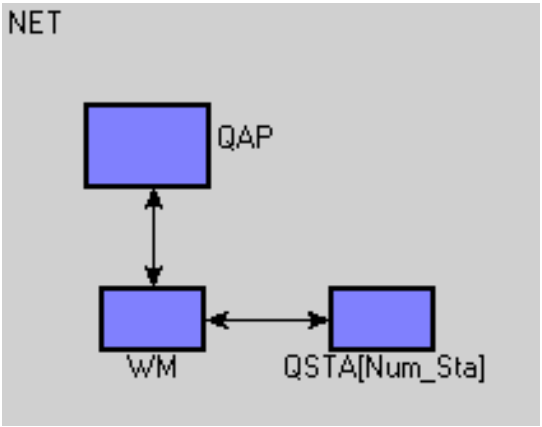

Figura 18. Arquitectura utilizada para realizar las pruebas de VoIP y Video

En la Tabla 2 se resumen los parámetros más importantes utilizados en los experimentos con tráfico de VoIP y Video.

Parámetro	Valor
Tasa de Transmisión Prueba VoIP [Mbps]	11
Tasa de Transmisión Prueba Video [Mbps]	24
Límite de Retransmisión	7
SIFS [μs]	10
PIFS [μs]	9
Intervalo de baliza [ms]	400
Máximo tamaño de MSDU [bytes]	2346

Tabla 2. Parámetros más importantes utilizados en las pruebas con tráfico VoIP y Video

4.1. Prueba con tráfico de VoIP

Para realizar la prueba con tráfico de VoIP se tomo como referencia el experimento realizado en [12]. Dicha prueba consiste en variar el número de conexiones VoIP y observar el retardo extremo a extremo promedio. Las propiedades del tráfico VoIP utilizado se muestran en la Tabla 3.

Parámetro TSPEC	VoIP
Tamaño de MSDU nominal [bytes]	60
Tamaño de MSDU máximo [bytes]	60
Intervalo de Servicio Mínimo [ms]	20
Intervalo de Servicio Máximo [ms]	20
Intervalo de Inactividad [s]	10
Tasa de datos mínima [kbps]	24
Tasa de datos promedio [kbps]	24
Tasa de datos pico [kbps]	24
Máximo tamaño de ráfaga [bytes]	60
Límite de Retardo [ms]	50
Tasa de Transmisión Mínima [Mbps]	11

Tabla 3. Propiedades del tráfico VoIP utilizado

En la Figura 19 se observa el retardo extremo a extremo experimentado en el modelo implementado y se compara con los resultados obtenidos en [12].

Se puede observar un comportamiento similar en los dos experimentos, sin embargo, el retardo promedio es siempre menor en el modelo implementado, esto se puede explicar teniendo en cuenta que en los experimentos practicados en [12] el canal tenía errores, mientras que en el modelo implementado el canal es libre de errores.

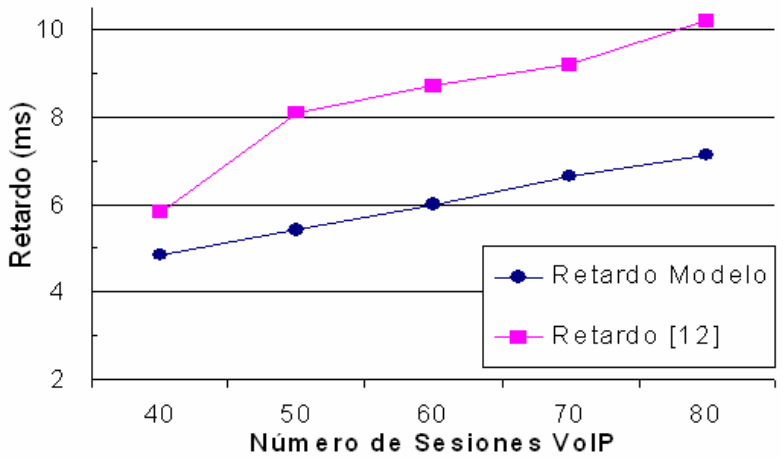

Figura 19. Comparación resultados obtenidos con tráfico de VoIP

4.2. Prueba con tráfico de Video

El artículo de referencia para esta prueba es [8]. El archivo para las trazas de video fue tomado de http://trace.eas.asu.edu/TRACE/ltvt.html. Para esta prueba se asumió un canal ideal. Los parámetros del tráfico de video se muestran en la Tabla 4.

Parámetro TSPEC	VIDEO
Tamaño de MSDU nominal [bytes]	320
Tamaño de MSDU máximo [bytes]	1917
Intervalo de Servicio Mínimo [ms]	0
Intervalo de Servicio Máximo [ms]	40
Tasa de datos promedio [kbps]	64
Tasa de datos pico [kbps]	383.4
Máximo tamaño de ráfaga [bytes]	1917
Límite de Retardo [ms]	40
Tasa de Transmisión Mínima [Mbps]	24

Tabla 4. Propiedades del tráfico de Video utilizado.

En este experimento se busca observar el retardo extremo a extremo de los paquetes de video a medida que varía el número de sesiones de video activas. En la Figura 20 se gráfica el retardo extremo a extremo promedio experimentado por los paquetes de video contra el número de sesiones activas. Se observa que el retardo del modelo implementado es más elevado que el observado en [8], esto se debe a que las trazas utilizadas no son exactamente las mismas trazas usadas en [8]. Adicionalmente se observa un comportamiento mucho más

uniforme en el retardo en el modelo. Dicha uniformidad se explica al tener en cuenta que el número de simulaciones realizadas para probar este modelo es mayor al utilizado en [8], por lo tanto el valor promedio del retardo en los experimentos con pocas sesiones de video es menos sensible a las trazas utilizadas en éste.

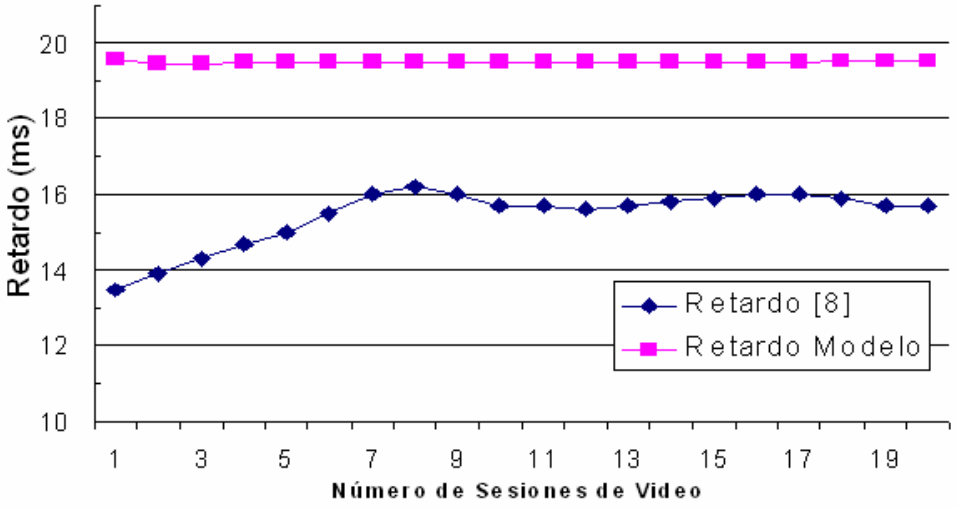

Figura 20. Comparación de resultados obtenidos con tráfico de Video

5. CONCLUSIONES

- El modelo desarrollado posee las características especificadas en [6]. Adicionalmente posee cualidades como: eficiencia, flexibilidad y fácil interacción con otros modelos.

- Los resultados obtenidos en las pruebas ejecutadas fueron significativos, puesto que, a pesar de no ser idénticos a los obtenidos en las investigaciones de referencia permitieron verificar el correcto funcionamiento de los módulos construidos.

- La herramienta de simulación utilizada es apropiada para implementar esta clase de modelos ya que sus características facilitan, entre otros, el modelaje de redes de comunicaciones.

- El programador y la unidad de control de admisión implementados no son eficientes, sin embargo, son un primer paso importante en la construcción del modelo.

6. TRABAJO A FUTURO

- Como se mencionó anteriormente el simulador OMNeT++ tiene un gran potencial. Es de vital importancia continuar con el desarrollo e implementación de [6] usando OMNeT++, de tal forma que en el futuro se pueda evaluar su desempeño y proponer algunas mejoras.

- Es necesario implementar nuevos algoritmos para el programador, puesto que el implementado actualmente no es lo suficientemente óptimo.

7. BIBLIOGRAFÍA

[1] IEEE Std 802.11, “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications”, 1999.

[2] IEEE 802.11a WG “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification: High Speed Physical Layer in the 5 GHz Band”, 1999.

[3] IEEE 802.11b WG “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification: Higher speed Physical Layer (PHY) extension in the 2.4 GHz band”, 1999.

[4] IEEE 802.11g WG “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification: Amendment 4: Higher speed Physical Layer (PHY) extension in the 2.4 GHz band”, 2003.

[5] IEEE 802.11i WG “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification: Amendment 6: Medium Access Control (MAC) Security Enhancements”, 2004.

[6] IEEE 802.11e/D 8.0 WG “Draft Supplement to Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification: Enhancements for Quality of Service (QoS)”, 2004.

[7] Y. Xiao, “IEEE 802.11e QOS Provisioning at the MAC Layer”, IEEE Wireless Communications, June, 2004.

[8] A. Salkintzis, G. Dimitriadis, D. Skyrianoglou, N. Passas, N. Pavlidou, “Seamless continuity of real-time video across UMTS and WLAN networks: challenges and performance evaluation”, IEEE Wireless Communications, June, 2005.

[9] S. Shankar, Z, Hu, M. van der Schaar, “Cross-layer optimized transmission of wavelet video over IEEE 802.11a/e WLANs”, in Proc. Packet Video 2004, Irvine, CA.

[10] S. Shankar, J, del Prado, P. Wienert, “Optimal packing of VoIP calls in an IEEE 802.11a/e WLAN in the presence of QoS constraints and channel errors”, in IEEE Global Telecommunications Conference, 2004

[11] P. Ansel, Q. Ni, T. Turletti, “An efficient scheduling scheme for IEEE 802.11e”, PLANETE Project, INRIA Sophia Antipolis, France. 2004.

[12] L. W. Lim, P. Y. Tan, C. Apichaichalermwongse, K. Ando, Y. Harada, “A QoS Scheduler for IEEE 802.11e WLANs”, IEEE CCNC, 2004.

[13] A. Varga, “OMNeT++**.** Discrete Event Simulation System. V3.0 User Manual”. http://www.omnetpp.org/.

[14] Crow B et al. “IEEE 802.11 Wireless Local Area Networks” IEEE Communications Magazine, Sept. 1997, pp 116-126.

[15] S. Mangold, S. Choi, G. Hiertz, O. Klein, B. Walke, ”Analysis of IEEE 802.11e for QoS Support in Wireless LANs,” IEEE Wireless Communications, Vol. 10, No. 6, December 2003.

[16] I. Meléndez, J. Chaves, “Desarrollo de la herramienta Qualnet para la simulación Y evaluación del desempeño de VoIP en una WLAN”. Tesis Ingeniería Eléctrica. Universidad de los Andes. 2004.

Anexo A – Conceptos Básicos y Abreviaciones

A continuación se presenta una breve descripción de algunos términos utilizados en [6].

- (AC) - Categoría de Acceso: Nombre asignado a un conjunto de parámetros EDCA usados por una QSTA para luchar por el canal cuando quiere transmitir un mensaje con ciertas prioridades.
- (CAP) - Fase de Acceso Controlado: Periodo de tiempo en el que el HC mantienen el control del medio, puede incluir TXOP asignadas a las QSTAs.
- (CFP) - Periodo Libre de Contienda: Periodo de tiempo durante el cuál el canal es asignado a las QSTAs únicamente por el coordinador híbrido, permitiéndoles intercambiar tramas sin necesidad de luchar por el canal.
- (CP) - Periodo de Contienda: Periodo de tiempo durante el cual las QSTAs deben luchar por el canal para poder transmitir información.
- (DL) - Enlace Directo: Enlace unidireccional de una QSTA a otra, que no pasa a través del QAP. Una vez es configurado un DL, el intercambio de información entre las dos QSTAs es directo.
- (EDCA) - Acceso al canal distribuido mejorado: Mecanismo de acceso CSMA/CA priorizado utilizado por las QSTAs en un QBSS.
- (HC) - Coordinador Híbrido: Tipo de coordinador que implementa las reglas de intercambio de tramas definidas por HCF. El HC opera tanto en el CP como en el CFP.
- (HCCA) - Acceso al canal controlado por HCF: Mecanismo de acceso al canal utilizado por el HC para coordinar el uso libre de contienda del medio.
- (HCF) - Función de coordinación Híbrida: Función de coordinación que combina y mejora las características de los métodos basados en contienda y libres de contienda buscando brindar a las QSTAs acceso priorizado y parametrizado al medio inalámbrico.
- (MSDU) – Unidad de datos de servicio de la capa MAC: Información que es entregada como una unidad entre puntos de servicio de acceso de la capa MAC.
- (NAV) – Vector de asignación del canal: Contador que indica a cada QSTA cuantas unidades de tiempo debe esperar para poder hacer uso del canal. Cuando una QSTA gana el derecho a usar el canal debe transmitir en sus tramas el nuevo valor del NAV, para que las otras QSTAs lo actualicen.
- (QAP) - Punto de Acceso de QoS: Punto de acceso que soporta las características de QoS descritas en [6].
- (QBSS) - Conjunto básico de servicio con QoS: Conjunto básico de servicio que ofrece las características de QoS descritas en [6].
- (QSTA) - Estación QoS: Estación que contiene las características descritas en [6].
- (SI) - Intervalo de Servicio: Tiempo transcurrido entre dos SP seguidos.
- (SP) - Periodo de Servicio: Periodo de tiempo durante el cual el QAP transmite una o varias tramas a las QSTAs y les asigna una o más TXOP.
- (TS) - Cadena de Tráfico: Flujo de datos a ser entregado cumpliendo con los requerimientos especificados en la TSPEC correspondiente.
- (TSPEC) - Especificación de Tráfico: Los requerimientos de QoS de un flujo de datos.

- (TXOP) - Oportunidad de Transmisión: Intervalo de tiempo en el cuál una QSTA tiene el derecho de iniciar una secuencia de intercambio de información usando el canala inalámbrico. Una TXOP tiene una duración máxima y puede ser obtenida luchando por el canal o puede ser asignada por el HC.

Anexo B – Código del Modelo Implementado

Para tener acceso al código completo del modelo implementado contactar a la Biblioteca General Universidad de los Andes