Desarrollo de mecanismos distribuidos para soporte a calidad de servicio en redes inalámbricas ad-hoc.

Realizado por:
Álvaro Torres Cortés

Dirigido por:
Carlos Miguel Tavares Calafate

13 de diciembre de 2010

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Las redes Wireless IEEE 802.11 [12] (Wifi) están disfrutando de una gran difusión en la sociedad actual gracias a su bajo coste, así como por la comodidad y versatilidad de uso que otorgan.

Una de las ventajas de esta tecnología frente a otro tipo de redes es su posibilidad de uso sin infraestructura, en un modo denominado ad-hoc.

Gracias a este modo de funcionamiento se puede formar de una una red de computadores funcional de manera fácil, rápida, y sin un coste económico extra asociado.

Normalmente estas redes sirven para el intercambio de información en áreas donde todos los nodos que participan de la red se encuentran en el mismo área de cobertura.

Actualmente se están haciendo esfuerzos en la investigación de las redes ad-hoc, que empleando protocolos de encaminamiento de paquetes consiguen una mayor flexibilidad, permitiendo que los paquetes pasen a través de diversas estaciones como si de Internet se tratase.

1.1. Motivación

Las redes móviles ad-hoc, también conocidas como MANETs, pueden resultar muy útiles en escenarios militares o de desastres naturales, en los cuales es necesaria la realización de comunicaciones sin ninguna infraestructura previa y que, además, requieren de cierto grado de movilidad. En este tipo de escenario se pueden realizar llamadas telefónicas (VoIP) o videoconferencias, situaciones dónde los usuarios implicados requieren unas garantías mínimas de QoS.

Dada la ausencia de una red ad-hoc real que sea capaz de ofrecer QoS de una forma efectiva, en este proyecto se hace un estudio de la ampliación de calidad de servicio de IEEE 802.11. Esto ha permitido realizar pruebas pioneras en este tipo

de redes en cuanto a rendimiento y retardo con QoS, que tan importantes son en VoIP y videoconferencia.

Se ha detectado además que las actuales MANETs no están dotadas de sistemas que permitan realizar el control de admisión para aplicaciones con requisitos de QoS para evitar saturar la red. Se ha considerado, pues, que esta carencia también debería ser subsanada de cara a lograr un soporte más robusto a la QoS.

1.2. Objetivos

Los objetivos de esta tesina son básicamente dos:

- En primer lugar se encuentra la realización, mediante diverso equipo informático, de un entorno flexible y controlable para la realización de experimentos. Este entorno debe ser una red Ad-hoc multisalto con soporte para calidad de servicio.

- El segundo objetivo de este proyecto es el desarrollo de un sistema de control de admisión que, haciendo uso del entorno de pruebas anteriormente configurado y de las características de QoS existente en él, sea capaz de ofrecer determinadas características de tiempo real y de prioridad a los flujos de datos que lo soliciten.

1.3. Estructura

Esta tesina se encuentra dividida en nueve capítulos: en el primero, que es en el que nos encontramos, se ofrece una visión general del trabajo que se ha realizado. En el segundo capítulo se realiza una introducción a las tecnologías empleadas, como pueden ser IEEE 802.11 y 802.11e; además, se ofrece una pequeña explicación sobre las redes MANET. En el capítulo 3 se describe como se implementó el entorno en el que se realizaron la primera parte de las pruebas, así como las soluciones encontradas a los diversos problemas que aparecieron al intentar crear la red y el entorno deseados. En el cuarto capítulo se detallan las diferentes plataformas empleadas para la realización de pruebas y recolección de datos que se han desarrollado o ampliado para conseguir las funcionalidades que deseamos. En el quinto capítulo se comprueba que el entorno de pruebas desarrollado funciona correctamente, y se comparan los resultados obtenidos en el entorno de pruebas real con unas simulaciones de entorno realizadas previamente mediante simulación. En el siguiente capítulo se describe un nuevo entorno de pruebas más realista y se comentan los resultados que de él se obtuvieron. En el capítulo 7 se describen algunos detalles de implementación del

sistema de control de admisión distribuido para redes MANET que se ha desarrollado, y se presentan diversos datos obtenidos al realizar los experimentos en nuestro entorno de pruebas. En el capítulo 8 se describen las interfaces de uso: una para programadores, otra (gráfica) para usuarios, y por último la que se ha implementado en la plataforma de realización de pruebas empleada. Finalmente, en el capítulo 9, se presentan las conclusiones del trabajo, así como las publicaciones resultantes de la labor realizada.

Capítulo 2

Soporte a QoS en MANETs

En este capítulo se explicarán qué son y cómo funcionan las redes IEEE 802.11. Tras esto se explicará la ampliación llevada a cabo en el estándar 802.11e que añade calidad de servicio (QoS) a este tipo de redes. Por último se explicará que son y cómo funcionan las redes MANET.

2.1. IEEE 802.11

Las redes que siguen el estándar IEEE 802.11 son las comúnmente conocidas como Wi-Fi, es decir, redes inalámbricas que permiten la comunicación entre dos o más nodos sin necesidad de cables. Están diseñados y pensados para su uso en redes inalámbricas de área local (WLANs).

Al ser un estándar de la rama de IEEE 802 es compatible con el nivel LLC definido en IEEE 802.2, por lo que es totalmente compatible con las redes locales tipo IEEE 802.3 (Ethernet); además, el funcionamiento de cara al usuario y sistema operativo es el mismo que en estas.

Este estándar se inició en 1997 con la aprobación de su primera versión en la que se describen las bases de la comunicación y el protocolo de detección y evasión de colisiones CSMA/CA (Carrier Sense Multiple Access / Collission Avoidance). En esta primera versión del estándar las velocidades alcanzadas eran de 1 o 2 Mbps, y habían tres opciones en cuanto a su nivel físico: infrarrojos, FHSS (*Frequency Hoping Spread Spectrum*) o DSSS (*Direct Sequence Spread Spectrum*). Estos dos últimos métodos ya utilizaban microondas en la frecuencia libre de 2'4 GHz. Este estándar nunca se llegó a implementar de manera comercial.

En 1999 se aprobaron las ampliaciones 802.11a y 802.11b que fueron las primeras implementaciones del estándar abiertas al gran público.

IEEE 802.11a Esta ampliación opera a una frecuencia de 5 GHz y permite velo-

cidades teóricas de entre 6 y 54 Mbps, aunque su rendimiento máximo real es de aproximadamente 20 Mbps. Debido a que la banda de frecuencias de 5 GHz no está tan colapsada como la usada por otras ampliaciones del estándar que usan la banda de 2'4 GHz (802.11b, 802.11g,...), no se ve afectada por tantas interferencias radioeléctricas y, por lo tanto, su rendimiento promedio es mejor. Sin embargo, como contras de la utilización de esta banda de frecuencias, podemos destacar que no tiene tanta penetración en los materiales sólidos como la de 2'4 GHz, por lo que se requiere una línea de visión más directa; además, alcanzar tanta frecuencia de transmisión desemboca en que el hardware necesario es también más caro y más complejo tecnológicamente, lo que inicialmente dificultó su implantación.
La modulación utilizada en este caso es la Multiplexación Ortogonal por División en Frecuencia (OFDM) con 52 frecuencias subportadoras.

IEEE 802.11b Al contrario que en la ampliación 802.11a, esta hace uso de la banda de frecuencias de 2'4 GHz lo que posibilita que los dispositivos tengan un menor coste para el usuario final. Dicha diferencia llevó a la adopción de esta parte del estándar por la gran mayoría de fabricantes y usuarios pese a que las velocidades ofrecidas (11 Mbps) son sustancialmente inferiores a las ofrecidas por 802.11a.
La modulación utilizada por 802.11b es diferente a la empleada en 802.11a aunque también utiliza técnicas de espectro ensanchado (DSSS). Esta ampliación introduce CCK (Complementary Code Keying) para alcanzar las velocidades de 5'5 y 11 Mbps.

Debido a la gran aceptación de las tecnologías de comunicación inalámbricas en 2003, se ratificó una tercera ampliación del estándar conocida como 802.11g [4].

IEEE 802.11g Es la tercera ampliación que se ha efectuado sobre el nivel físico y sus modulaciones en el estándar IEEE 802.11. En este caso se alcanzan velocidades teóricas de 54 Mbps (como en 802.11a) pero sobre la banda de 2'4 GHz (la misma que 802.11b). El aumento de velocidad es debido a la modificación de las modulaciones utilizadas para hacer uso de las mismas que en el protocolo 802.11a. Sin embargo, al utilizar la misma banda que la ampliación 802.11b, también se hace uso de la modulación original de 802.11b para proporcionar compatibilidad total con el anterior estándar.
Actualmente esta ampliación de IEEE 802.11 es la más extendida a nivel mundial, y es la que utilizaremos en todo el entorno de pruebas que se describirá más adelante.

IEEE 802.11n Es la última ampliación realizada al estándar IEEE 802.11 y con

ella se permiten alcanzar velocidades teóricas de hasta 600 Mbps tanto en bandas de 2'4 GHz como en las de 5 GHz.
Este aumento espectacular del ancho de banda se consigue por diversas mejoras:

1. Modificación del OFDM (orthogonal frequency-division multiplexing) de las ampliaciones a/g para conseguir una mejor eficiencia, pasando de 54 a 65 Mbps.
2. Reducción del intervalo de guarda entre símbolos (de 800ns a 400ns), con esto se consigue alcanzar los 72,2 Mbps.
3. Modificación del ancho de banda de la señal para llegar a ser de 40 MHz (también se puede mantener un ancho de banda de 20 MHz como en las anteriores ampliaciones del estándar). Gracias a que se dobla el espectro el ancho de banda que se consigue aumentar la velocidad de transmisión en la capa física a un poco más del doble que empleando un solo canal de 20 MHz, pasando a ser la velocidad de 150 Mbps.
   Esta ampliación del estándar es opcional y además provoca una gran cantidad de interferencias en la banda de 2'4 GHz ya que necesita dos canales que no se solapen.
4. Por último una de las ampliaciones más importantes en la inclusión de MIMO (Multiple-input multiple-output). Con MIMO se emplean diversas antenas para emitir y recibir datos pudiendo emitir diversos flujos de información a la vez y mejorando el SNR del receptor. Esta ampliación del estándar permite la utilización de hasta cuatro flujos, por lo que se multiplica por cuatro la capacidad del canal alcanzando los 600 Mbps. Sin embargo si empleamos un ancho de banda de 20 MHz esta velocidad queda reducida a unos 300 Mbps.

2.1.1. Formato de trama

2.1.1.1. Nivel físico

Aunque anteriormente se ha comentado la total compatibilidad con redes Ethernet el formato de la trama varía ligeramente para adaptarse al nuevo medio por el que circulan los datos.

La capa física (PHY) de IEEE 802.11 se separa en dos partes:

- PMD (Physical Medium Dependent)

- PLCP (Physical Layer Convergence Procedure)

La sub-capa PMD es la encargada de la transmisión de los bits en el canal, mientras que la sub-capa PLCP es la encargada de la sincronización con la capa MAC.

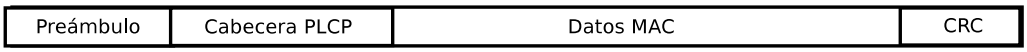

Figura 2.1: Frame PLCP IEEE 802.11

En la figura 2.1 podemos observar que PLCP añade su propio encabezado a la trama a transmitir. Este consiste en una secuencia de 80 bits que se usa para sincronización seguida de un delimitador de 16 bits llamado SFD (Start Frame Delimiter).

Tras el delimitador aparece la "Cabecera PLCP" que siempre es transmitida a 1 Mbit/s y contiene información que permite decodificar la trama en el nivel físico; estos campos son:

- Señal : Velocidad a la que se debería de transmitir.
- Longitud : Tamaño de la trama
- HEC (Header error check) : Es un CRC de 16 bits para detectar errores en la cabecera PLCP.

2.1.1.2. Nivel MAC

Al analizar en detalle una trama MAC de IEEE 802.11 como la que observamos en la figura 2.2, observamos que se encuentra dividida en nueve campos principales:

Frame control Este campo de 16 bits contiene gran cantidad de información como se puede observar en la figura X. Entre estos campos podemos encontrar información acerca de la versión del protocolo, el tipo y el subtipo de la trama, si la trama es un reenvío, y varios bits más con diferentes propósitos. Entre ellos destacan los bits "*To DS*" y "*From DS*" que identifican el modo de comunicación (entre estaciones, entre puntos de acceso, de una estación a un punto de acceso o de un punto de acceso a una estación).

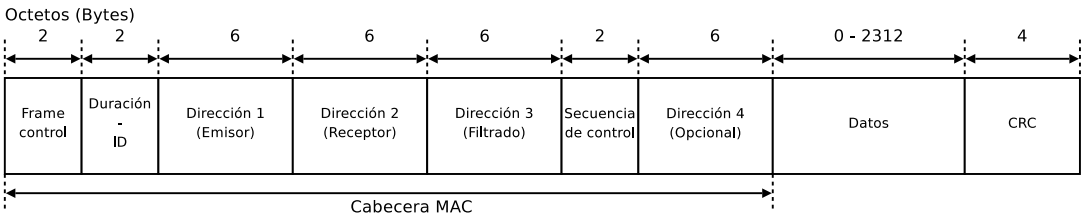

Figura 2.2: Formato de trama MAC de IEEE 802.11

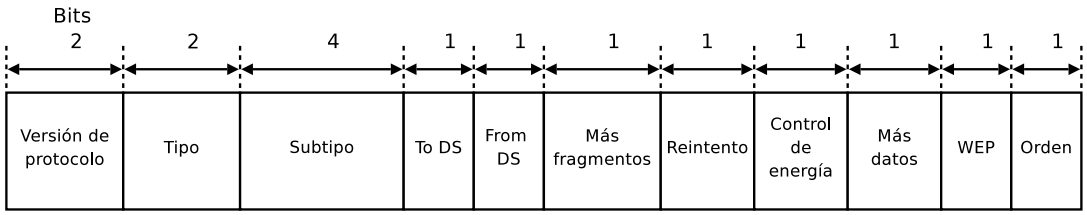

Figura 2.3: Información contenida en el campo "Secuencia de control"

Duración / ID Este campo depende si se está haciendo uso del modo de ahorro de energía o no, ya que puede contener o el identificador de la estación en el modo ahorro de energía o la duración usada para calcular el NAV *(Network Allocation Vector).*

Direcciones Se incluyen cuatro direcciones para permitir la mayor flexibilidad posible en el envío de paquetes. La dirección 1 siempre hace referencia al receptor de la trama en ese instante, así como la dirección 2 lo hace al transmisor. Las direcciones 3 y 4 identifican a la fuente y el destino de la trama en las situaciones en las que uno o más puntos de acceso son necesarios para la transmisión de una trama. En la tabla 2.1 podemos observar cómo se organizan las direcciones; en esta tabla DA (*Destination Address*) hace referencia al nodo final de la comunicación, SA (*Source Address*) se refiere al nodo que inició la comunicación, BSSID se refiere a la dirección del punto de acceso que se ve envuelto en la comunicación entre dos nodos. En el último caso, en el que se usan las cuatro direcciones, empleamos RA (*Receiver Address*) y TA (*Transmiter Address*) donde estos son el receptor y el transmisor de la trama (normalmente puntos de acceso).

Secuencia de control Este campo de 16 bits se usa tanto para fragmentar como para descartar tramas duplicadas. Se compone de un campo de 4 bits con el número de fragmento, seguido de un campo de 12 bits con el número de secuencia dentro de ese fragmento. Un detalle de los bits incluidos en este campo se puede ver en la figura 2.3.

2.1.2. Método de acceso al medio

Las estaciones pueden acceder al medio físico de dos formas diferentes, usando DCF (*Distributed Coordination Function*) o PCF (*Point Coordination Function*). Tanto una forma como la otra se engloban dentro del estándar CSMA/CA (*Carrier Sense Multiple Access / Collision Avoidance*) definido en la primera versión del estándar de IEEE 802.11.

Cuadro 2.1: Significado de los bits *To DS* y *From DS* así como el uso de las cuatro direcciones dependiendo de estos.

To DS	From DS	Explicación	Dir. 1	Dir. 2	Dir. 3	Dir. 4
0	0	Comunicación entre nodos dentro de un IBSS	DA	SA	BSSID	-
0	1	Envío de una trama desde un punto de acceso	DA	BSSID	SA	-
1	0	Envío de una trama a un punto de acceso	BSSID	SA	DA	-
1	1	Envío de una trama entre puntos de acceso	RA	TA	DA	SA

2.1.2.1. Distributed Coordination Function (CSMA/CA)

El mecanismo CSMA/CA se basa en un principio muy sencillo: escuchar antes de transmitir y mandar el mensaje cuando no se escuche nada. Debido a este principio de funcionamiento tan sencillo los mensajes se mandan sin garantías ni de llegada ni de ancho de banda o latencia.

Este mecanismo deriva de CSMA/CD (*Collision Detection*) empleado en Ethernet. La mayor diferencia entre estos dos mecanismos es que en el caso de CSMA/CA no se pueden detectar las colisiones ya que las tarjetas inalámbricas necesitan cierto tiempo para cambiar de emisión a recepción imposibilitando así la detección de colisiones. En este caso CSMA/CA intenta evitarlas empleando la siguiente metodología :

1. Escuchar el canal y esperar a que esté libre. (De ahí las dos primeras siglas *Carrier Sens*e)
2. Esperar un tiempo predeterminado (DIFS *Distributed InterFrame Space*) para confirmar que el canal sigue libre.
3. Esperar un tiempo de contención aleatorio para evitar las colisiones. (Ventana de contención o CW)
4. Transmitir.

Debido al punto 3 que es el que realmente consigue evitar la mayoría de colisiones, los nodos que eligen un tiempo de contención menor son los que consiguen transmitir antes que los demás. De todas maneras se verifica que en media, todos los nodos que participan de una comunicación tienen la misma probabilidad de transmitir su paquete.

Además, para conseguir evitar todavía más colisiones, las transmisiones se realizan siempre al principio de una ranura de tiempo.

Colisiones y pérdidas de paquetes

Naturalmente, el protocolo no es perfecto y, aunque se consigue evitar un gran número de colisiones, es posible que éstas ocurran. Por otra parte, el medio por el que se realiza la transmisión, el aire, es muchísimo más propenso a interferencias externas que un cable, lo que provoca altas tasas de pérdidas de paquetes. Para conseguir corregir esto, en IEEE 802.11 se hace uso del mecanismo denominado *Stop & Wait*, es decir, una vez mandado un mensaje se espera una confirmación de llegada; esta confirmación es una pequeña trama ACK que se envía tras un tiempo SIFS (*Short InterFrame Space*).

Si por cualquier razón la trama o el ACK se pierde, el nivel MAC se encarga de realizar el reenvío de la trama hasta un cierto número de veces. Además, y por si la pérdida de la trama ha sido debido a una colisión con otra estación que estaba emitiendo a la vez, el rango en el que se elige el tiempo de espera aumenta de una manera exponencial (hasta un máximo establecido "*CWmax*").

2.1.2.2. Point Coordination Function

PCF es el modo de acceso al medio que puede ser empleado cuando se requieren de ciertas restricciones temporales en ciertos servicios, como pueden ser la voz y el vídeo en un BSS (modo infraestructura).

Para el correcto funcionamiento de PCF es necesario que un nodo central PC (*Point Coordinator*), que normalmente es el punto de acceso, haga consultas a los demás nodos para priorizar unos paquetes frente a otros. Para ello hace uso de un tiempo de espera PIFS (*Point InterFrame Space*) para poder enviar sus tramas antes que los demás nodos.

Cuando se hace uso de PCF se alternan dos tipos de periodos CFP (*Contention-free periods*) y CP (*Contention periods*). Estos dos periodos juntos forman una supertrama.

Los periodos CFP comienzan normalmente tras la generación de un *Beacon o trama de control* (ya que permiten a los nodos sincronizarse). En estos periodos no hay competición por el acceso al medio, ya que es el PC el que dice a cada nodo cuando tiene que enviar su trama y por cuanto tiempo. Estos periodos finalizan con el envío de una trama "*CF-End*" que avisa a todos los demás nodos del inicio de un periodo CP.

Durante los periodos CP el acceso al medio es libre para todas las estaciones (de esta manera se evita la inanición por parte de algunos nodos), por lo tanto en los periodos CP todas las estaciones acceden al medio como se ha explicado en el punto 2.1.2.1.

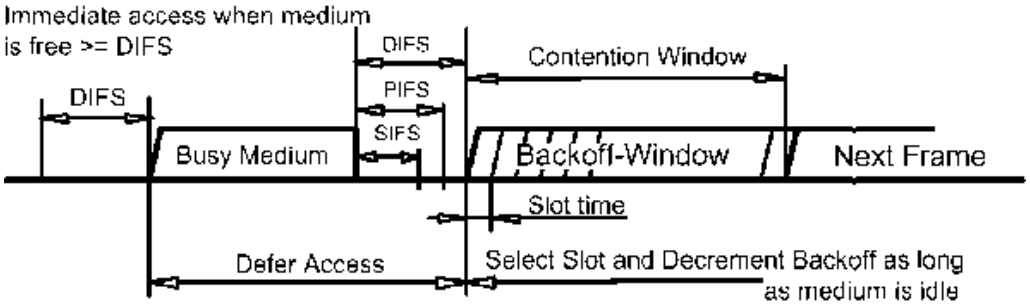

Figura 2.4: Detalle de CSMA/CA

2.1.3. Posibles estructuras de la red

Gracias a un acertado diseño de la trama IEEE 802.11 se permite una gran flexibilidad en el momento de crear la red y su topología. Existen tres modos diferentes por los que se puede conectar la red.

Basic Service Set (BSS) Es el comúnmente conocido como modo infraestructura. Para su formación se requiere que un nodo maestro (típicamente conocido como punto de acceso) establezca la red para que posteriormente los demás nodos se conecten a él. Debido a esta configuración todas las comunicaciones pasan por el AP (punto de acceso) indistintamente de la distancia existente entre los nodos que se comunican. Normalmente el AP tiene una conexión cableada con una red de mayores dimensiones (red de una empresa, internet,...).

Independent Basic Service Set (IBSS) También conocido como modo Ad-hoc, se suele emplear para conexiones de corta duración entre diferentes nodos. Como ventajas cabe destacar que no se requiere de ninguna infraestructura previa, es decir que simplemente con dos o más nodos que soporten IEEE 802.11 se puede crear una red Ad-hoc. Normalmente todos los que participan de esta red tienen que estar en el mismo radio de cobertura para poder enviar mensajes unos a otros; si no lo están, uno o más nodos deben de disponer de funciones de encaminamiento para que la comunicación se pueda llevar a cabo.

Extended Service Set (ESS) Es el más adecuado para un entorno empresarial. En un ESS se encuentran varios puntos de acceso conectados entre sí y cuyos rangos de cobertura se solapan parcialmente, por lo que se puede considerar como un conjunto de BSS. Gracias a que se permite asociación dinámica, y por lo tanto roaming entre puntos de acceso, se consigue que los usuarios de esta red dispongan de una gran movilidad manteniendo unos mínimos de calidad de señal y funcionamiento. En un ESS los AP deben de implementar técnicas de cooperación para intercambiar tramas; de esta manera los usuarios que

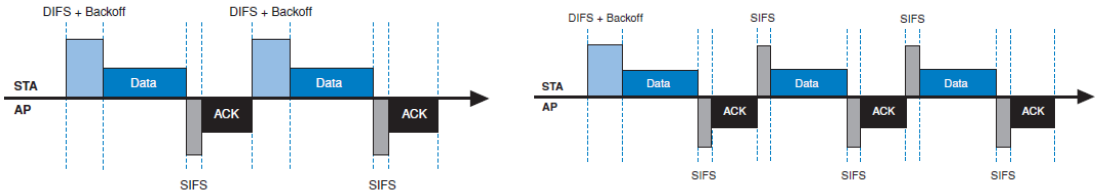

Figura 2.5: Envío de tramas normal (izquierda) y en ráfaga (derecha).

hacen uso de los servicios disponibles en la red disfrutan de ellos de manera transparente, como si todos formaran parte del mismo BSS.

2.1.4. IEEE 802.11e

En IEEE 802.11e ([13], [7] y [17]) se han centrado los esfuerzos en conseguir soporte nativo para Calidad de Servicio (QoS) en la MAC de IEEE 802.11. Para ello ha habido que modificar la MAC, así como modificar ligeramente los métodos de acceso al medio.

En esta nueva ampliación del estándar se define la *Hybrid Coordination Function* (HCF) que añade dos nuevos métodos de acceso al medio. El primero de ellos es *HCF Controlled Channel Access* (HCCA), que sustituye a PCF, y el segundo es *Enhaced Distributed Channel Access* (EDCA) [6], que sustituye a DCF.

Como en este proyecto solamente haremos uso de EDCA, y teniendo en cuenta que todavía no existen productos comerciales que soporten HCCA, no detallaremos su funcionamiento. Para más información sobre HCCA se pueden dirigir al estándar IEEE 802.11e [13].

En esta ampliación al estándar se definen cuatro nuevas categorías de tráfico que, ordenadas de menor a mayor prioridad, son las siguientes:

- Background (AC_BK)
- Best effort (AC_BE)
- Vídeo (AC_VI)
- Voice (AC_VO)

Según el estándar IEEE 802.11 original todos los paquetes pasan por la misma cola y tienen la misma prioridad. Sin embargo en IEEE 802.11e se definen cuatro colas distintas (una para cada categoría de acceso "AC"), de modo que los paquetes se tratan según su prioridad.

Para los nodos con soporte de IEEE 802.11 a y g los tamaños de la ventana de contención mínimos y máximos (CWmin y CWmax) está fijados a 15 y 1023,

Cuadro 2.2: Parámetros estándar de la MAC IEEE 802.11e

Categoría de acceso	AIFSN	CWmin	CWmax	Límite TXOP (ms)
Background	7	15	1023	0
Best Effort	3	15	1023	0
Vídeo	2	7	15	3008
Voz	2	3	7	1504

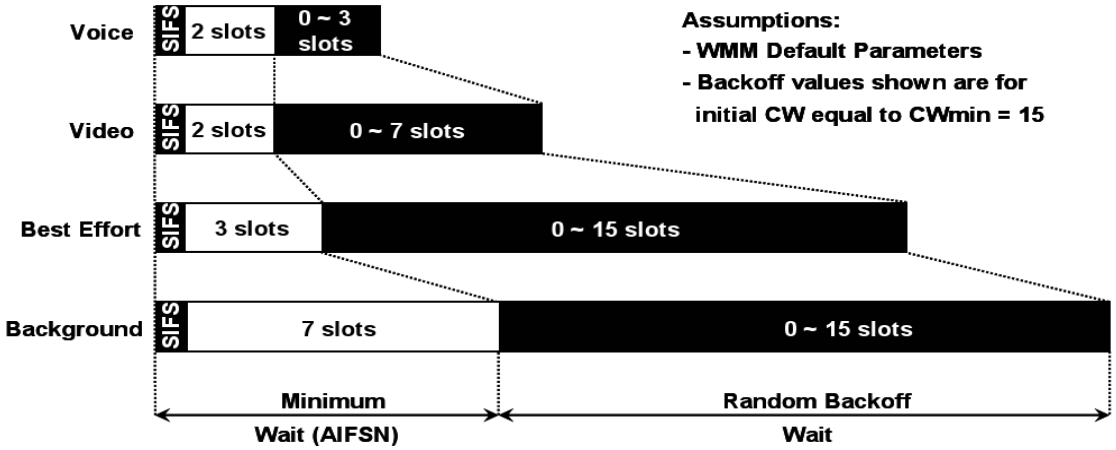

Figura 2.6: IFS, CWmin y CWmax para cada uno de las cuatro AC

respectivamente, y el tiempo de espera entre tramas (Inter-frame space) está fijado a DIFS. Cuando entra el juego la extensión de calidad de servicio esto cambia. Ahora cada categoría dispone de un IFS propio (AIFS[AC]), así como de un tamaño de ventana de contención propio (CWmin[AC], CWmax[AC]). Con esto se consigue una priorización del tráfico de las categorías de acceso con un menor tiempo entre tramas y unos tamaños de ventana menores.

En IEEE 802.11e, además de estas ampliaciones, se añade la posibilidad de un envío en ráfaga, es decir, un nodo puede enviar una gran cantidad de tramas seguidas esperando un tiempo SIFS tras recibir un ACK en vez de esperar AIFS[AC]; esto lo podemos observar más claramente en la figura 2.5. Naturalmente hay un límite máximo llamado oportunidad de transmisión (TXOP) que varía dependiendo de la categoría de acceso. Con la transmisión en ráfaga se consigue una mejora del rendimiento ya que no hay que hacer tantas esperas para comprobar si el canal está libre.

En la tabla 2.2 se pueden observar los valores estándar que toman estos parámetros, y en la figura 2.6 se puede observar gráficamente como el menor tamaño de estos parámetros desemboca en una mayor prioridad de acceso al medio.

Otra de las mejoras introducidas para mejorar el rendimiento y el retardo de las redes Wifi en IEEE 802.11e es la posibilidad de tratar de varias formas los ACK.

No ACK Aumenta la eficiencia al no mandar los paquetes ACK, esto es útil para

aplicaciones muy sensibles a altas latencias pero menos sensibles a las pérdidas de tramas, como puede ser VoIP.

ACK por bloques Se consigue aumentar la eficiencia al no tener que mandar un ACK para cada trama. Si un nodo envía una ráfaga de paquetes se puede contestar con un ACK de bloque aumentando así el rendimiento de la comunicación.

ACK normal Se envía un ACK para cada trama.

Comercialmente la inclusión de una parte de la extensión IEEE 802.11e en las tarjetas inalámbricas de los fabricantes se está publicitando como WMM (Wireless Multi-Media) o WME (Wireless Multimedia Extensions). La parte que deben de incluir los fabricantes para que su producto pueda optar a esta certificación es meramente la inclusión de EDCA en modo infraestructura.

En la última ampliación del estándar (IEEE 802.11n) se ha incluido obligatoriamente el soporte para IEEE 802.11e por lo que aparte de buscar el logotipo WMM todas las tarjetas y dispositivos inalámbricos que soportan el nuevo estándar también disponen de EDCA en modo infraestructura.

2.1.4.1. Diffserv [15]

Los parámetros de calidad de servicio están definidos normalmente por los programadores o los usuarios a nivel de aplicación. Dado que al hacer uso de un protocolo de nivel 3 (IP) o de nivel 4 (TCP) no pueden saber qué tipo de red se encargará de la transmisión del paquete existe el campo "Type Of Service" o "Traffic Class" tanto en IPv4 como en IPv6 que sirve para traducir las exigencias del usuario al nivel MAC como se puede observar en la tabla 2.3.

Cuadro 2.3: Traducción de prioridades a nivel usuario a categorías de acceso IEEE 802.11e

Prioridad de usuario	Nombre	Categoría de acceso	Nombre IEEE 802.11e
1	BK (Background)	AC_BK	Background
2	BK (Background)	AC_BK	Background
0	BE (Best Effort)	AC_BE	Best Effort
3	EE (Vídeo / Excellent-effort)	AC_BE	Best Effort
4	CL (Vídeo / Controlled Load)	AC_VI	Vídeo
5	VI (Vídeo)	AC_VI	Vídeo
6	VO (Voice)	AC_VO	Voice
7	NC (Network Control)	AC_VO	Voice

2.2. MANET

Una MANET (acrónimo en inglés de *Mobile Ad-hoc Network*) [16] y [11] es un conjunto heterogéneo y móvil de nodos agrupados formando una red ad-hoc.

En este tipo de redes no existe ningún nodo central que haga de encaminador de paquetes, es decir, que no requieren de ningún tipo de infraestructura.

Como consecuencia de no requerir infraestructura se consigue un alto nivel de flexibilidad. Entre sus puntos positivos podemos destacar que los nodos disponen de un alto grado de movilidad, auto reconfiguración y adaptabilidad a características variables de la comunicación (como puede ser potencia de transmisión, balanceo de carga...).

Por todas estas cuestiones las redes MANET están despertando un gran interés ya que su aplicación puede ser muy variada, para fines militares, por ejemplo, ofreciendo comunicación entre soldados y comandos militares en terreno enemigo; para fines civiles, dando soporte a: comunicación privada de los servicios de emergencias, extensión de internet en zonas rurales con difícil instalación de infraestructura, etc.

Sin embargo, debido a la alta movilidad de que disponen los nodos de este tipo de redes, se producen cambios frecuentes en la topología de la red, por lo que en contraposición a los anteriores puntos aparecen nuevos retos a superar. Así siendo, se deben desarrollar nuevos protocolos, nuevos algoritmos y nuevo middleware para conseguir que una MANET sea realmente una red descentralizada y auto configurable.

Uno de los primeros pasos para conseguir mejorar este tipo de redes es la calidad de servicio (QoS), que una vez integrada nos permitirá proveer de ciertas garantías de calidad a diversos flujos de datos que así lo requieran (vídeo, voz,...). En esta parte es en la que se centra este proyecto final de carrera, estudiando de una forma práctica los efectos de la activación de la calidad de servicio en una red Wifi ad-hoc multisalto. En este proyecto también se desarrolla una aplicación que, haciendo uso de las posibilidades que nos brinda esta activación, sea capaz de ofrecer ciertas garantías de tiempo real a los flujos de datos.

2.2.1. Protocolos de encaminamiento para MANET

El encaminamiento de paquetes en una red MANET es distinto al que se podría realizar en una red convencional. Un protocolo de encaminamiento para este tipo de redes debe de cumplir varias características para que sea útil.

La primera de ellas es que el protocolo debe ser distribuido, es decir, todos los nodos actúan como encaminadores, no habiendo ningún nodo central que se encargue de la toma de decisiones al enviar una trama.

La segunda de ellas es la complejidad del mismo. En un entorno en el que la mayoría de dispositivos funcionarán a base de baterías, el protocolo de encaminamiento no debe de suponer una carga para los dispositivos móviles ya que eso conllevaría una drástica reducción del tiempo de uso con baterías.

Existen también varias características más, como puede ser el soporte a la seguridad o la presencia de enlaces unidireccionales.

Es por ello que se han desarrollado diferentes algoritmos de encaminamiento específicos para esta plataforma siguiendo filosofías diferentes:

Protocolos de encaminamiento proactivos Este tipo de protocolos se basa en el intercambio periódico de información de encaminamiento, ya sea por tiempo o cuando se detecta algún cambio de topología. Con esto se consigue una vista relativamente completa de la red y, de esta manera, cada nodo calcula las distancias y las rutas disponibles a los demás nodos.

La principal ventaja de este tipo de protocolos es que cuando se requiere el envío de un mensaje a un nodo no hay retraso inicial debido a tener que calcular la ruta.

El protocolo más conocido de estas características es el OLSR [18] (Optimized Link State Routing).

Protocolos de encaminamiento reactivos Los protocolos reactivos no dependen de un intercambio periódico de mensajes para el cálculo de la ruta. Este tipo de protocolos se basan en encontrar la ruta hacia un nuevo nodo cuando se necesita. El método para conseguir esto es transmitir un paquete de *"Petición de ruta*" y esperar una respuesta antes de enviar paquetes a ese destino.

El punto fuerte de este tipo de protocolos es su bajísimo nivel de consumo de recursos ya que no se tiene que preocupar de mantener las rutas que no se usan.

El protocolo más conocido de estas características es el AODV [10] (*Ad-hoc On demand Distance Vector*).

2.2.1.1. OLSR

Dado que este es el algoritmo que vamos a emplear en las diferentes pruebas que se comentarán más adelante pasaré a detallar su funcionamiento.

Tal y como se ha comentado anteriormente OLSR es el máximo exponente de los algoritmos de encaminamiento proactivos es decir, son los algoritmos que periódicamente envían mensajes para mantener una ruta actualizada hacia todos los nodos de la red.

Para ello hace uso de un mecanismo de "flooding" denominado MPR (MultiPoint Relay). Este mecanismo se basa en la selección de ciertos nodos para poder alcanzar

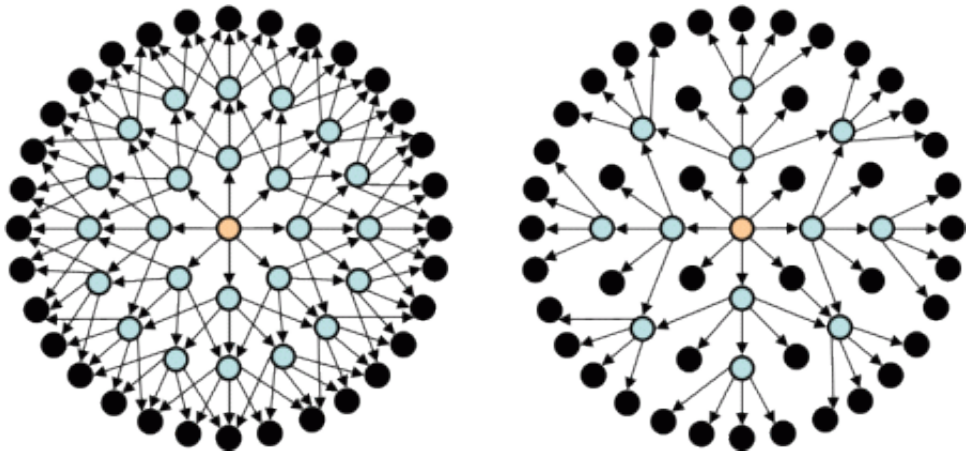

Figura 2.7: MPR: Difusión de los TC en OLSR

a todos nuestros vecinos a uno y a dos saltos al menos una vez. Los MPR de un determinado nodo se eligen procesando la información que nos llegan de los mensajes de control. Un ejemplo de cómo funciona este mecanismo se puede encontrar en la figura 2.7.

OLSR dispone de dos tipos de mensajes:

HELLO Son enviados periódicamente por cada nodo de la red a sus nodos vecinos, pero nunca son retransmitidos más allá del primer salto (1 hop) desde su emisor (alcance local). Estos mensajes contienen la lista de vecinos conocidos por el nodo emisor así como la identidad de los multipoint relays seleccionados por transmisor. Su intercambio permite a cada nodo de la red conocer los nodos situados a 1 y 2 saltos de distancia (es decir, aquellos a los que se puede hacer llegar un mensaje con una transmisión directa o con una transmisión y una retransmisión) y saber si ha sido seleccionado como MPR por alguno de sus vecinos.

Traffic Control (TC) Son enviados periódicamente y de forma asíncrona. A través de ellos, los nodos informan al conjunto de la red acerca de su topología cercana. Al contrario que los HELLO, los mensajes TC son de alcance global y deben llegar a todos los nodos de la red. El conjunto de los mensajes TC recibidos por un nodo inalámbrico le permite reconstruir su base de datos topológica, computar el árbol de caminos mínimos (mediante el algoritmo de Dijkstra) y calcular así la tabla de enrutamiento hacia todas las posibles destinaciones. La diseminación de mensajes TC se hace de acuerdo con el mecanismo de flooding basado en MPR.

Capítulo 3

Desarrollo de un testbed IEEE 802.11e

En este capítulo se describirá el entorno en el que se han realizado todas las pruebas relativas al proyecto.

El capítulo se divide en tres secciones diferenciadas. En la primera de ellas se explicará cual ha sido el hardware empleado para la realización de las pruebas, así como los problemas que conllevó usar este hardware y las soluciones encontradas para los problemas detectados.

La segunda sección trata sobre la topología lógica de la red que queremos adoptar para la realización de las pruebas, y cómo se ha conseguido definir esta topología mediante la utilización de herramientas software.

Por último se describirá el software que se ha creado para la automatización de las pruebas.

3.1. Detalles relativos al hardware

El hardware empleado para las pruebas ha sido muy variado. Para conseguir crear una red multisalto con las características requeridas dispusimos de ocho equipos, seis de los cuales son considerados netbooks de bajas prestaciones, y los otros dos ordenadores restantes son un portátil de gama media-alta y un sobremesa de gama media, por lo que el entorno empleado para las pruebas puede ser considerado representativo de lo que se puede esperar en cuanto a potencia de los dispositivos de una red MANET.

A continuación se listan específicamente los modelos empleados.

- 5 Asus EEE PC 1000H
- 1 Asus EEE PC 901
- 1 HP Compaq 6720s

- 1 Ordenador de sobremesa genérico.

Otra parte importante de nuestro entorno de pruebas son las tarjetas de red inalámbricas, ya que son la parte fundamental de nuestro estudio. Se utilizaron varios tipos de tarjetas inalámbricas distintas, donde todos los modelos compartían la característica de utilizar un chipset Ralink.

La elección de estas tarjetas con chipset de este fabricante en concreto está directamente relacionado con la liberación de los drivers como código GPL, ya que fue necesaria su modificación para conseguir el funcionamiento del protocolo IEEE 802.11e en modo Ad-hoc.

Los chipsets en cuestión son los siguientes:

- RT2860 - Tarjeta inalámbrica integrada en los 6 Asus EEE PC
- RT2870 - Tarjeta inalámbrica con interfaz USB disponible comercialmente como "Cisco Linksys WUSB600N-EU" y "D-Link DWA-140"

3.1.1. Drivers

Como se ha comentado anteriormente, la disponibilidad del código fuente de los drivers de las tarjetas de red inalámbricas es indispensable para poder realizar las modificaciones pertinentes, permitiendo así realizar las pruebas necesarias para el estudio que deseamos llevar a cabo.

Los drivers empleados para las pruebas son los siguientes:

- **RT2860** Versión 1.7.0.0
- **RT2870** Versión 1.3.1.0

Estos drivers eran los últimos disponibles para cada una de las tarjetas en la fecha de inicio del proyecto [2].

3.1.1.1. Activación de IEEE 802.11e en modo ad-hoc.

Para conseguir que el protocolo de QoS en redes Wifi se activase al funcionar en modo ad-hoc se podían seguir dos líneas de actuación distintas:

1. Modificar los beacons para anunciar extensiones WMM.
2. Asumir que recibimos extensiones WMM en los beacons.

La solución más limpia para solucionar el problema es la contemplada en el estándar: anunciar en los beacons la compatibilidad con WMM. Sin embargo, debido a que la formación de los beacons también se realiza por firmware (además de en los drivers),

se optó por la segunda solución que, aunque no sea estándar, nos sirve para alcanzar nuestro propósito, es decir, la activación incondicional del protocolo IEEE 802.11e en modo ad-hoc.

3.1.1.2. Corrección de la velocidad en modo ad-hoc.

Los drivers empleados, además de las modificaciones comentadas anteriormente, tuvieron que ser modificados para conseguir velocidades de 54 Mbits en modo ad-hoc cuando se conectaban más de cinco nodos.

Este fallo venia provocado por que, al realizar la asociación de esos nodos, las tarjetas disminuían su velocidad de transmisión de 54 a 11 Mbps. Esto se debe a que en el proceso de unión a la red del sexto nodo, los demás terminales detectaban esa velocidad independientemente de cual fuese su velocidad real soportada y, para mantener compatibilidad, todas descienden automáticamente a la menor velocidad común (en este caso ,11 Mbits).

Este problema se solucionó modificando el driver para que siempre detectase que los nodos conectados funcionan a 54 Mbits.

3.1.1.3. Compatibilización con nuevas versiones del Kernel de Linux.

Debido a que a partir de la versión 2.6.27 del Kernel de Linux se modificaron diversas llamadas al sistema, los drivers empleados dejaron de ser compatibles con éste. Por esta razón se procedió a modificar el driver de las tarjetas de red para conseguir compatibilidad con las últimas versiones del núcleo de Linux.

Actualmente los drivers han sido probados y funcionan para versiones del núcleo desde la 2.6.24 hasta 2.6.28. En la versión 2.6.29 se han vuelto a modificar diversas llamadas al sistema que imposibilitan el correcto funcionamiento de los drivers modificados con esta nueva versión del núcleo.

3.2. Estructura lógica

Dado que queremos conseguir un entorno de pruebas multisalto como el de la figura 3.1 y no disponemos del suficiente espacio para conseguir que el radio de cobertura de las tarjetas inalámbricas no se solape, tuvimos que usar las órdenes *iptables* [3] y *route* disponibles para el sistema operativo GNU/Linux.

El primer paso que hay que realizar para conseguir una red multisalto funcional es que los nodos que participan de la comunicación puedan encaminar paquetes (por defecto desactivado en las distribuciones de GNU/Linux); para conseguir el comportamiento correcto hay que activar explícitamente esta característica.

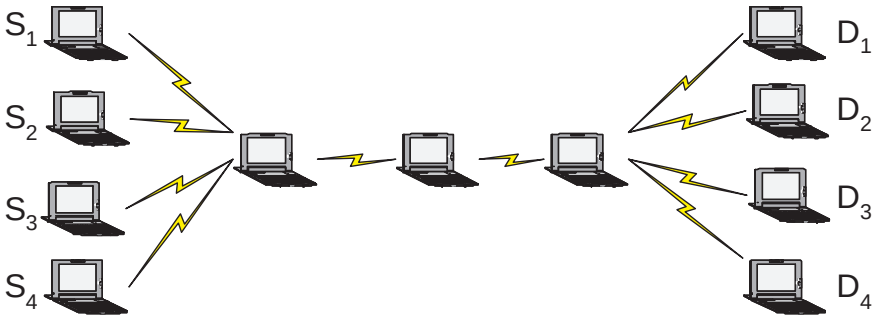

Figura 3.1: Estructura lógica de la red.

```
1. echo 1 > /proc/sys/net/ipv4/ip_forward
```

Figura 3.2: Activación del encaminamiento de paquetes en GNU/Linux

La orden necesaria para la ejecución se puede ver en la figura 3.2, y tiene que ser ejecutada como *"root"*. No se puede hacer uso del comando *"sudo"* ya que el funcionamiento no es el deseado.

3.2.1. Iptables

Iptables es una herramienta muy potente que nos permite el filtrado de paquetes de una manera sencilla y eficiente.

Para conseguir el entorno multisalto tuvimos que realizar un filtrado por MAC para impedir que los paquetes que se envían sean recibidos por algún nodo no deseado. Para ello utilizamos las órdenes que podemos ver en la figura 3.3.

En cada nodo se filtraron todos los nodos que, según la estructura, no se debían ver entre si. Además, para evitar un funcionamiento incorrecto, también hay que filtrar la posibilidad de *"Forward"* (encaminamiento) para las MAC censuradas anteriormente.

Aparte del filtrado MAC, hubo que filtrar también los paquetes de ICMP Redirect que nos llegan de los diversos nodos del camino ya que impedían el correcto funcionamiento de la red. La orden para realizar el filtrado es la indicada en la figura 3.4.

Tras conseguir que los nodos viesen la red como si de una MANET se tratase, y para conseguir la medición de los retardos de los paquetes así como la monitorización

```
1. iptables -A INPUT -m mac –mac-source "MAC" -j DROP
2. iptables -A FORWARD -m mac –mac-source "MAC" -j DROP
```

Figura 3.3: Ordenes para el filtrado de paquetes a nivel MAC

1. iptables -A INPUT -p icmp –icmp-type 5 -j DROP
2. iptables -A OUTPUT -p icmp –icmp-type 5 -j DROP

Figura 3.4: Filtrado de ICMP Redirect

- Primero: Todos los paquetes que salen de una máquina llevan su IP

iptables -t nat -A POSTROUTING -p udp -j SNAT –to-source 192.168.2.100

- segundo: Los paquetes que se reciben por cada puerto concreto se redirigen a la máquina origen.

iptables -t nat -A PREROUTING -p udp –dport 64000 -j DNAT –to-destination 192.168.2.101

Figura 3.5: Redirección de los paquetes a las máquinas origen

y lanzamiento de las instrucciones remotas, se utilizó una red secundaria Ethernet cableada de 100Mbps.

Para conseguir que el tráfico sea encaminado correctamente se utilizaron las instrucciones de iptables que podemos ver en la figura 3.5.

La primera de ellas es para que todos los paquetes que enviemos nosotros sean por el rango de IPs de la red Ethernet.

La segunda de ellas se encarga de analizar el tráfico que recibimos y redirecciona a la máquina origen utilizando su dirección de red cableada.

Para conseguir que este proceso fuese automático las direcciones IP utilizadas en todas las máquinas son conocidas y fijas. En nuestro caso las IP utilizadas en las conexiones inalámbricas son del estilo 192.168.1.10X, donde X es el número de máquina (de 0 a 7). Así mismo, y siguiendo el mismo patrón, las IPs asignadas a la red cableada solamente difieren en el tercer dígito de la IP (192.168.2.10X).

La automatización de este proceso se entenderá mejor tras ver los apartados 4.1.3.1 y 4.1.3.3.

3.2.2. Route

Como la red empleada para las pruebas es estática no es necesaria la instalación de ningún protocolo de encaminamiento para MANETS (como podría ser OLSR o AODV) por lo que utilizaremos la aplicación *"route"* que viene por defecto en la mayoría de sistemas operativos GNU/Linux.

Route es una utilidad que nos permite tanto modificar como visualizar las tablas de encaminamiento en el ordenador deseado.

1. route add 192.168.1.101 gateway 192.168.1.105
2. route del 192.168.1.102 gateway 192.168.1.105

Figura 3.6: encaminamiento estático

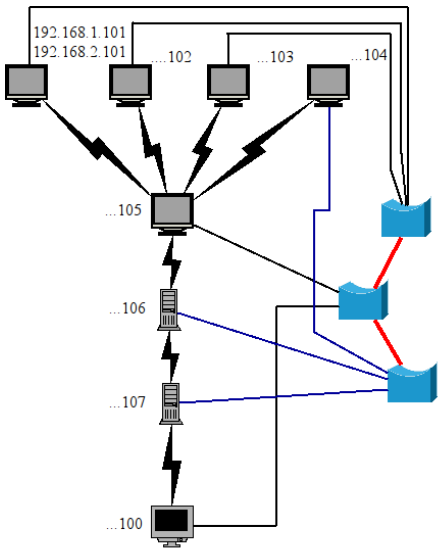

Figura 3.7: Diagrama de la infraestructura resultante

Su funcionamiento es sencillo: debemos especificar la IP destino final y la IP de la máquina que encaminará el tráfico.

Como se puede observar en la figura 3.6, para añadir una ruta hay que utilizar la opción *"add"* y para eliminarla la opción *"del"*.

Como inconveniente podemos destacar que hay que incluir una línea para cada ordenador destino, ya que al emplear las rutas por defecto el comportamiento no es el esperado a causa de los ARP.

3.2.3. Resultado conseguido

Tras la aplicación de todos los comandos el resultado es el que se puede apreciar en la figura 3.7. En este diagrama se pueden apreciar todas las IPs empleadas en cada una de las máquinas así como la conexión cableada. Si se observa detenidamente esta conexión se puede observar que todos los equipos que emiten tráfico y el que recibe (fuentes y destino) están a la misma distancia para que el retardo de la red backbone sea constante.

Una imagen del entorno en el que se realizaron todas las pruebas se puede encontrar en la figura 3.8, se puede observar el tamaño del entorno de las pruebas.

Figura 3.8: Imagen del banco de pruebas

Capítulo 4

Automatización de baterías de pruebas

4.1. Fase 1: Desarrollo manual

Para conseguir mediciones de retardo y tasa de bits se desarrolló un generador de tráfico UDP y una plataforma de lanzamiento de ejecuciones programadas, así como de recolección de datos.

4.1.1. UdpFlow

Una de las partes principales de nuestro banco de pruebas es la inclusión de un generador de tráfico UDP con una tasa constante de bits (CBR).

La aplicación en cuestión consta de dos módulos:

1. Generador de tráfico (Cliente).

2. Receptor de tráfico (Servidor).

El generador de tráfico es encargado de enviar un flujo con una tasa constante de bits y, en caso de ser necesario, es el que reserva el socket DACME (como veremos más adelante en los capítulos 7 y 8). También es el encargado de, al comienzo de cada paquete, incluir el tiempo actual.

El receptor de tráfico es el encargado de las mediciones de retardo y de tiempo con el que llegan los paquetes respecto al primero recibido.

La medición del retardo solamente es válida si se realiza en la misma máquina por el problema de sincronización de relojes, aunque si los relojes estuvieran sincronizados con NTP (Network Time Protocol) [1] se podría considerar que el tiempo medido sería bastante aproximado (pero no exacto).

Parte del cliente

1. Generar el par de claves rsa con el comando *"ssh-keygen -t rsa"*
2. Copiar la clave pública al servidor
   "scp ~/.ssh/id_rsa.pub usuario@servidor:.ssh/id_rsa.pub"
3. Almacenar las claves para poder utilizarlas al acceder al servidor ssh *"ssh-add"*

Parte del servidor

1. Modificar el fichero *"/etc/ssh/sshd_config"* y añadir las líneas
   - RSAAuthentication yes
   - PubkeyAuthentication yes
2. Añadir la clave del cliente a nuestra lista de claves autorizadas *"cat .ssh/id_dsa.pub >> .ssh/authorized_keys"*

Figura 4.1: Configuración de SSH para acceso sin contraseña interactiva

1. ssh 192.168.2.101 ~/TestBed/UDP/UdpFlowClient 192.168.1.100 64000 512 250 300 1 135 1
2. scp 192.168.2.101:~/tiempos_64000.txt ./Retrasos/nombre_puerto_caso.txt

Figura 4.2: Ejemplo de uso de SSH

4.1.2. Secure Shell (SSH)

La principal herramienta usada para la ejecución remota de los diversos programas necesarios es *"ssh"*. Esta aplicación permite la ejecución remota así como la transferencia de ficheros entre dos máquinas debidamente configuradas.

En nuestro caso todos los ordenadores disponían de un servidor ssh configurado de tal forma que el acceso por parte del ordenador principal (desde el que se coordina la ejecución de las aplicaciones) no requiera de contraseña introducida de modo interactivo para poder realizar su cometido.

En la figura 4.1 se puede ver cómo configurar ssh para conseguir un acceso sin tener que teclear una contraseña.

En la figura 4.2 se muestran dos ejemplos de funcionamiento de ssh: en el primero se lanza una ejecución en una máquina de la aplicación UdpFlowServer, y en el segundo se muestra un ejemplo de recolección de resultados desde un computador remoto a una carpeta local.

4.1.3. Lanzador de ejecuciones

Las ejecuciones de las diferentes pruebas se automatizaron y se sincronizaron mediante una plataforma desarrollada principalmente con el lenguaje de scripting bash.

Esta plataforma debía de cumplir diversas características muy específicas:

- Posibilidad de seleccionar origen y destino de las comunicaciones, así como las características de la comunicación, como pueden ser tasa de bits, tamaño de paquete y categoría con la que serán enviados los paquetes.
- Cuando se lancen ejecuciones usando sockets de DACME también se debe poder elegir el retardo máximo.
- Cada prueba individual se puede repetir un número ilimitado de veces.
- Las pruebas en su conjunto se pueden repetir un número ilimitado de veces.
- Las mediciones de los parámetros de la red se deben de realizar en diversas máquinas de la red sin afectar al rendimiento de esta.
- Todos los resultados de cada una de las iteraciones de pruebas en su conjunto deben quedar agrupados y claramente diferenciados.
- Los resultados de cada prueba individual debe de ser accesible individualmente.
- La aplicación de lanzamiento de ejecuciones debe de controlar que los puertos utilizados por las diversas máquinas sean compatibles para evitar funcionamientos incorrectos (Dos aplicaciones enviado datos al mismo receptor, o dos receptores escuchando a la vez por el mismo puerto).
- El diseño debe de ser modular para facilitar las modificaciones así como la detección y corrección de errores.

En esta sección se irán describiendo las diversas partes de que dispone el lanzador de ejecuciones.

4.1.3.1. Parser

El parser desarrollado para este entorno de pruebas es el corazón de la aplicación ya que todos los componentes que necesitan alguna información sobre lo que deben hacer demandan la información a esta aplicación. Por sencillez a la hora de procesar los ficheros de configuración, este se ha programado en C++, disponiendo de varios modos de funcionamiento adaptados a las necesidades de cada módulo de la plataforma de lanzamiento de pruebas.

Como se puede observar en la figura 4.3, el parser recibe dos argumentos, ambos enteros positivos; el primero de ellos -*"Número de caso"*- es el caso que queremos utilizar para las pruebas.

./parser *"Número de caso"* *"Opción"* < *"Fichero de configuración"*

Figura 4.3: Funcionamiento del parser

El segundo argumento -*"Opción"*- está comprendido entre 1 y 7, y cada uno nos proporciona lo siguiente por salida estándar:

1. Lista de capturadores.
2. Lista de servidores UDP.
3. Lista de prioridades.
4. Lista de clientes.
5. Numero de procesos a esperar.
6. Numero de escenarios.
7. Nombre del fichero de salida.

El parser dispone de varias opciones a la hora de compilar que dan soporte a diferentes tipos de ficheros de configuración.

DELAY Con esta opción se consigue lanzar en la misma máquina tanto el proceso Cliente como el Servidor, consiguiendo así una medición exacta del retardo. En caso contrario el Cliente se ejecutaría en la máquina origen y el Servidor en la máquina destino, sirviendo así solamente para medición de rendimiento.

CAPTURADORES Al compilar con esta opción se activa el soporte para la inclusión de diversos capturadores de tráfico en las distintas máquinas del entorno durante un tiempo establecido. Si no se compila con esta opción se omite la lectura de los capturadores en el fichero.

SALTO Si se compila con esta variable definida en un mismo fichero se pueden incluir varios casos con configuraciones distintas. En caso contrario se repetirá siempre el primer caso completo que aparezca en el fichero.

DACME Para la segunda parte del proyecto (Capítulo 7) se necesita que la identificación de las prioridades sea numérica y no las que se usan en iptables. Si no se compila con esta opción la salida de prioridades pasa a ser la estándar de iptables *"CS1", "CS5", "BE", ...*

```
#Comentarios

5pc-4flujo1000KB
9

0
4
1 0 30 0 0 300 512 250
2 0 30 0 1 300 512 250
3 0 30 0 2 300 512 250
4 0 30 0 3 300 512 250
```

Figura 4.4: Ejemplo de fichero de configuración

4.1.3.2. Ficheros de configuración

Los ficheros de configuración tienen el aspecto que podemos ver en la figura 4.4.

Estos ficheros pueden contener comentarios al comienzo del mismo. Los comentarios son todas aquellas líneas que comienzan con el carácter '#'.

A los comentarios les sigue el nombre que será usado en los ficheros de salida; tras el nombre solamente deben aparecer números enteros.

El significado de estos números es el siguiente:

En primer lugar se encuentra el número de casos a procesar. Dependiendo del método de compilación del programa parser (ver 4.1.3.1) se repite siempre el primer caso o se selecciona el caso correspondiente.

Para cada caso se dispone de:

- El número N de instancias de programas de captura de tráfico a lanzar seguido por N líneas, cada una de las cuales indica los siguientes datos:
  - Máquina donde lanzar el capturador.
  - Tiempo de funcionamiento del capturador.
  - Interfaz de red por la que se lanzará la captura.
- El número de flujos de trafico a crear, cada uno de los cuales consta de la siguiente información:
  - Máquina origen del tráfico.
  - Máquina destino del tráfico.
  - Tiempo de ejecución de cada flujo.
  - Retraso previo a la creación del flujo.

- Prioridad con la que será etiquetado el tráfico de este flujo.
  - 0 $\Longrightarrow$ Background (AC_BK)
  - 1 $\Longrightarrow$ Best Effort (AC_BE)
  - 2 $\Longrightarrow$ Vídeo (AC_VI)
  - 3 $\Longrightarrow$ Voice (AC_VO)
- Retardo demandado para el flujo al utilizar DACME (Ver capítulo 7).
- Tamaño del paquete enviado en bytes.
- Bitrate del flujo en concreto.

4.1.3.3. Lanzamiento de órdenes remotas

El método de lanzamiento de las órdenes a cada uno de los ordenadores que participan en las pruebas está estructurado en varias partes para conseguir una mejor organización y facilidad de introducción de cambios.

El lanzamiento de los diferentes comandos de iptables, UDPFlow y tcpdump (capturador de tráfico) se debía de realizar desde la máquina "central", es decir, la que dispone de todos los ficheros de configuración.

Para la ejecución de estas aplicaciones se hace uso del comando ssh, como vimos en la sección 4.1.2. Naturalmente para que este tráfico de red no desvirtúe nuestras pruebas, estos comandos se lanzan todos por la red Ethernet de backbone que disponemos y explicamos anteriormente en el punto 3.1.

El lanzador de pruebas se divide de la siguiente forma:

1. En primer lugar disponemos del componente que se encarga de conseguir que los bancos de pruebas se ejecuten varias veces, así como de organizar los resultados obtenidos en las diferentes carpetas: *"Lanzador principal"*

2. La segunda parte *"Lanzador de ficheros"* es la encargada de escanear el directorio actual en busca de los ficheros de configuración, e ir ejecutándolos uno a uno. Para ello hace uso de diversos componentes especializados en cada una de las partes de que se compone el entorno de pruebas. El orden que sigue para la ejecución es el siguiente:

   a) En primer lugar se inicializa el semáforo para evitar problemas de sincronización: *"Inicializar mutex"*

   b) En segundo lugar se llama al proceso*"Capturadores de tráfico"* encargado de lanzar los capturadores de tráfico.

c) Tras los capturadores se lanza el proceso *"Redirección de puertos"* que se encarga de redireccionar los puertos utilizados en las pruebas a sus máquinas origen, como vimos en la figura 3.5 del apartado 3.2.1.

d) El siguiente paso es lanzar la ejecución de los *"Servidores UDP"* (recolectores de estadísticas) en las máquinas destino del tráfico.

e) Posteriormente se lanza el proceso *"Prioridades de los flujos"* que asigna prioridades a los diferentes flujos mediante comandos iptables.

f) Una vez realizado todo lo anterior solamente nos queda lanzar los *"Clientes UDP"* o generadores de tráfico.

g) Por último se entra en la fase de limpieza (*"Recolección y ordenación de los datos"*); esto incluye, tanto la espera de los procesos lanzados anteriormente como la eliminación de las distintas prioridades asociadas a cada flujo y la recolección de los diferentes resultados que se almacenan en cada una de las máquinas.

Como se comentó anteriormente, todos estos módulos tienen acceso al parser y es al que le demandan toda la información requerida para las diversas ejecuciones.

Para obtener una visión más gráfica de este entorno de pruebas véase la figura 4.5.

4.1.3.4. Salidas

Las salidas que proporciona el programa están estructuradas de la siguiente manera:

En directorios separados cada una de las ejecuciones del bando de pruebas. Dentro de estos directorios una lista de archivos cuyo nombre está estructurado de la forma que podemos ver en la figura 4.6, es decir, en primer lugar encontramos el nombre que habíamos definido en el fichero de configuración correspondiente, en segundo lugar tenemos el puerto por el que han sido realizadas las mediciones, y en tercer lugar el número de medición.

Cada uno de estos ficheros dispone de tres columnas:

1. Retardo del paquete desde que se envió (incluye el tránsito por la red backbone)
2. Tiempo transcurrido desde la llegada del primer paquete.
3. Tamaño en bytes del paquete recibido

Con estas informaciones se puede obtener fácilmente la productividad y retardo de la red, así como mediante un pequeño procesamiento dibujar diversas gráficas informativas como puede ser la evolución del retardo o de la tasa de bits a lo largo de todo el tiempo de transmisión.

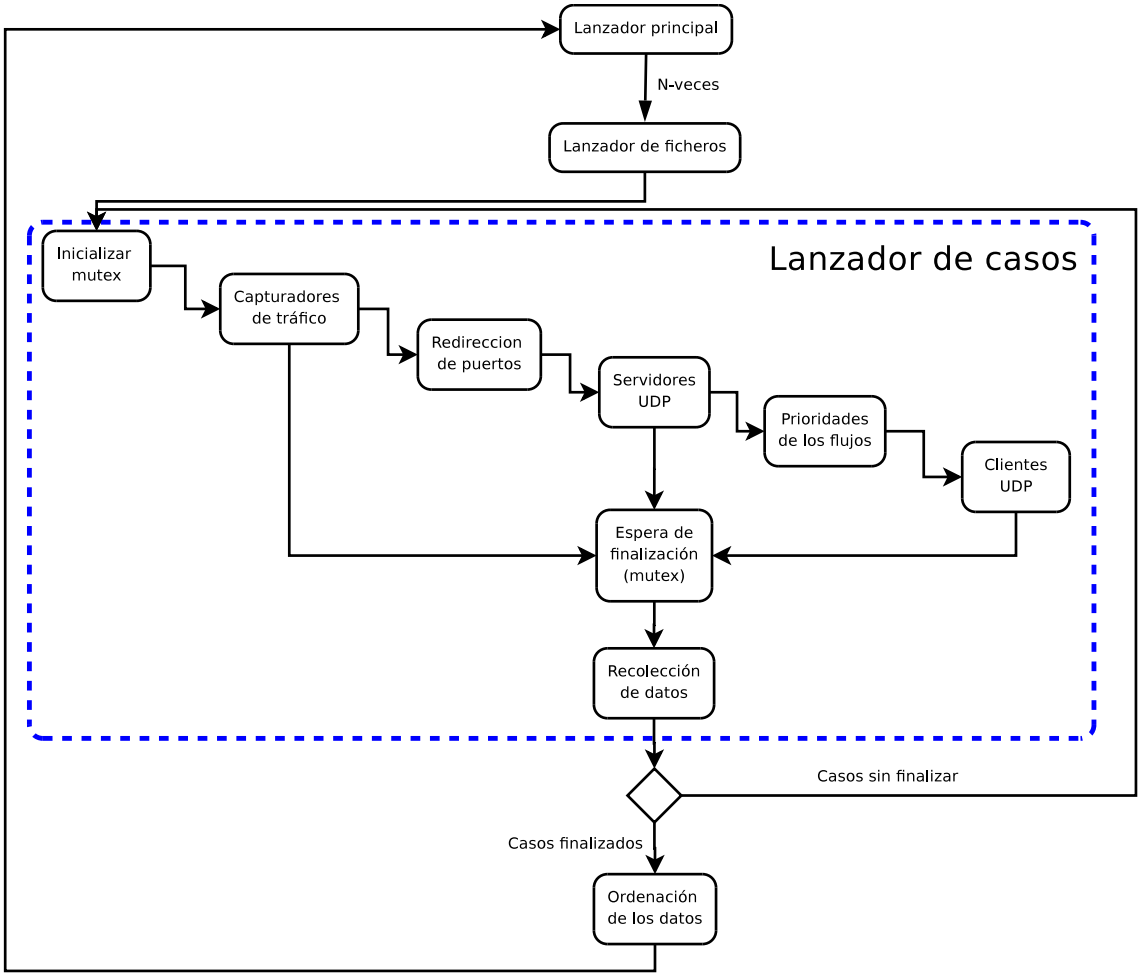

Figura 4.5: Organización del lanzador de ejecuciones

4.1.3.5. Problemas más críticos y soluciones adoptadas

El primer problema que se planteó al lanzar las diversas ejecuciones es la forma de ejecutar varias aplicaciones en paralelo pero, a la vez, siguiendo un orden secuencial, es decir, que hasta que no se hayan lanzado todos los comandos de ejecución de los servidores UDP no se pase a lanzar los de los clientes.

Para conseguir esto se creó una "capa intermedia" que se encarga de leer secuencialmente las ordenes del parser y ejecutar en background las órdenes remotas.

Derivado de la solución anterior otro de los principales problemas de la realización

5pc-4flujo400KB_64013_9.txt

5pc-4flujo1800KB_64014_1.txt

Figura 4.6: Nomenclatura de los ficheros de resultados

del lanzador de ejecuciones en el lenguaje de scripting de bash es la sincronización entre ejecuciones y procesos.

Para poder solucionar esto se desarrolló un semáforo para controlar cuando acaban todas las ejecuciones y poder pasar con seguridad a la siguiente prueba.

La efectividad del semáforo no es del 100 %, pero ofrece la suficiente robustez para poder lanzar las ejecuciones con seguridad.

Además de la inclusión de este semáforo se optó por dejar tiempos de espera de varios segundos entre cada ejecución del test. Con esto se consiguió evitar cualquier fallo en la sincronización de la ejecución de las pruebas.

4.2. Fase 2: Integración en Castadiva

Tras la creación de un entorno de pruebas funcional con el que se obtuvieron los resultados que se presentan posteriormente en el capítulo 5.

Como uno de los objetivos de este proyecto es facilitar la realización de pruebas en MANETs con QoS habilitada, se decidió integrar las mejoras en un emulador de redes MANET desarrollado previamente por el Grupo de Redes de Computadores, ya que el entorno de ejecución desarrollado en la primera fase tiene limitaciones de uso y de configuración que limitan el tipo de pruebas que se pueden realizar.

4.2.1. Descripción

Castadiva [14] es un emulador de redes MANET desarrollado en Java que proporciona una interfaz gráfica amigable para el usuario permitiéndole probar aplicaciones de código real emulando movilidad de los nodos.

Castadiva proporciona una alternativa barata a la simulación, empleando para ello dispositivos reales de bajo coste con los únicos requisitos de emplear un sistema operativo GNU/Linux y que sus tarjetas de red soporte el modo ad-hoc. Actualmente una gran variedad de dispositivos de bajo coste cumplen con estas restricciones ya que podemos encontrar tanto netbooks como routers/nat inalámbricos totalmente compatibles.

Castadiva presenta una arquitectura modular, tal y como se puede observar en la figura 4.7, Castadiva consta de un servidor que es el encargado de controlar las emulaciones y de un número indeterminado de nodos. Los nodos, como se ha comentado anteriormente, deben de estar funcionando con un sistema GNU/Linux, ya que se hace un uso intensivo de herramientas como “ssh”, “nfs” o “iptables”.

En la figura 4.8 se puede observar la ventana más importante de Castadiva. En ella se puede configurar tanto la disposición de los nodos como su rango de cobertura

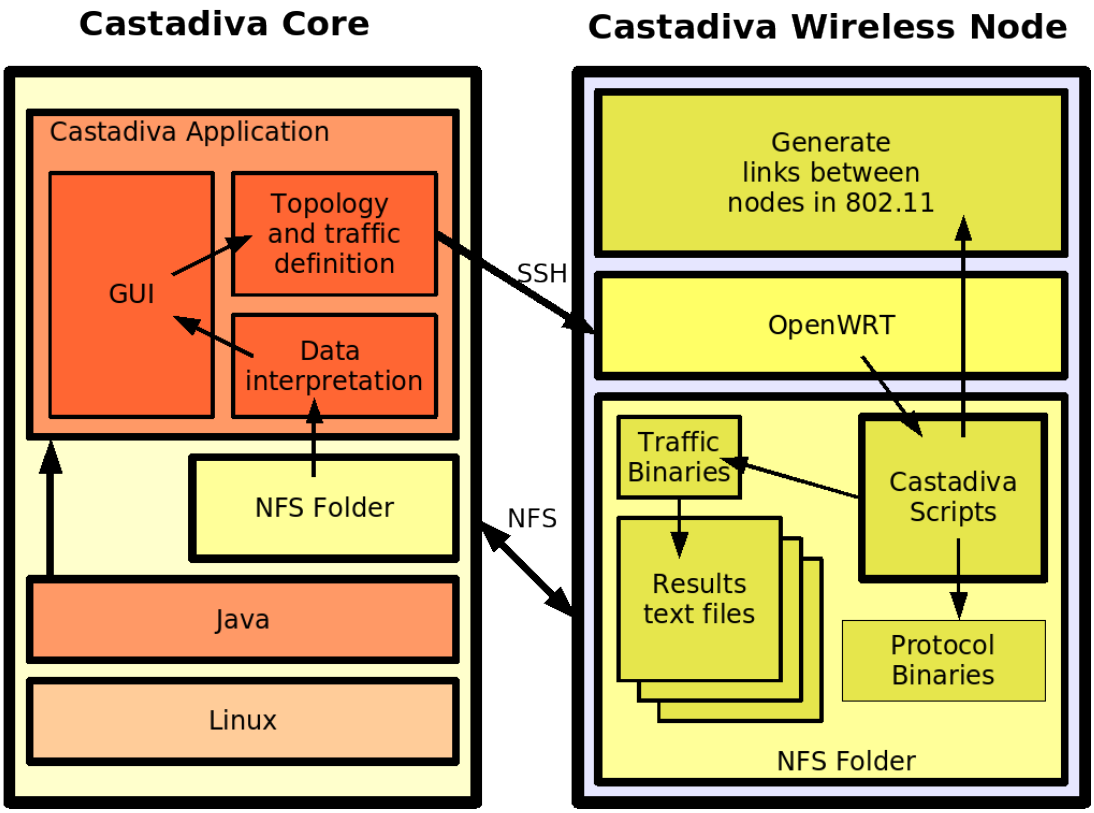

Figura 4.7: Arquitectura de Castadiva

emulado, la velocidad de movilidad que tendrán así como los flujos de tráfico que existirán en el escenario (al hacer click en "Declare Traffic").

Además se pueden observar las opciones de la parte inferior derecha, que permiten entre otras cosas guardar y cargar emulaciones, para facilitar la tarea de mantener la repetibilidad de los experimentos.

4.2.2. Funcionalidades añadidas

Dado que se iba a modificar el código de la aplicación existente se abrieron dos nuevos frentes, el primero es el de la inclusión de las prioridades definidas en el estándar IEEE 802.11e, por otra parte se modificó la aplicación para permitir una mayor versatilidad a la hora de lanzar los tests y recolectar estadísticas.

4.2.2.1. Soporte para IEEE 802.11e

Como se ha explicado en la sección 2.1.4, en IEEE 802.11e existen diferentes categorías de acceso, por lo que en la interfaz gráfica de Castadiva se ha añadido un selector de la categoría. Esta categoría solamente está disponible para tráfico UDP, ya que es el que se puede aprovechar de estas mejoras (voz y vídeo) mientras que

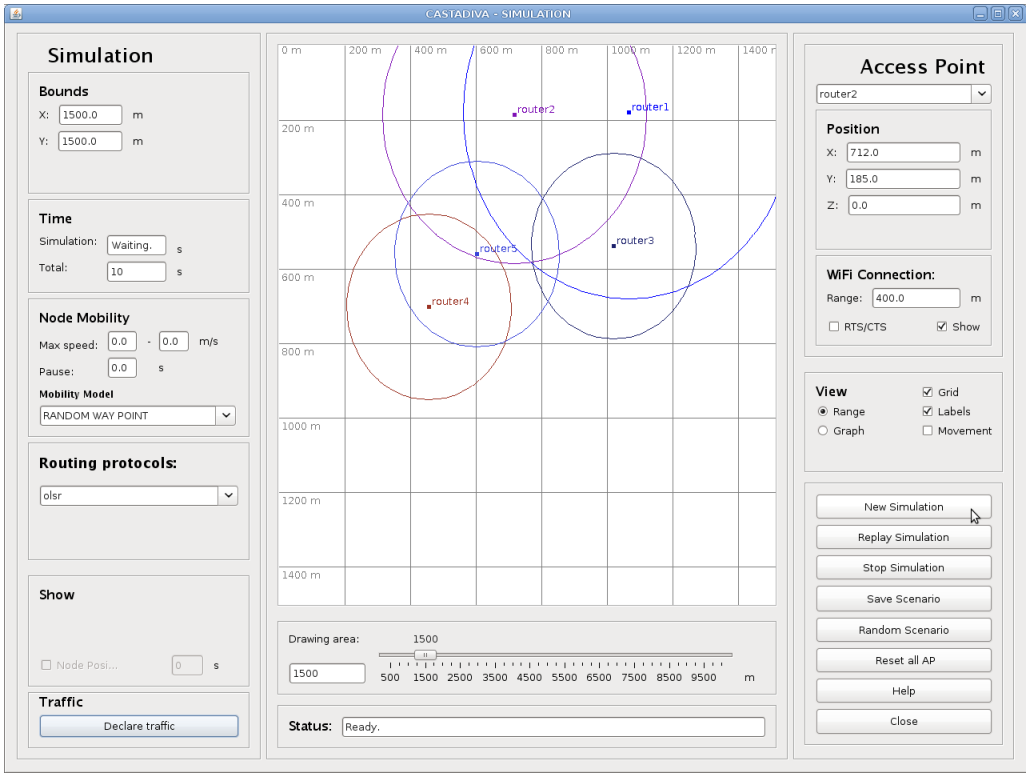

Figura 4.8: Ventana de configuración de una emulación

para el tráfico TCP está desactivado, ya que en este caso, el patrón de tráfico simula ser un FTP que por definición es Best Effort.

En la figura 4.12 se puede observar que el círculo marcado con 1 corresponde a la lista desplegable donde seleccionar la categoría de acceso.

Para que las aplicaciones puedan disfrutar de la QoS proporcionada por la MAC los datagramas deben tener el campo TOS (Type of service) de la cabecera IP rellenados de acuerdo a la prioridad que desee el usuario. Como se ha comentado anteriormente la relación prioridad IP con categoría de acceso MAC se puede ver en la tabla 2.3 de la página 15.

En el caso de que la red subyacente no tenga IEEE 802.11e habilitado marcar el campo TOS no tendrá ningún efecto, ni negativo ni positivo, sin embargo si la red si que posee IEEE 802.11e habilitado la diferenciación de tráfico será efectiva.

4.2.2.2. Medición del retardo para flujos UDP

La medición del retardo para paquetes UDP es un problema que no tiene una fácil solución en un entorno real (no así en simulación donde se dispone de un reloj global y se puede determinar cuándo un paquete se envía o se recibe en una tarjeta de red). Sin embargo en un entorno real debemos de convivir con la desincronización de relojes existente entre diferentes nodos que dificulta enormemente el proceso.

Para lidiar con este problema se han implementado don métodos distintos, uno más preciso y otro sencillamente orientativo. La base de ambos métodos es la misma, en el generador de tráfico UDP, en el nodo fuente se obtiene el tiempo y se marca el paquete con él, en el nodo destino se repite la operación de obtención de tiempo para cada paquete que se recibe y se compara con la marca que llevaba el paquete.

El primer método lucha contra la desincronización de los relojes usando NTP, pero como ya se comenta anteriormente este método no tiene la suficiente precisión para poder realizar buenas mediciones y además durante el transcurso de la emulación los relojes se van desincronizando y dado que necesitamos una precisión de milisegundos cualquier pequeña desviación nos invalida los resultados. Sin embargo para la obtención de valores estimados (alto retardo o bajo retardo) es suficiente.

El segundo método requiere emplear la red Ethernet por la que se mandan las órdenes de Castadiva para redirigir el tráfico otra vez al nodo fuente empleando órdenes iptables como las que podemos observar en la figura 3.5 (página 23), por lo que en este caso los dos programas (origen y sumidero) se ejecutarían en la misma máquina y no habría desfase de reloj. Naturalmente el empleo de la red Ethernet introduce un retardo que debe ser tenido en cuenta a la hora de analizar los resultados, pero aún así el retardo introducido por esta red puede ser considerado despreciable ya que se encuentra varios órdenes de magnitud por debajo del retardo típico de una red ad-hoc multisalto.

En la figura 43 se puede observar cómo seleccionar entre los dos métodos para la medición del retardo. Si en la columna redirect el flujo se encuentra marcado como false se empleará el primer método (ntp) mientras que si está marcado como true se empleará el método preciso (redirección al origen).

4.2.2.3. Sistema de plugins

La primera versión de Castadiva era completamente funcional, pero solamente poseía un único modelo de movilidad y tres algoritmos de encaminamiento. con la creación y carga dinámica de plugins se ha conseguido un soporte para casi cualquier protocolo de encaminamiento y modelo de movilidad.

Protocolos de encaminamiento Las versiones anteriores de Castadiva ya soportaban unos pocos protocolos de encaminamiento. Por ejemplo OLSR y AODV.

Para que los protocolos fuesen funcionales, debían estar previamente instalados y configurados en cada uno de los nodos que formaban parte de la emulación. Lo que hacia Castadiva cuando comenzaba una emulación era activar el protocolo seleccionado, enviando un sencillo comando vía SSH. Cuando la emulación concluía el protocolo se desactivaba de la misma manera.

Sin embargo el método tenía varias limitaciones, en primer lugar el comando enviado estaba incluido en el código de la aplicación, por lo que si se requería cambiar algo del comando había que recompilar Castadiva al completo.

Con el desarrollo de este nuevo sistema de plugins la idea es permitir al usuario configurar y lanzar la ejecución de cualquier protocolo de encaminamiento que soporten los nodos.

Para ello, como se puede observar en la figura 4.9 el usuario dispone de una interfaz relativamente autoexplicativa. En primer lugar se encuentra el nombre que aparecerá en Castadiva para seleccionar el protocolo de encaminamiento. En segundo lugar se debe de proporcionar la ruta absoluta al ejecutable en los nodos. En el campo *Flags* se pueden introducir parámetros de ejecución para el protocolo de encaminamiento. En el centro de la ventana se encuentra un gran espacio para introducir el contenido del fichero de configuración del algoritmo de encaminamiento (si se desea). Por último se debe introducir la ruta donde debe ser copiado ese fichero de configuración en cada uno de los nodos para que el algoritmo funcione según nuestras preferencias.

El funcionamiento en detalle es el siguiente:

1. Se copia el contenido del fichero de configuración a la ruta especificada en cada uno de los nodos.

2. Se ejecuta en cada uno de los nodos el fichero binario empleando la ruta especificada en "*Bin*".

3. Se ejecuta la emulación.

4. Se detiene el algoritmo de encaminamiento en cada uno de los nodos.

Movilidad También se extendió Castadiva para poder definir modelos de movilidad personalizados durante una emulación. Como se puede observar en la figura 40 la interfaz de usuario de Castadiva permite a los usuarios aprovechar toda la potencia del lenguaje de programación JAVA para calcular la posición de cada nodo en cada segundo de la emulación.

Dada la información de la simulación (lista de nodos, máxima velocidad, mínima velocidad, posiciones iniciales...) el usuario puede escribir el código deseado cumpliendo un solo requisito: rellenar la matriz *NodeCheckPoint[i][j]* con la posición de los nodos. En esta matriz i representa un punto de acceso y j el segundo de la emulación. Hay que tener en cuenta que todos los nodos deben de tener una posición definida en todos los segundos de la emulación.

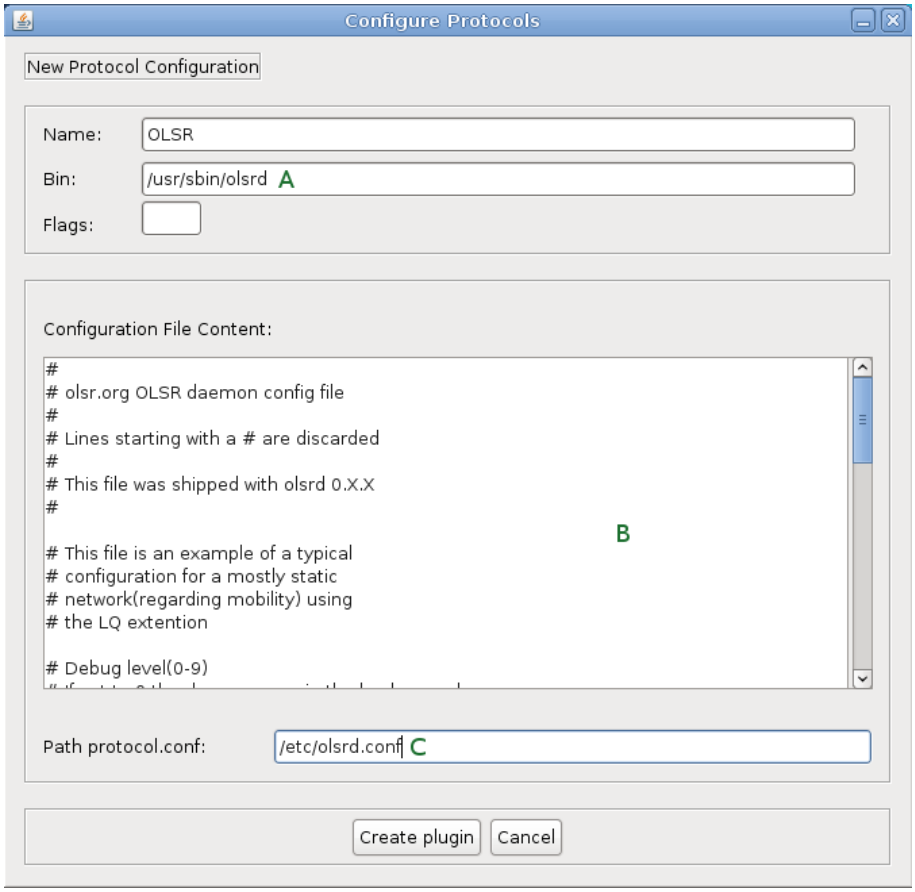

Figura 4.9: Creación de un nuevo plugin de encaminamiento

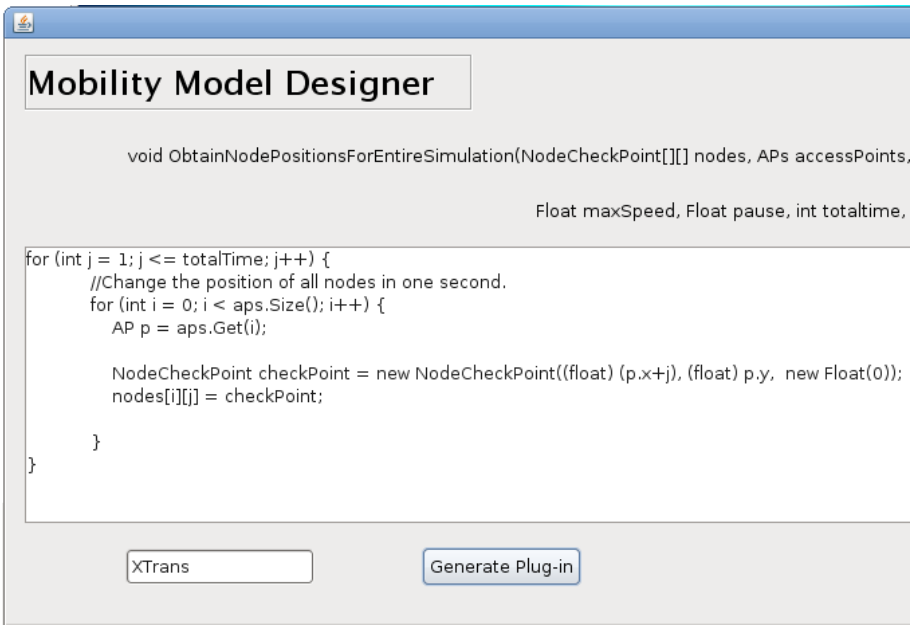

Figura 4.10: Creación de un nuevo plugin de movilidad.

Si el código es correcto, el plugin será compilado e integrado en Castadiva, pasando a estar disponible en la ventana de configuración de la emulación. En caso de que no compile correctamente un mensaje con el error de compilación será mostrado al usuario.

Un ejemplo de un código 100 % funcional se puede ver en la figura 42. En este caso este algoritmo de movilidad emula a un usuario que se desplaza diagonalmente por un escenario. El nodo móvil es el último de la lista (por lo que sabemos exactamente cual es cuando configuramos la emulación).

4.2.2.4. Planificador de ejecuciones

La primera versión de Castadiva tenia un gran inconveniente, las pruebas debian de configurarse y ejecutarse manualmente una a una haciendo que la creación de un gran número de experimentos sea una tarea muy ineficiente. Con esta nueva característica se ha conseguido ejecutar una gran cantidad de pruebas secuencialmente permitiéndonos recolectar y procesar una gran cantidad de estadísticas.

El planificador de ejecuciones por si solo ya es una herramienta que proporciona una gran funcionalidad, pero si además se combina con el nuevo sistema de plugins se puede automatizar el proceso de lanzamiento de ejecuciones, variando no solamente los nodos origen/destino o la cantidad de tráfico de los diferentes flujos, si no que se puede variar tanto el algoritmo de encaminamiento como el modelo de movilidad. Es por eso que consideramos que Castadiva es una buena plataforma para evaluar los resultados de ns-2 y efectuar las comparaciones pertinentes.

Como se puede observar en la figura 4.13, la interfaz del planificador proporciona una manera fácil de crear y automatizar la evaluación de redes MANET. Incluye funcionalidades como “Load list” que importa una lista de emulaciones guardada previamente con la opción “Save list”. El componente principal de la ventana es una tabla que contiene las diferentes emulaciones que Castadiva procesará (columna “Source Folder”), así mismo incluye la ruta donde se almacenarán los resultados (columna “Results folder”), las siguientes columnas (“Runs” y “Status”) corresponden al número de veces que repetirá cada emulación y a un mensaje de información para el usuario detallando el estado de la emulación, respectivamente.

Los resultados se almacenan en un directorio llamado “Iterations” dentro de la carpeta especificada en “Results folder”. Dentro de ese directorio cada emulación crea un fichero llamado “X_DefinedTraffic.txt” donde X corresponde a un número entre 1 y el número de repeticiones especificado en la columna “Runs”. Un ejemplo de este fichero se puede encontrar en la figura 4.14.

```
//X and Y direction of the movement
int sentidoX = 1;
int sentidoY = 1;
//The last position for the node in movement
float lastX = 0;
float lastY = 0;

for (int j = 1; j <= totalTime; j++) {
        //Change the position of all nodes in second j
        for (int i = 0; i < accessPoints.Size()-1; i++) {
                AP p = accessPoints.Get(i);
                nodes[i][j] = checkPoint;
        }

        //Get the information of the last node
        int   lastNode = accessPoints.Size()-1;
        AP p = accessPoints.Get(lastNode);

        //Get the movement of the node
        float incX = sentidoX * minSpeed;
        float incY = sentidoY * minSpeed;

        //Position at the first second
        if( j == 1) {
                lastX = p.x;
                lastY= p.y;
        }

        //Calculating the new position
        float newX = lastX + incX;
        float newY = lastY + incY;

        //Check if we are not out of the scenario
        if(newX > X || newX < 0) {
                sentidoX = sentidoX * -1;
                newX = lastX - incX;
        }
        if(newY > Y || newY < 0) {
                sentidoY = sentidoY * -1;
                newY = lastY - incY;
        }
        System.out.println("Node "+ lastNode +  " posX = " + newX + "
            posY = " + newY);

        //Put the new position of this second
        NodeCheckPoint checkPoint = new NodeCheckPoint(newX, newY,  new
             Float(0));
        nodes[lastNode][j] = checkPoint;
        //Update the old position of the node
        lastX = newX;
        lastY = newY;
}
```

Figure 4.11: Ejemplo de plugin de movilidad válido. Movimiento diagonal de un solo nodo.

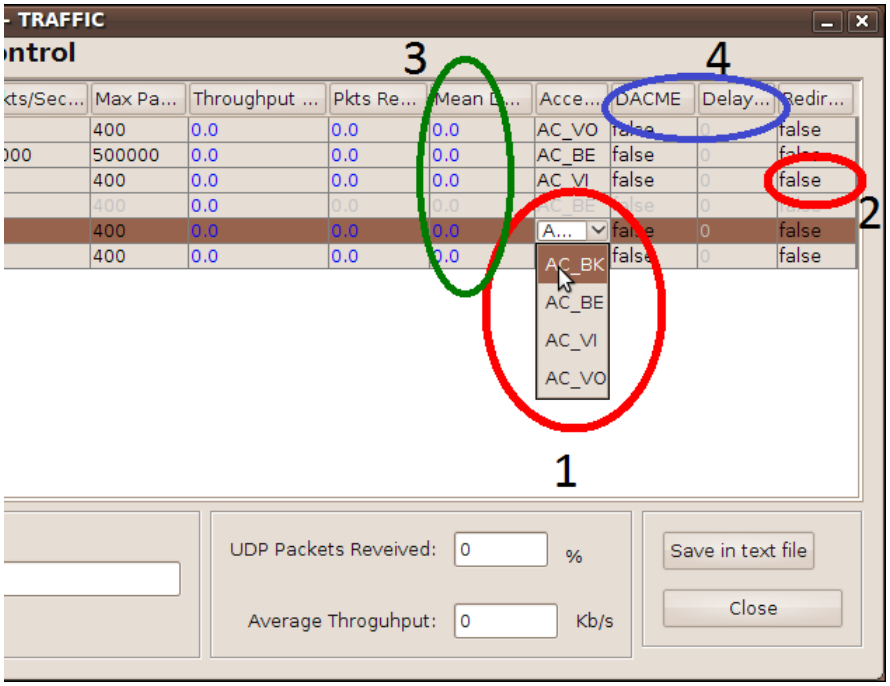

Figura 4.12: Funcionalidades añadidas a Castadiva: Flujos de datos

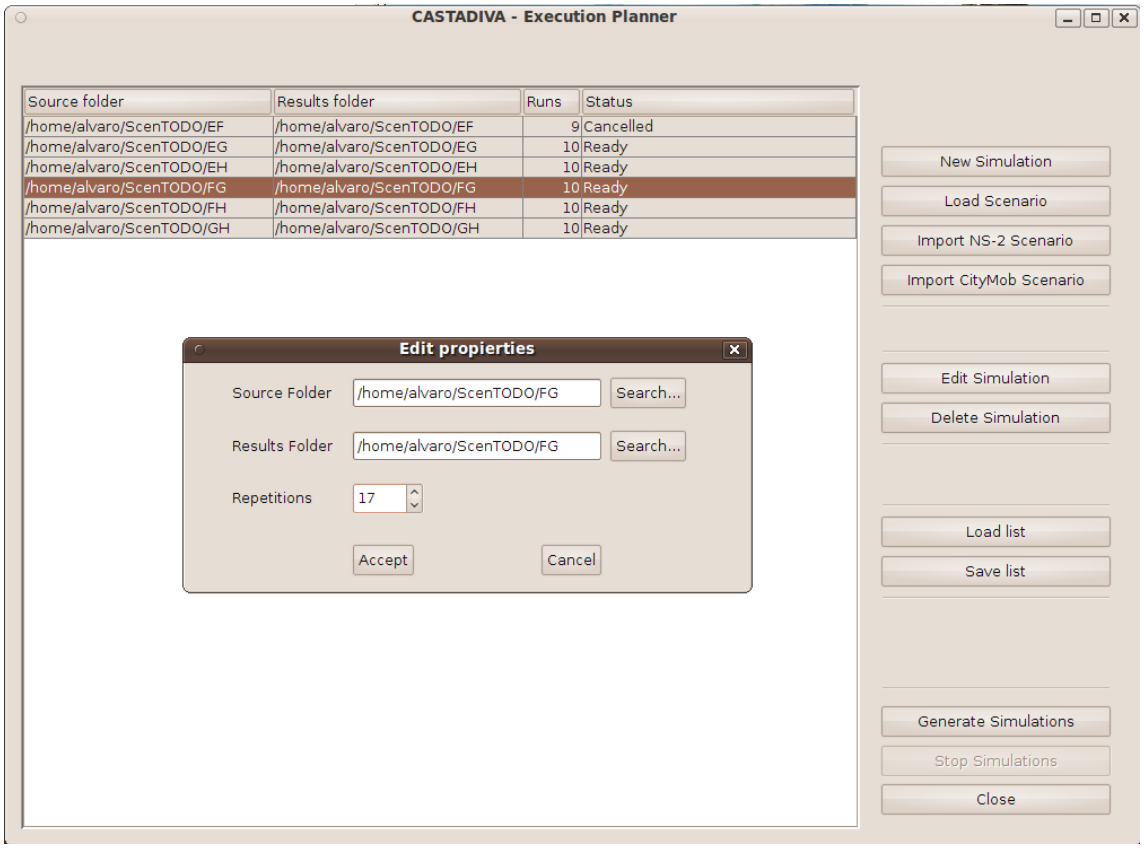

Figura 4.13: Planificador de ejecuciones

Line	Strt	Stop	Srce	Addr	Traff	Transf. Size	Pkt/sec	Packt	Thrghpt	Received		Mean Delay		AC	DACME	Delay	Redirect
1	20	100	EEEPC2	EEEPC6	UDP	640	300	24000	1401.6	91.25	273.825	AC_VI	true	0	true		
2	30	110	FIRENZE	EEEPC3	UDP	640	300	24000	326.4	21.25	561.205	AC_VI	true	0	true		
3	40	120	EEEPC1	EEEPC5	UDP	640	300	24000	1459.2	95.0	126.98	AC_VI	true	0	true		
4	50	130	EEEPC7	EEEPC4	UDP	640	300	24000	1536.0	100.0	89.973	AC_VI	true	0	true		

Total UDP packets received: 76.875
Average throughput: 1180.8

Figura 4.14: Resultados obtenidos con Castadiva

Capítulo 5

Validación del testbed desarrollado

En este capítulo se validará el correcto funcionamiento del entorno de pruebas de dos formas diferentes.

En la primera de ellas se validará el entorno "físico" de ejecución de las pruebas; es decir, comprobar que realmente el entorno funciona con QoS y es multisalto.

Por otra parte se mostrarán resultados experimentales obtenidos y se compararán con unos estudios teóricos anteriores obtenidos mediante simulaciones con ns-2 [9].

5.1. Validación del entorno

Para validar el correcto funcionamiento del entorno multisalto con QoS activada se utilizó la aplicación Wireshark que, al emplear una tarjeta externa no asociada a la red, permite ver todo el tráfico que circula por esta, así como los parámetros de QoS con los que se ha transmitido por la capa MAC.

5.1.1. Camino multisalto

En primer lugar, para validar que el entorno filtra bien los mensajes y crea un entorno multisalto, se procedió a comprobar el camino que siguen los mensajes. Para ello se utilizó el comando ping que, como podemos ver en la figura 5.1, nos muestra que el camino de ida y vuelta de un paquete icmp de *echo* es el correcto.

La segunda forma de comprobar que este camino era totalmente real fue al utilizar la herramienta Wireshark ya citada anteriormente.

Como podemos observar en la figura 5.2, los paquetes son reenviados de un nodo a otro ya que los paquetes numerados como "60, 61, 63 y 64" es el mismo paquete pero con máquinas fuente y destino diferentes (es decir son los cuatro saltos que realiza el paquete); lo mismo sucede con los paquetes "66, 68, 70 y 71". Además también vemos que el mensaje de contestación *"reply"* no aparece hasta que el mensaje de

```
$ ping -R 192.168.1.102
PING 192.168.1.102 (192.168.1.102) 56(124) bytes of data.
64 bytes from 192.168.1.102: icmp_seq=1 ttl=61 time=12.0 ms
RR:     192.168.1.100
        192.168.1.105
        192.168.1.106
        192.168.1.107
        192.168.1.102
        192.168.1.102
        192.168.1.107
        192.168.1.106
        192.168.1.105

64 bytes from 192.168.1.102: icmp_seq=2 ttl=61 time=3.27 ms (same route)
64 bytes from 192.168.1.102: icmp_seq=3 ttl=61 time=2.74 ms (same route)

--- 192.168.1.102 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 2.740/6.010/12.013/4.250 ms
```

Figura 5.1: Ejecución de ping mostrando el entorno multisalto.

60	8.294195	192.168.1.100	192.168.1.101	ICMP	Echo (ping) request
61	8.328379	192.168.1.100	192.168.1.101	ICMP	Echo (ping) request
63	8.346755	192.168.1.100	192.168.1.101	ICMP	Echo (ping) request
64	8.347280	192.168.1.100	192.168.1.101	ICMP	Echo (ping) request
66	8.347673	192.168.1.101	192.168.1.100	ICMP	Echo (ping) reply
68	8.350468	192.168.1.101	192.168.1.100	ICMP	Echo (ping) reply
70	8.361232	192.168.1.101	192.168.1.100	ICMP	Echo (ping) reply
71	8.361481	192.168.1.101	192.168.1.100	ICMP	Echo (ping) reply

Figura 5.2: Trafico multisalto real

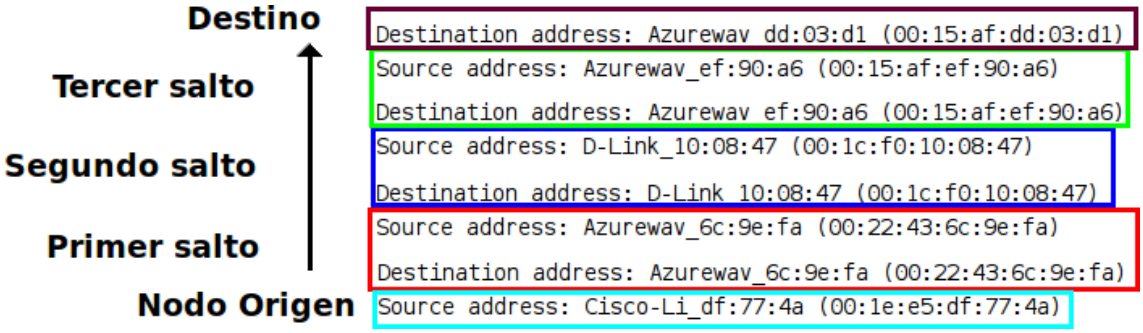

Figura 5.3: Seguimiento por MAC de un paquete

petición *"request"* llega a su destino, por lo que tanto las estaciones finales como las intermedias están filtrando correctamente el tráfico.

Además, si realizamos un seguimiento del paquete desde su origen hacia el destino, podemos observar como el paquete viaja a través de los diferentes nodos intermedios mediante los cambios en las direcciones MAC fuente y destino. En la figura 5.3 podemos ver uno de estos seguimientos. Se puede apreciar claramente como existen 4 saltos desde el nodo origen al nodo destino y como los nodos intermedios realizan correctamente su función de retransmitir los paquetes.

Tras la realización de estas pruebas concluimos que el entorno multisalto estaba correctamente configurado, ajustándose fielmente al modelo lógico establecido en el apartado anterior.

5.1.2. Calidad de servicio (QoS)

Para comprobar que la calidad de servicio estaba activada se realizaron capturas de tráfico de paquetes de manera individual. Además esas mismas capturas se utilizaron para demostrar que, dada una regla de iptables que modificaba la prioridad, ésta se veía correctamente modificada a nivel MAC.

En la figura 5.4 tenemos en un primer lugar, una captura donde se observa específicamente la inclusión de los parámetros QoS en la capa MAC de IEEE 802.11. Las siguientes capturas son pruebas de cada una de las cuatro categorías definidas

en el estándar IEEE 802.11e conseguidas mediante la aplicación de los comandos de iptables anteriormente descritos.

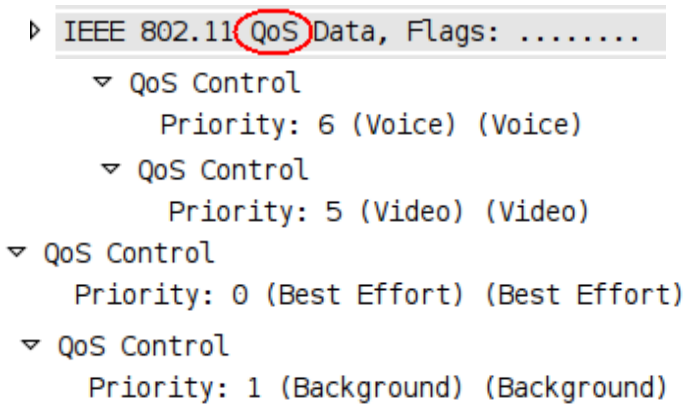

Figura 5.4: Comprobación de la correcta activación de la calidad de servicio y del funcionamiento de todas las categorías.

Esta misma comprobación se realizó para los paquetes enviados por los nodos intermedios que mantenían correctamente su prioridad original.

Por todo esto se puede concluir que el protocolo IEEE 802.11e está activado correctamente y además que el comportamiento esperado por parte del protocolo se ajusta a la realidad.

5.2. Resultados experimentales

Tras haber validado la correcta implementación de entorno de pruebas se procedió a obtener diversos resultados que mostrasen las capacidades de diferenciación de tráfico que nos brinda le tecnología IEEE 802.11e.

En primer lugar se presentan las pruebas obtenidas para tests realizados con una configuración de un solo salto para comprobar el funcionamiento de la tecnología en el contexto para el que está específicamente diseñada.

En segundo lugar presentaremos las pruebas obtenidas en el entorno de pruebas descrito anteriormente.

En último lugar se compararán los resultados obtenidos en nuestro entorno de pruebas con las simulaciones realizadas con anterioridad en [9]. Esta comparación aparecerá publicada en [5].

5.2.1. Un salto

En este primer conjunto de pruebas se mide la efectividad del protocolo IEEE 802.11e en una transmisión de datos de un nodo a otro en su mismo rango de visión.

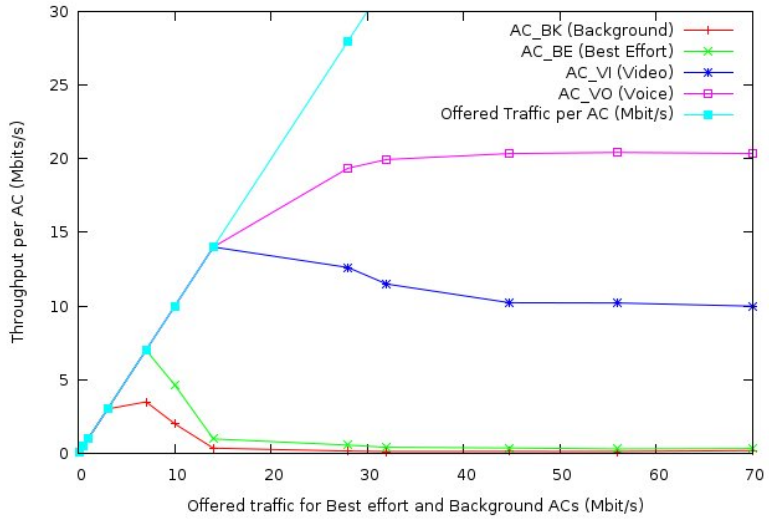

Figura 5.5: Ancho de banda generado y obtenido en cada categoría (1 salto).

Todos los experimentos se realizaron en un entorno semi-controlado ya que, aunque en la red no se generase más tráfico que el usado para realizar las mediciones, es posible que tengamos las interferencias de otras redes que comparten el mismo canal de comunicaciones.

En la figura 5.5 se puede observar como el efecto diferenciador de tráfico que se pretende conseguir con IEEE 802.11e funciona perfectamente.

Además, si observamos el rendimiento conseguido, vemos que supera los 30 Mbits (cuando en una red IEEE 802.11g normal raramente se alcanzan los 20 Mbits). Esto es debido a la reducción del tiempo de espera entre paquetes debido a la reducción de tiempo que supone pasar de tener todos los paquetes esperando un tiempo DIFS a tener a la mayoría de paquetes esperando solamente un tiempo *AIFS[AC_ VO]* o *AIFS[AC_ VI]* (tiempos entre tramas de voz y vídeo).

También se puede observar que, cuando la red comienza a soportar un tráfico considerable (20 Mbits en total), tanto las categorías de *Best Effort* como *Background* ven reducido drásticamente su ancho de banda, alcanzando unos niveles que podríamos considerar inanición (20/30 Kbits).

Cuando la red alcanza su punto de saturación máximo (14 Mbits por flujo) se puede observar como el tráfico de la categoría más prioritaria (*Voz*) tiene una ligera crecida que perjudica a la categoría de *Vídeo.*

En cuanto a las mediciones obtenidas del retardo medio de los paquetes se puede observar en la figura 5.6 que también existe la misma diferenciación que la existente en el retardo.

En este caso se puede observar como tanto los flujos de *Background* y *Best Effort* ven sensiblemente incrementado el tiempo que un paquete tarda en llegar al

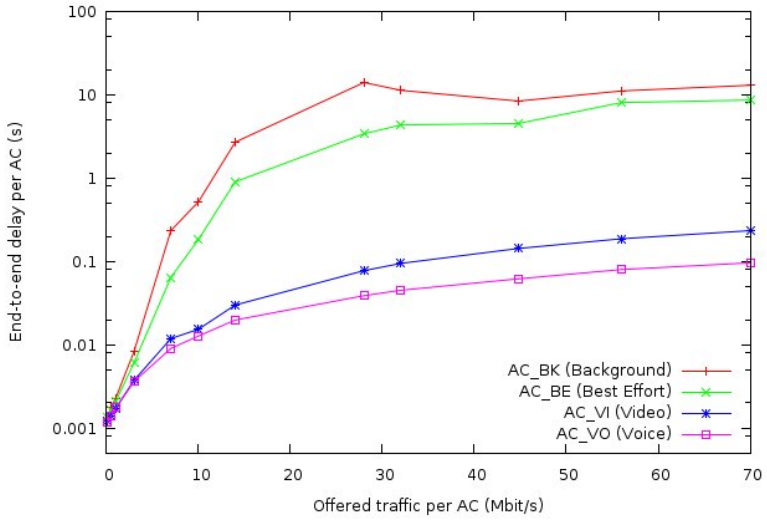

Figura 5.6: Retardo obtenido en cada categoría al aumentar el tráfico (1 salto)

destino *(nótese la escala logarítmica)* llegando incluso a superar los 10 segundos (en ejecuciones de 120).

Por el contrario, aunque el retardo de los paquetes de *Voz* y *Vídeo* también se ve afectado por la saturación de la red, se mantiene en niveles aceptables alcanzando en el caso de *Voz* valores ligeramente inferiores a los 100 ms y en el caso de *Vídeo* valores en torno a los 300 ms.

Dados estos resultados se puede observar como la diferenciación de tráfico es funcional y efectiva tanto en ancho de banda como en retardo.

5.2.2. Comparación con simulaciones previas (Entorno Multisalto con 4 saltos).

En este apartado vamos a comparar los resultados obtenidos con la ejecución de las pruebas con un conjunto de simulaciones previas aparecidas en [9].

Estas pruebas están divididas en dos fases, en primer lugar se estudiará si al aumentar el número de saltos se conserva la efectividad que en un salto nos ofrece IEEE 802.11e en lo referente a diferenciación de tráfico. En la segunda parte se expondrán las pruebas realizadas para verificar la robustez de la calidad de servicio en las categorías de *Voz* y *Vídeo* cuando aumenta el tráfico de *Background* y *Best Effort.*

5.2.2.1. Análisis de diferenciación de tráfico

Cuando en una red la carga es baja no es necesario activar ningún mecanismo de calidad de servicio ya que no son efectivos porque no es necesario priorizar ningún

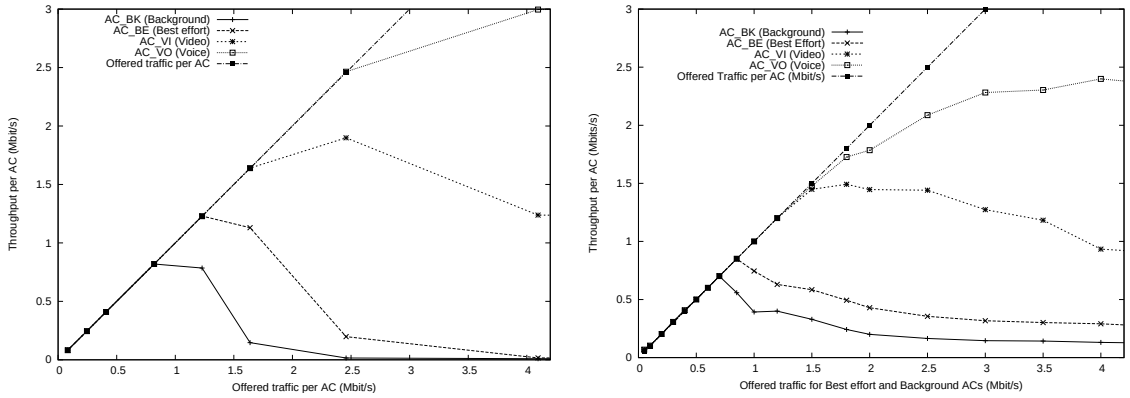

Figure 5.7: Rendimiento alcanzado mediante simulación (izquierda) y en nuestro banco de pruebas (derecha)

tipo de trafico ya que todos reciben un buen servicio. Sin embargo, al aumentar la carga de la red a una media-alta, los mecanismos de calidad de servicio se tornan fundamentales para conseguir diferenciar los diferentes tipos de tráfico. Es por eso que en los siguientes experimentos aumentaremos gradualmente el tráfico para todas las categorías de acceso existentes en IEEE 802.11e.

En la figura 5.7 observamos la eficiencia de IEEE 802.11e en la diferenciación de tráfico a diferentes cargas de la red. En ambos casos se comienza ofreciendo aproximadamente 100 Kbps a cada una de las cuatro categorías de acceso, aumentando progresivamente la cantidad hasta ofrecer a cada categoría de acceso 5 Mbps.

Como se puede ver en las gráficas de la figura 5.7 la diferenciación de tráfico funciona, pero se puede observar que:

1. En un entorno real los valores de rendimiento alcanzados antes de llegar al punto de saturación no siempre son exactamente iguales al rendimiento ofrecido en simulación. Esto es debido a una mayor pérdida de paquetes en un entorno real; este efecto es especialmente visible en las categorías de *Vídeo* y *Voz,* y se asocia normalmente a ruido en el canal provocado por interferencias externas.

2. En el entorno de pruebas el rendimiento para todas las categorías de tráfico decae de una forma más rápida cuando se alcanza el punto de saturación para cada categoría. También se observa que el punto de saturación de cada categoría es sensiblemente menor en el entorno real que en las simulaciones. Esto es debido a que la capacidad del canal en los experimentos es menor debido al solapamiento de la cobertura de todos los dispositivos, así como a la presencia de tramas de control.

3. La efectividad de la diferenciación de tráfico disminuye ya que en el entorno real

las categorías de tráfico de *Best Effort* y *Background* no sufren de inanición, afectando por tanto al tráfico de alta prioridad (*Vídeo* principalmente).

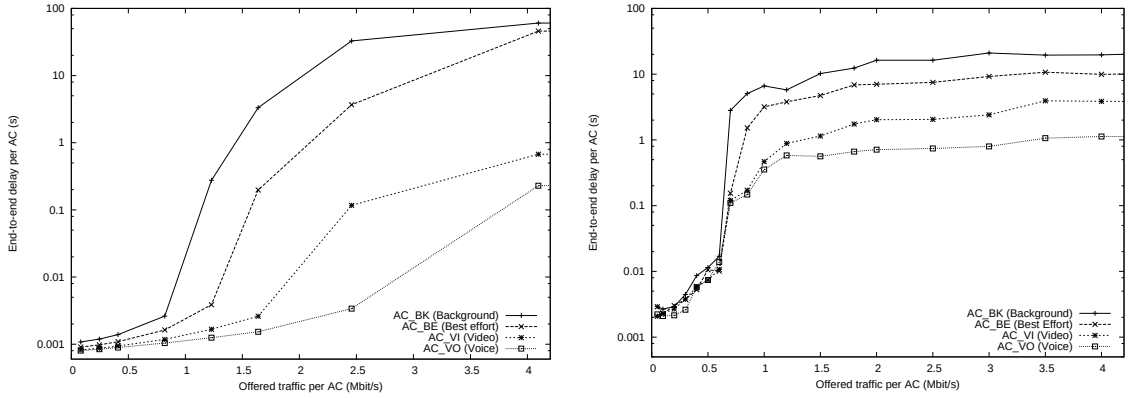

Figure 5.8: Retardo obtenido en las simulaciónes (izquierda) y en nuestro banco de pruebas (derecha)

En la figura 5.8 se encuentran los resultados del retardo para este mismo conjunto de experimentos. Como se puede observar las diferencias son notables entre las dos gráficas.

Principalmente cabe destacar dos puntos:

1. Al contrario que en las simulaciones, en el entorno real el retardo para bajas tasas de bits experimenta una gran crecida (nótese la escala logarítmica del eje Y).

2. Cuando se alcanza el punto de saturación de la red, la diferenciación de tráfico pierde la mayor parte de su efectividad. Particularmente los valores de las categorías de *Voz* y *Vídeo* experimentan una crecida de casi dos ordenes de magnitud.

5.2.2.2. Análisis de la robustez de la QoS

En este segundo conjunto de experimentos intentamos evaluar la capacidad del protocolo IEEE 802.11e para mantener la calidad de servicio de flujos de *Voz* y *Vídeo* cuando se aumenta el tráfico de baja prioridad.

Para conseguir nuestro propósito fijamos dos flujos de tráfico constantes de alta prioridad, el primero de 0.5 Mbit/s sustituiría una posible conversación telefónica realizada a través de VoIP, por lo tanto la categoría de tráfico empleada es la más alta, AC_VO (*Voz*); el segundo flujo de comunicación de aproximadamente 1 Mbit/s se correspondería a una videollamada, de este modo la categoría de este flujo de datos es AC_VI (*Vídeo*). Tanto el tráfico de *Background* como el de *Best Effort* se van incrementando progresivamente.

En la figura 5.9 se muestran los resultados obtenidos en relación al ancho de banda obtenido, tanto para las simulaciones como para el entorno real.

Es evidente que los valores de rendimiento tanto para *Voz* como para *Vídeo* no se mantienen estables, experimentando una pérdida de paquetes cercana al 12 % (11,5 % para Vídeo y 11,8 % para Voz). Sin embargo, en la simulación se puede observar que esto no ocurre, el tráfico de alta prioridad no se ve prácticamente afectado por el de baja prioridad.

Otra notable diferencia es que el tráfico de categoría Best Effort en las simulaciones alcanza los 2,4 Mbit/s (valores no visibles debido a la escala), mientras que en el entorno de pruebas solamente alcanza 1 Mbit/s. Las diferencias observadas se deben principalmente a la menor capacidad del canal en el entorno de pruebas, como ya se explicó en el punto 5.2.2.1.

En cuanto al retardo de este conjunto de pruebas, se puede observar en la figura 5.10 la dificultad de mantener un bajo retardo para las categorías de *Voz* y *Vídeo* en un entorno real. La principal diferencia es la existente cuando se produce un repentino aumento la saturación de la red, aproximadamente a los 0,7 Mbit/s.

En este instante se produce un pronunciado aumento del retraso para todas las categorías de tráfico, siendo este especialmente alto respecto a los resultados obtenidos en la simulación. En este caso el retardo de las categorías prioritarias (*Voz* y *Vídeo*) sufre un aumento de dos órdenes de magnitud en comparación a las simulaciones, mientras que en el caso del tráfico de Background y Best Effort solamente es de un orden de magnitud.

En cifras el retardo obtenido para la categoría de *Vídeo* (460 ms) se encuentra en el límite de lo aceptable para realizar una videoconferencia en tiempo real. Por el contrario, el retardo existente para la categoría de *Voz* (380 ms) es demasiado elevado para poder mantener una conversación telefónica vía VoIP con buenos niveles de calidad.

5.3. Conclusiones

Este apartado se ha centrado en dos puntos: el primero ha sido la validación del correcto funcionamiento del entorno, mientras que en el segundo punto se han presentado los resultados experimentales obtenidos en el entorno de pruebas, comparándolos además con unos resultados previos obtenidos mediante simulaciones en ns-2.

Si nos centramos en el primer punto se puede afirmar que se ha conseguido el objetivo: se ha creado un entorno de pruebas que poseía las características requeridas (ser multisalto y tener la calidad de servicio activada).

En lo referente al segundo punto, tras la presentación de los resultados, se puede observar que, al contrario que en las simulaciones, la efectividad del protocolo IEEE 802.11e se ve mermada en un entorno multisalto, aunque en general en el rendimiento y en las pruebas realizadas se puede observar como se consigue la diferenciación de tráfico de alta y baja prioridad.

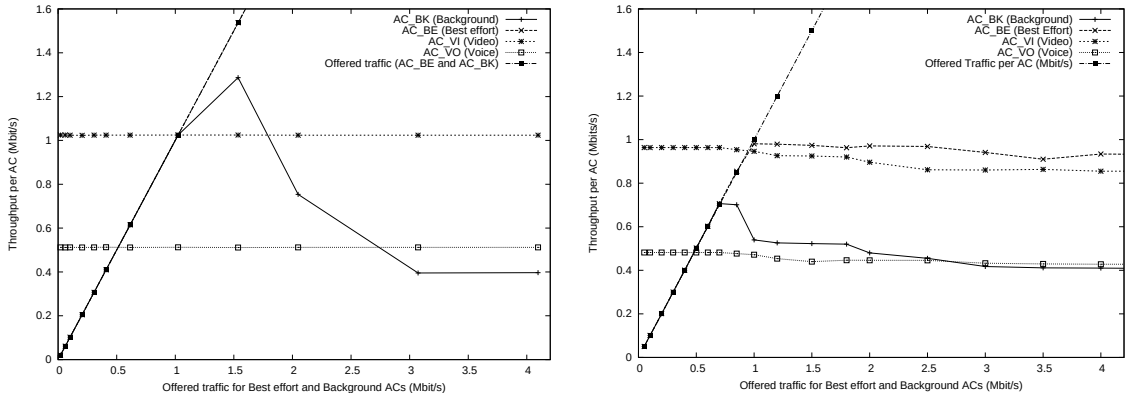

Figure 5.9: Estabilidad del ancho de banda para los flujos de las categorias de Vídeo y Voz cuando se va incrementando el tráfico de Background y Best Effort. Datos de simulación (izquierda) y datos experimentales (derecha)

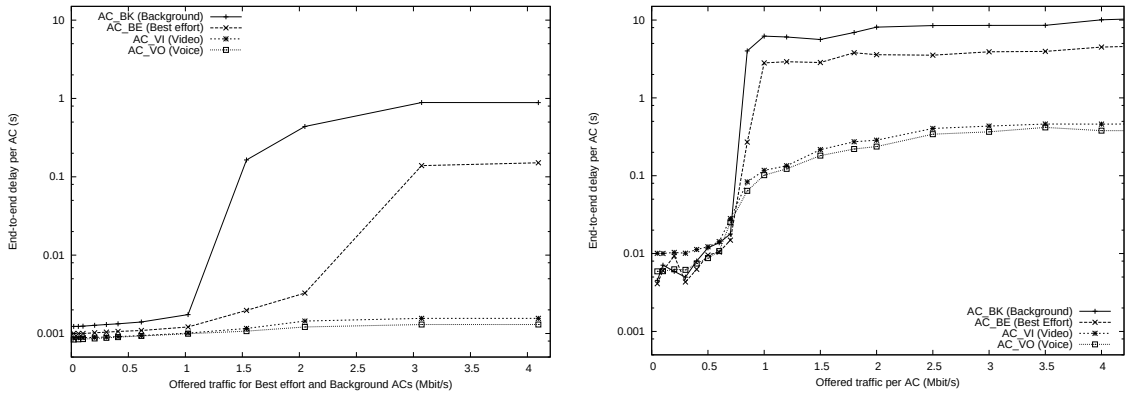

Figure 5.10: Estabilidad del retardo para los flujos de las categorias de Vídeo y Voz cuando se va incrementando el tráfico de Background y Best Effort. Datos de simulación (izquierda) y datos experimentales (derecha)

Capítulo 6

Pruebas en un testbed distribuido

Los resultados del capítulo anterior se efectuaron con todos los nodos concentrados en un espacio muy acotado. Este escenario nos permitió usar IEEE 802.11g fijando la velocidad de transmisión a 54 Mbit/s para conseguir la máxima similitud posible con los resultados de simulación.

En este capítulo nos desharemos de las limitaciones que nos impusimos en el capítulo anterior sacando el máximo rendimiento de las tarjetas inalámbricas; para ello:

1. Habilitamos el modo IEEE 802.11n, aunque se mantiene el ancho de banda del canal en 20 MHz y se usa la banda de 2.4 GHz por limitaciones del hardware.

2. Habilitamos el auto ajuste de velocidad de transmisión de las tarjetas que se emplea por defecto en IEEE 802.11, permitiendo a la interfaz de red seleccionar la mejor velocidad posible de acuerdo a la relación Señal-Ruido (SNR).

El objetivo de estos experimentos era doble: por un lado queríamos evaluar las diferencias de rendimiento en un escenario más realista donde existen mayores distancias entre los nodos, así como barrerás físicas (paredes, puertas, estanterías...) que provocan que el SNR sea mejor que en el escenario anterior. Por otra parte queríamos comparar encaminamiento estático contra encaminamiento dinámico para evaluar cómo la selección y la variabilidad de la ruta puede afectar al rendimiento de IEEE 802.11e.

6.1. Despliegue de los nodos

Para nuestras pruebas se han distribuido ocho portátiles Asus EEEPC por toda una planta de la Universidad obligando a una conexión multi-salto de una manera natural debido a la existencia de obstáculos como pueden ser cristales, paredes, puertas, etc...

En la figura 6.1 se muestra un mapa detallado de la última planta de la Escuela Técnica superior de Ingeniería Informática de la Universidad Politécnica de Valencia. Los puntos oscuros etiquetados con letras representan la localización de los nodos, y las líneas representan los enlaces inalámbricos; sólo los "buenos" enlaces, es decir, enlaces que se mantienen durante bastante tiempo, están representados.

Como se esperaba, los obstáculos físicos se convierten en un serio problema para la propagación de la señal en la banda de 2.4 GHz. Hay que destacar en particular el enlace C-G, que atraviesa una gran distancia en espacio "libre" siendo así el enlace más largo (y también el más débil).

Para estos experimentos se siguió una estrategia similar a la del capítulo anterior, es decir, una red cableada se usa para el control de los nodos. La red cableada es una Fast-Ethernet (100 Mbps) y conecta los nodos con un servidor de Castadiva.

Al generar el tráfico evaluaremos las condiciones de QoS que experimenta un flujo bidireccional que emula una sesión de videoconferencia. Esta videoconferencia se emula generando flujos UDP independientes para poder disfrutar de las ventajas que nos proporciona IEEE 802.11e a nivel MAC al poder separar las categorías de Voz y Vídeo del resto del tráfico. La tasa de bits empleada para emular el vídeo fue de 1 Mbit/s en cada sentido, mientras que para emular la voz se empleó una tasa de 15 Kbit/s en cada sentido.

Además del tráfico con QoS se lanzó un flujo TCP entre ambos nodos como tráfico Best Effort. El cometido de este tráfico es doble :

1. Aumentar el nivel de congestión para alcanzar una situación más realista

2. Gracias a la avaricia de TCP, estimar el ancho de banda residual en la conexión entre los nodos.

Cuando se emplea encaminamiento estático se rellenan las tablas de todos los nodos siguiendo el algoritmo del camino más corto según la topología que se puede ver en la figura 6.1. Las pruebas con encaminamiento dinámico emplean el protocolo OLSR. Los detalles de configuración se pueden ver en la tabla 6.1.

Todos los resultados que se presentan en la siguiente sección son valores medios de diez experimentos, de los cuales se eliminan los dos mejores y los dos peores (efectuando por lo tanto la media sobre los seis restantes). Cada uno de los experimentos duraba 120 segundos.

6.2. Resultados

En las gráficas que se presentarán en este apartado se muestran los siguientes datos:

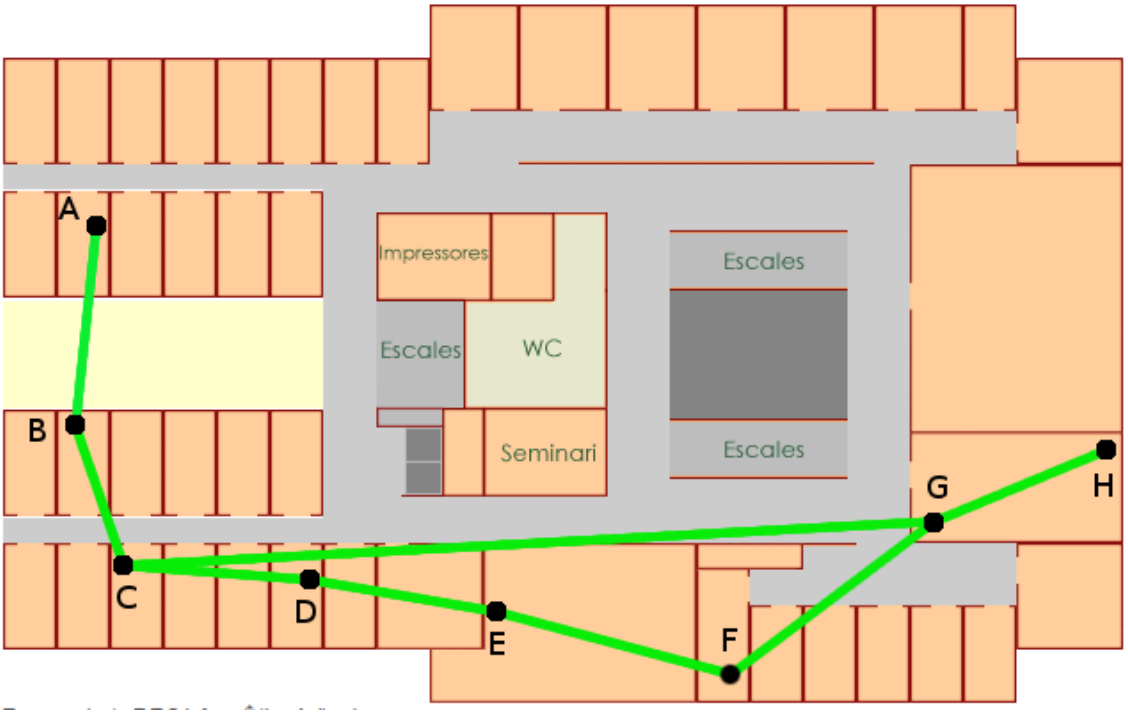

Figura 6.1: Mapa del tercer piso de la Escuela de informática. Dimensiones aproximadas 60 x 40 m.

Parámetro	Valor
HelloInterval	2.0 s
HelloValidityTime	6.0 s
TcInterval	3.5 s
TcValidityTime	10.0 s
TosValue	16

Cuadro 6.1: Detalles de configuración de OLSR.

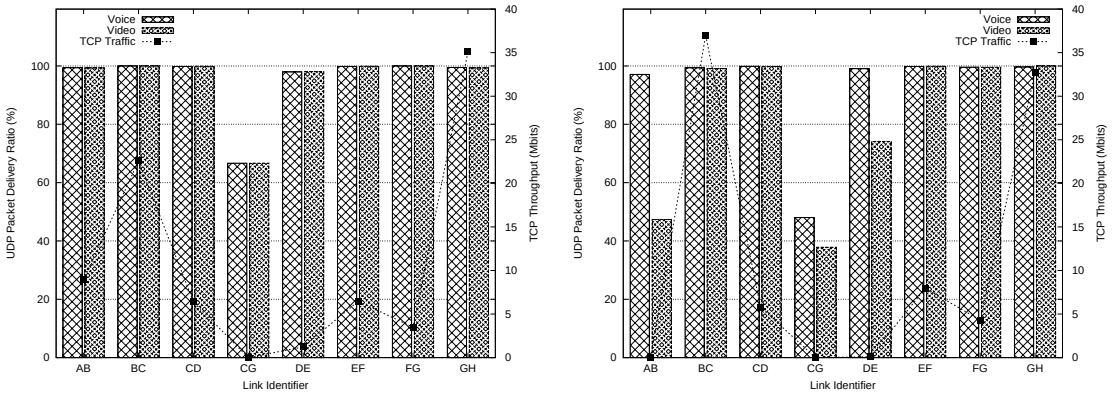

Figura 6.2: Tasa de entrega de los paquetes UDP y ancho de banda del flujo TCP para todas la parejas de nodos a un salto usando encaminamiento estático (izquierda) y dinámico (derecha).

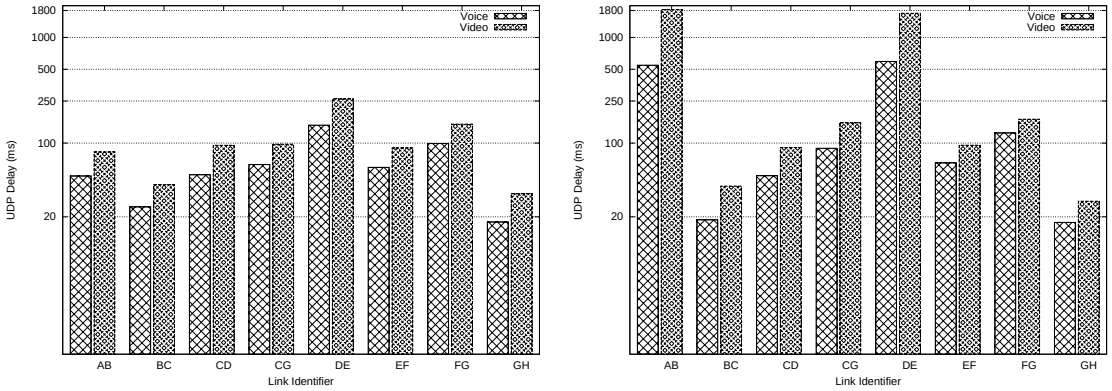

Figura 6.3: Valores del retardo de extremo a extremo para todas las parejas de nodos a un salto usando encaminamiento estático (izquierda) y dinámico (derecha).

- Para el tráfico UDP se ha obtenido la tasa de entrega de paquetes (%) y el retardo extremo a extremo (ms).
- Para el tráfico TCP se ha obtenido el ancho de banda medio obtenido.

En la figura 6.2 se muestran los resultados obtenidos a un salto. Se puede observar que el enlace C-G es el enlace más débil debido a la gran distancia existente entre las dos estaciones. Por el contrario, G-H y B-C son los enlaces que consiguen un mejor rendimiento. Se puede observar en ambas gráficas que la tasa de entrega de los paquetes UDP es cercana al 100 %, mientras que el tráfico TCP también consigue un importante ancho de banda (hasta 37 Mbit/s en el enlace B-C).

Si se compara el encaminamiento estático con el encaminamiento dinámico se pueden apreciar diferencias significativas en los enlaces A-B, C-G y D-E, especialmente en términos de tráfico de vídeo. Dado que la tasa de entrega del tráfico de voz se mantiene constante, la pérdida no se puede achacar a pérdidas de la ruta, sino que

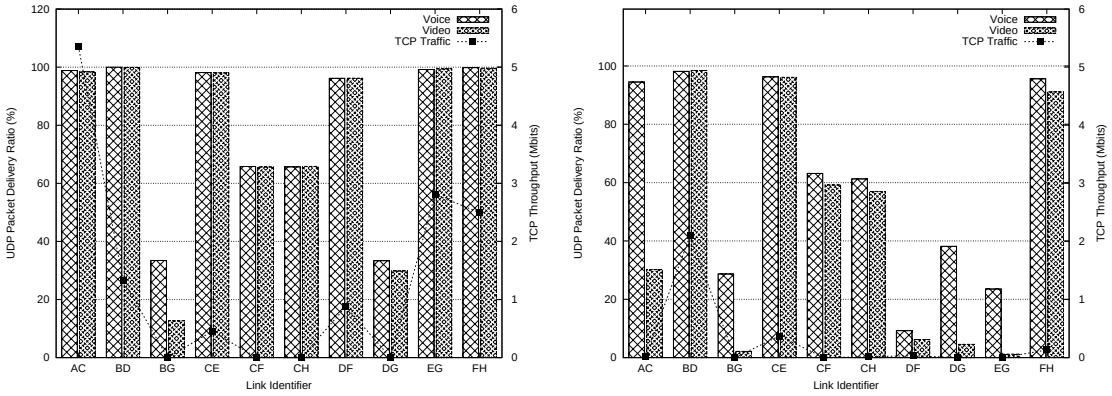

Figura 6.4: Tasa de entrega de los paquetes UDP y ancho de banda del flujo TCP para todas la parejas de nodos a dos saltos usando encaminamiento estático (izquierda) y dinámico (derecha).

deben de estar producidas por una mayor contención en el canal por el incremento del tráfico de control. Además, dado que la mayoría de paquetes que emplea OLSR son broadcast, el estándar IEEE 802.11 indica que la transmisión se debe efectuar a la mínima velocidad, por lo que el medio se mantiene ocupado durante mucho más tiempo.

Los resultados del retardo (figura 6.3) evidencian la capacidad de IEEE 802.11e de priorizar el tráfico de Voz respecto al de Vídeo, ya que en todos los casos el retardo experimentado por el tráfico de Voz es menor. También se observa la proporcionalidad existente entre la cantidad de tráfico de Background de TCP y el retardo que experimentan los flujos de voz y vídeo. Tal y como se esperaba, a mayor tráfico TCP menor retardo. Esto es así porque una mayor saturación del canal suele estar asociado con retardos mayores. La excepción se encuentra en el enlace C-G, donde se detectan retardos muy buenos a pesar de la gran pérdida de paquetes. Un análisis en profundidad reveló que la gran tasa de pérdidas de paquetes es debida a desincronizaciones entre los nodos C y G durante la duración de las pruebas.

En la figura 6.4 se pueden observar los resultados a dos saltos. En los enlaces BG, CF, CH y DG se puede observar un rendimiento mucho peor que en el resto. Esto es así porque todos ellos hacen uso del enlace C-G para alcanzar su destino que, como se vio anteriormente, era el que peores resultados obtenía. Las conexiones restantes mantienen un rendimiento razonablemente bueno aunque, al existir una mayor contención en el canal, se producen más pérdidas en el tráfico con QoS. Un ejemplo de esto se puede observar en el enlace A-C donde a pesar de disponer de 5,3 Mbit/s de ancho de banda para TCP, se pierden entre un 1 y un 2 % de los paquetes de alta prioridad.

Al analizar el escenario dinámico se vuelve a observar que el rendimiento está degradado en comparación con el escenario estático. Hay que tener en cuenta que

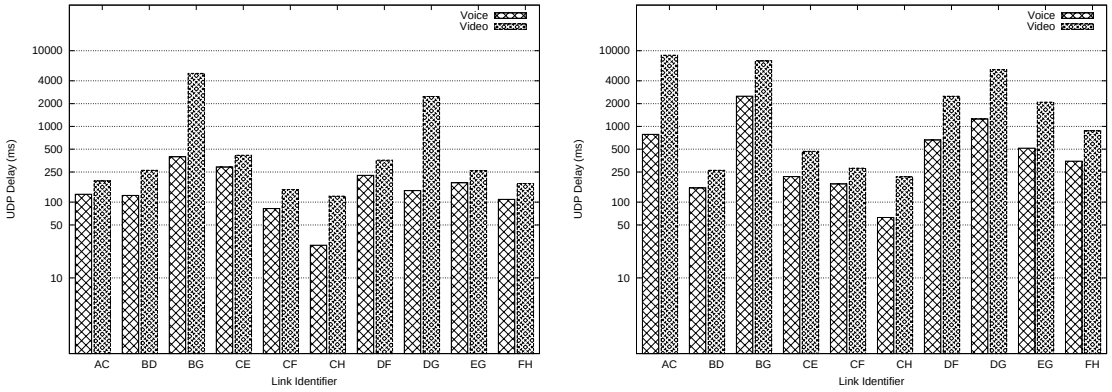

Figura 6.5: Valores del retardo de extremo a extremo para todas las parejas de nodos a dos saltos usando encaminamiento estático (izquierda) y dinámico (derecha).

OLSR está continuamente actualizando rutas, por lo que es posible que la ruta usada en algunas de las pruebas difiera de las pruebas realizadas en estático. Si se analiza en detalle la figura 6.4 se observan dos cosas: en la mayoría de los casos se observa un comportamiento similar al entorno estático, con la diferencia de que el rendimiento es ligeramente inferior para el tráfico de voz y significativamente menor en el caso de vídeo. Sin embargo, hay un par de casos (conexiones D-F y E-G) en los que el protocolo de encaminamiento (OLSR) no es capaz de mantener la ruta durante el test completo, causando que el rendimiento sea mucho peor que usando el encaminamiento estático.

Los valores de retardo para dos saltos que se pueden observar en la figura 6.5 muestran que, al igual que en los resultados a un salto, el encaminamiento dinámico produce resultados mucho peores que al usar encaminamiento estático (nótese la escala logarítmica de las gráficas). También se verifica que ya no existe una relación entre la tasa de entrega de paquetes UDP, el rendimiento de TCP y los valores del retardo. Esto es particularmente visible en el escenario de enrutamiento dinámico donde las conexiones C-F y C-H muestran pequeñas diferencias en términos de pérdida de paquetes UDP y, sin embargo, sí que presentan diferencias significativas para el retardo (especialmente en voz).

Una degradación similar se puede observar en los casos de tres y cuatro saltos (figuras 6.6 y 6.8). Es interesante ver como las conexiones que no emplean el enlace más débil (C-G) son capaces de mantener las pérdidas de voz y de vídeo al mínimo teniendo una tasa de entrega cercana al 100 %. Para confirmar esto simplemente hay que echar un vistazo a las conexiones A-D, B-E, y E-H si hablamos de tres saltos y el enlace A-E para el de cuatro saltos. Esos resultados nos muestran claramente que IEEE 802.11e es capaz de mantener la QoS a niveles altos aunque se trabaje en un entorno multisalto, pero sólo si ninguno de los enlaces que atraviesa es dé-

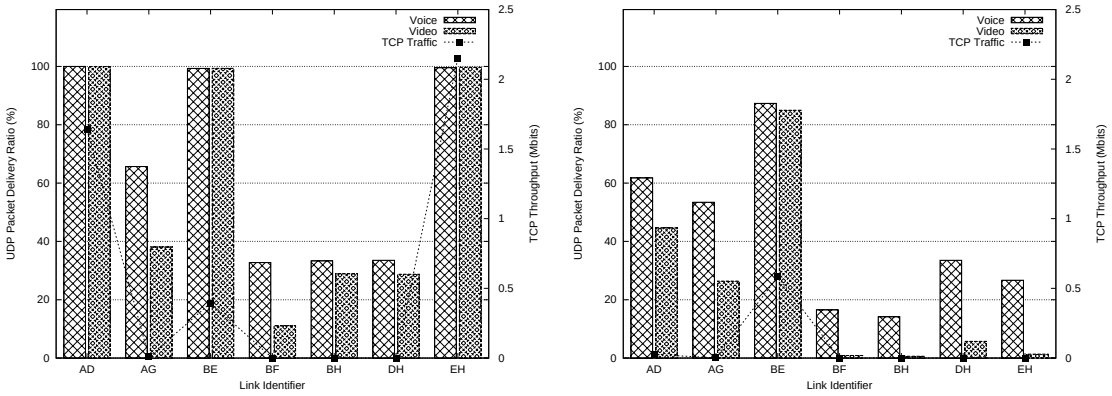

Figura 6.6: Tasa de entrega de los paquetes UDP y ancho de banda del flujo TCP para todas la parejas de nodos a tres saltos usando encaminamiento estático (izquierda) y dinámico (derecha).

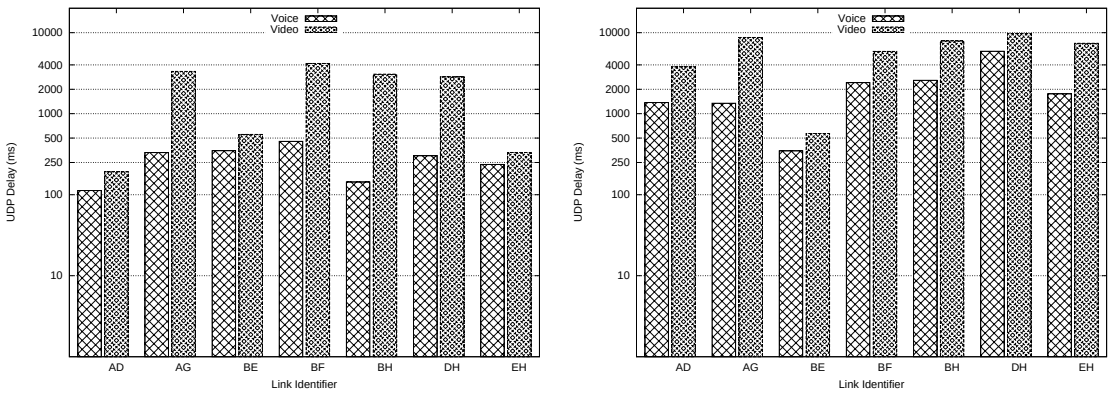

Figura 6.7: Valores del retardo de extremo a extremo para todas las parejas de nodos a tres saltos usando encaminamiento estático (izquierda) y dinámico (derecha).

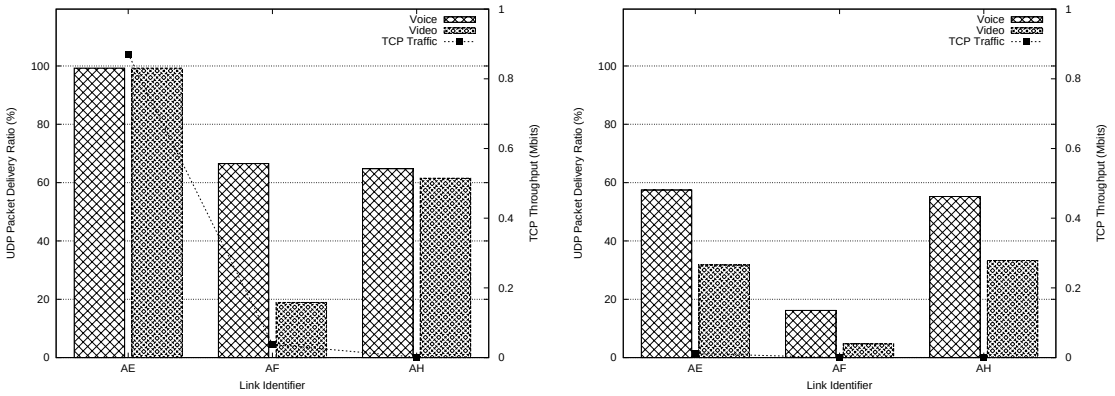

Figura 6.8: Tasa de entrega de los paquetes UDP y ancho de banda del flujo TCP para todas la parejas de nodos a cuatro saltos usando encaminamiento estático (izquierda) y dinámico (derecha).

bil, es decir, siempre que la tasa de transferencia empleada en esos enlaces sea lo suficientemente alta para no convertirse en un cuello de botella. De otra forma el rendimiento decrecerá hasta niveles inaceptables, haciendo que el tráfico con QoS tenga una gran cantidad de pérdidas.

Centrándonos en las diferencias existentes entre encaminamiento estático y dinámico para los casos de tres y cuatro saltos, encontramos que conforme aumenta el número de saltos la inestabilidad en la ruta tiende a incrementarse; concretamente, los resultados obtenidos muestran que la tasa de entrega de paquetes es siempre inferior al 90 % cuando se transmite a tres saltos, e inferior al 60 % si se transmite a cuatro saltos.

El tráfico Best Effort (TCP) también es gravemente afectado por la inestabilidad de la ruta sufriendo de inanición (12 Kbit/s en el mejor caso) cuando se transmite a cuatro saltos.

En lo referente a los resultados del retardo, las figuras 6.7 y 6.9 muestran que el rendimiento de QoS alcanza niveles críticos. De hecho, si el tráfico tiene requisitos de tiempo real, los flujos de vídeo sufren un retardo que puede ser considerado demasiado alto (más de 250 ms). En cuatro saltos ninguna de las dos soluciones ofrece unos niveles de QoS aceptables ni para voz ni para vídeo, al contrario de lo que ocurre en simulación. Si se compara el encaminamiento estático con el dinámico existe una gran pérdida de prestaciones del encaminamiento dinámico, especialmente en tres saltos.

Todos los resultados presentados antes muestran que los protocolos de encaminamiento para MANETs todavía requieren de mejoras importantes para conseguir la estabilidad y eficiencia que se requieren para operar con flujos de tiempo real.

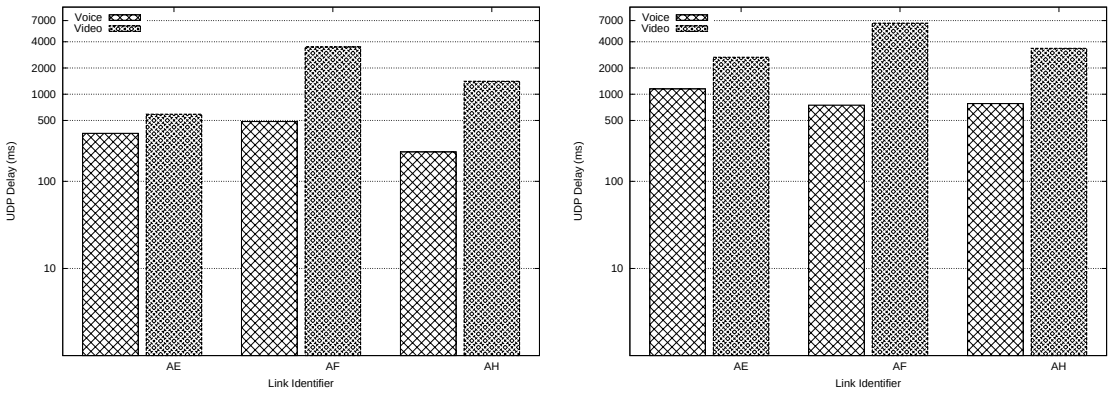

Figura 6.9: Valores del retardo de extremo a extremo para todas las parejas de nodos a cuatro saltos usando encaminamiento estático (izquierda) y dinámico (derecha).

6.3. Conclusiones

Tras desplegar y conseguir hacer funcionar correctamente un testbed de una red ad hoc multisalto en una planta del edificio de la Universidad, los resultados experimentales muestran que IEEE 802.11e es capaz de mantener su eficiencia incluso en un entorno multisalto. De hecho, es capaz de mantener un bitrate adecuado incluso a cuatro saltos siempre y cuando los enlaces por los que se encaminen los paquetes sean buenos enlaces. Sin embargo, en lo referente al retardo, se ha constatado que para comunicaciones a más de tres saltos se produce un retardo excesivo para la correcta realización de estas comunicaciones.

Las pruebas realizadas también muestran que, comparado con el encaminamiento estático, el encaminamiento dinámico (en este caso OLSR) es propenso a causar conectividad intermitente a pesar de no haber movilidad. Estos problemas de encaminamiento afectan tanto al tráfico con QoS como al que no tiene QoS, provocando una gran pérdida de paquetes y un gran aumento del retardo.

Consideramos que, para conseguir buenos niveles de QoS al desplegar un entorno MANET basado en IEEE 802.11e, todavía queda mucho trabajo por hacer. Los puntos más críticos pasan por el desarrollo de un algoritmo de encaminamiento con conocimiento de QoS y de estado de las rutas, de un control de admisión, y la optimización de los drivers de IEEE 802.11 para un buen funcionamiento en modo ad-hoc.

Capítulo 7

Desarrollo de un sistema de control de admisión distribuido para MANETs

Los mecanismos de reserva de un canal de comunicaciones han existido desde el invento de la telefonía. Al principio esta reserva se realizaba de manera manual, pero ahora con la evolución de la tecnología se realiza de manera automática. Estos mecanismos de reserva son esenciales para conseguir una comunicación provechosa por parte de los usuarios que en ella participan, aportando garantías de calidad de servicio a esa comunicación.

En una MANET, conseguir que un determinado flujo tenga unas garantías de calidad de servicio determinadas es un problema complejo debido, entre otras cosas, a los recursos limitados del canal, a la movilidad de los nodos, a la posible existencia de varias rutas, así como a la baja potencia computacional que suelen tener los dispositivos móviles.

En este capítulo presentaremos la implementación de un sistema liviano y que no impone ningún tipo de restricción a los nodos intermedios para conseguir ciertas garantías de QoS en los flujos que seleccione el usuario.

Este sistema, denominado DACME [8] (*Distributed Admission Control for MANET Environments*), se presentará en este capítulo comenzando por una corta introducción teórica para poder entender el funcionamiento del sistema. En segundo lugar se presenta el análisis del problema que se ha realizado de cara a su implementación. En el tercer punto de este capítulo se expondrá el diseño que se ha seguido al realizar la implementación de DACME. En el último punto de este capítulo se mostrarán varios resultados en los que se puede apreciar el correcto funcionamiento de la aplicación así como las posibles mejoras que se obtienen en las comunicaciones al usar este sistema.

7.1. Fundamentos teóricos

DACME es un sistema de control de admisión basado en el paradigma prueba-respuesta que realiza mediciones de calidad de servicio entre los nodos finales de la comunicación. El propósito de este sistema es estimar si la calidad ofrecida se ajusta a los requisitos de QoS de los flujos multimedia.

Para que DACME opere en condiciones óptimas, todas las interfaces de red deben de tener habilitado el soporte para IEEE 802.11e, aunque no es un requisito fundamental ya que DACME funcionará correctamente sobre redes sin QoS.

Las restricciones que impone DACME sobre los nodos de una MANET son mínimas: solamente los nodos fuente y destino de la comunicación han de tener un agente DACME funcionando. El resto de los nodos simplemente deben encaminar los paquetes de DACME como cualquier otro tipo de paquetes.

Una aplicación que desee beneficiarse de las posibilidades que ofrece DACME tiene que registrarse en el agente DACME indicando las características del flujo: puertos fuente y destino, IP destino, así como los parámetros de QoS requeridos (ancho de banda, retardo...).

Cuando el registro se completa, DACME se encarga de realizar pruebas entre la fuente y el destino con el propósito de comprobar si las características demandadas se cumplen. Estas mediciones se repiten periódicamente para activar o bloquear los flujos registrados.

El agente destino, cuando recibe los paquetes de prueba, va actualizando las estadísticas de ese flujo y, cuando se reciben todas las pruebas, o se excede un tiempo límite se envía una contestación que la fuente, a su vez, deberá procesar.

En la figura 7.1 se puede observar cómo se estructura un agente DACME de una manera funcional. Dispone de dos bloques principales, “Packet filter” y “QoS measurement module”.

Packet filter es el encargado de lidiar con el nivel IP para filtrar los paquetes de los flujos suscritos en el agente DACME.

Por otra parte, *QoS measurement module* es el encargado de comunicarse con otros agentes DACME para realizar las mediciones de cumplimiento de la QoS requerida por cada flujo, por esto, emplea tanto el nivel de transporte UDP (para enviar mensajes a otros agentes DACME) y el nivel IP (para recolectar información de la ruta, TTL, ...). También es el encargado de comunicarse con el módulo *Packet filter* para que este bloquee los flujos que no cumplen los requisitos de QoS y además es el módulo que se encarga de mantener actualizadas las estadísticas de cada uno de los flujos suscritos.

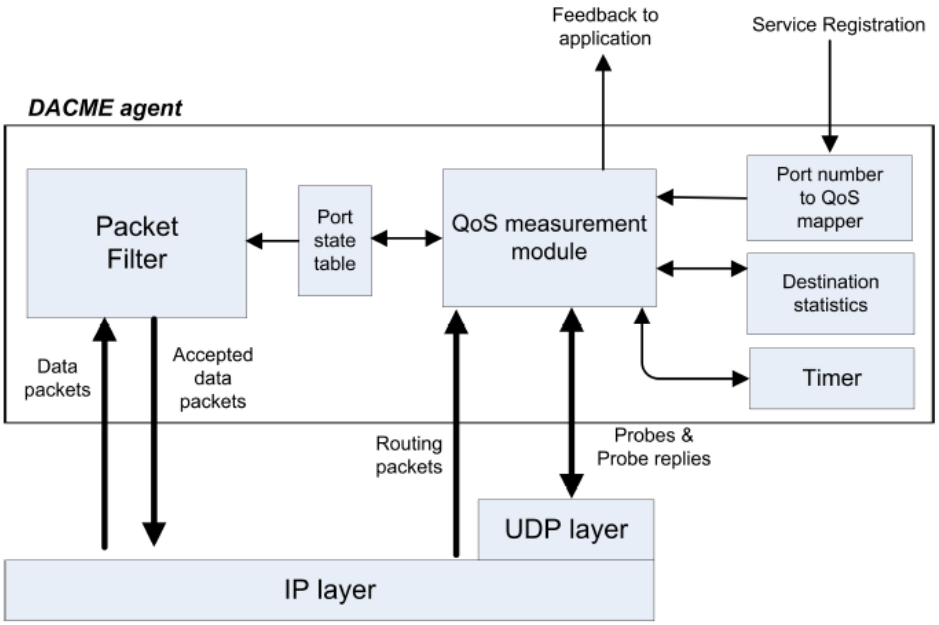

Figura 7.1: Diagrama de bloques de un agente DACME

7.1.1. Soporte para aplicaciones con requerimientos de ancho de banda

Uno de los principales requisitos de QoS es conseguir un ancho de banda mínimo garantizado para un determinado flujo de datos. Para conseguir saber si la red está preparada para esa transmisión, DACME realiza mediciones de ancho de banda extremo-a-extremo de forma periódica.

Estas mediciones consisten en la realización de varios ciclos de "Prueba - Respuesta". Cada uno de estos ciclos consiste en el envío de 10 paquetes de prueba al destino, el cual responde cuando recibe los 10 en su totalidad, o si pasado un tiempo (que se calcula en función de la situación de la red; ver punto 7.1.1.1), se han recibido por lo menos 6 paquetes. El procedimiento a seguir por el destino consiste en enviar un paquete de respuesta al agente fuente.

Los paquetes en la fuente se deben generar y enviar en ráfaga, intentando que el tiempo entre el primero y el último tienda a cero, tal y como se puede observar en la figura 7.2. De esta manera los paquetes llegan al destino con una determinada separación que nos permitirá calcular el ancho de banda disponible.

Cuando el agente DACME destino reciba un paquete de prueba debe obtener una medida del ancho de banda disponible estimado y devolvérselo a la fuente. El ancho de banda se obtiene mediante esta expresión:

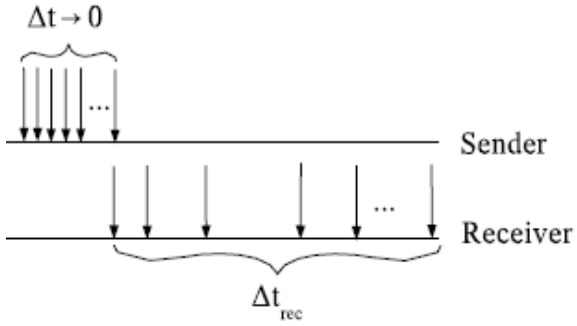

Figura 7.2: Esquema de una prueba en una medición de ancho de banda

Figura 7.3: Mecanismo de admisión para flujos con restricciones de ancho de banda

Tras recibir una respuesta de ancho de banda **HACER** **{**
corregir el ancho de banda estimado con todos los valores disponibles
SI (existe un nivel de confianza del 95 % que el ancho de banda disponible es mayor que el solicitado)
ENTONCES aceptar por ancho de banda
SINO
SI (existe un nivel de confianza del 95 % que el ancho de banda disponibles es mejor que el solicitado)
ENTONCES rechazar flujo
SINO
SI (el número de pruebas usadas es menor que el máximo disponible)
ENTONCES enviar una nueva prueba
SINO
mantener la aceptación/denegación por ancho de banda **}**

$$B_{medido} = \frac{8 \cdot T_{paquete}}{AIT}$$

Donde B_{medido} será el ancho de banda estimado medido en bits/segundo, $T_{paquete}$ es el tamaño en bytes del paquete de prueba recibido, y AIT se refiere al tiempo medio de llegada entre paquetes (*Average Inter-arrival Time*) cuya fórmula es la siguiente:

$$AIT = \frac{\triangle t_{rec}}{N-1}$$

En esta fórmula $\triangle t_{rec}$ hace referencia al tiempo entre el primer y el último paquete recibido (tal y como se puede observar en el figura 7.2), y N es el número de paquetes recibidos (no el número de paquetes enviados).

Todo este proceso se puede repetir hasta 5 veces para conseguir una mayor precisión en la medición.

En la figura 7.3 se puede observar el algoritmo que sigue el agente DACME cuando recibe las respuestas a las pruebas enviadas. Como se puede apreciar, este algoritmo solamente hace referencia a mediciones de ancho de banda. Si el flujo requerido tiene otras restricciones, se deberán ejecutar posteriormente otros algoritmos para los demás parámetros.

Este algoritmo permite reducir a dos el número de pruebas que se requieren para tomar una decisión en los casos en que se hace evidente rápidamente que el ancho de banda existente es mucho más alto o más bajo que el demandado.

Se ha de tener en cuenta que DACME reserva un pequeño ancho de banda para evitar saturar la red al realizar las mediciones; además, con esto se evita que la QoS ofrecida varíe de forma brusca, consiguiéndose así lograr una mayor estabilidad para los flujos activos.

Para evitar problemas con los flujos existentes, en el momento de la medición todas las pruebas se realizan asignando la categoría de *Vídeo* a los paquetes que forman parte de la prueba. Además, estos paquetes tienen el tamaño especificado para cada flujo.

7.1.1.1. Temporizadores

Cuando se diseña un algoritmo para un entorno de red en el que se pueden perder paquetes fácilmente se deben tener en cuenta estas pérdidas y actuar de una manera consecuente y correcta. En DACME se utilizan varios temporizadores para conseguir hacerse cargo de estas pérdidas.

Cada agente fuente mantiene un temporizador para ser capaz de reaccionar en el supuesto de que una contestación de otro agente nunca se reciba. Así siendo, tras enviar todas los paquetes de prueba, se pone en marcha un temporizador de 500 ms. Si no se recibe respuesta, y por tanto el temporizador se dispara, o si se recibe la respuesta, se programa un nuevo bloque de pruebas para dentro de 3 segundos.

De la misma manera, los agentes destino deben de tener en cuenta que es posible que también se pierdan los paquetes de prueba. De esta manera, cuando el destino recibe dos o más paquetes, va actualizando constantemente un temporizador interno de acuerdo con la siguiente fórmula:

$$T = \frac{T_{ultimo} - T_{primero}}{N_{recibidos} - 1} \cdot (N_{faltan} + \varepsilon) + \tau$$

En esta ecuación T_{ultimo} y $T_{primero}$ son, respectivamente, los tiempos del primer y el último paquete que ha llegado, $N_{recibidos}$ es el número de paquetes recibidos en ese instante, N_{faltan} es el número de paquetes que falta por recibir, ε es un número fijo de paquetes para dar cierto margen de tolerancia temporal, y τ es una pequeña

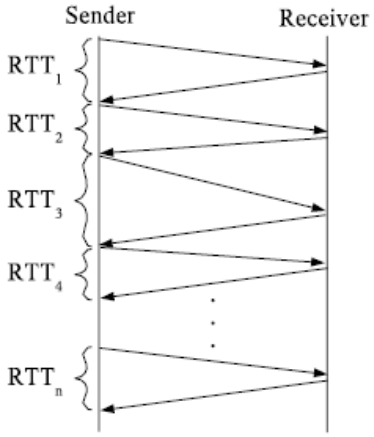

Figura 7.4: Esquema de una prueba en una medición de retardo

constante de tiempo (50 ms) para tener en cuenta los posibles cambios de ruta.

El número mínimo de paquetes que se tienen que recibir para considerar buena una medición es 6 (la mitad de los paquetes que se envían, más uno).

7.1.2. Soporte para aplicaciones con requerimientos de retardo

Cuando una aplicación demanda ciertas características de ancho de banda y retardo (o simplemente de retardo), DACME emplea otra técnica diferente para realizar estas mediciones.

La técnica empleada por DACME para la medición del retardo en la ruta de los paquetes es similar a la empleada por las aplicaciones de tipo ping, con la diferencia de que los paquetes de *petición* son enviados inmediatamente después de recibir los de *respuesta* con intención de reducir, en la medida de lo posible, el tiempo necesario para la prueba. Una representación gráfica de su modo de funcionamiento se puede ver en la figura 7.4.

Todos los paquetes se envían con la categoría demandada y con el tamaño requerido en cada flujo.

Se necesitan cuatro rondas de medición (*petición* - *respuesta*) para conseguir un resultado fiable. En la primera de ellas el valor obtenido se descarta, ya que se utiliza para activar, en caso de que sea necesario, los protocolos de encaminamiento. Además, si por alguna razón alguno de los dos paquetes se perdiese, se considera que no existe camino entre los dos nodos que se quieren comunicar, así que como resultado se bloquea el flujo. Con el resto de las mediciones se obtienen el mejor y el peor caso que nos servirán para aceptar/denegar el flujo.

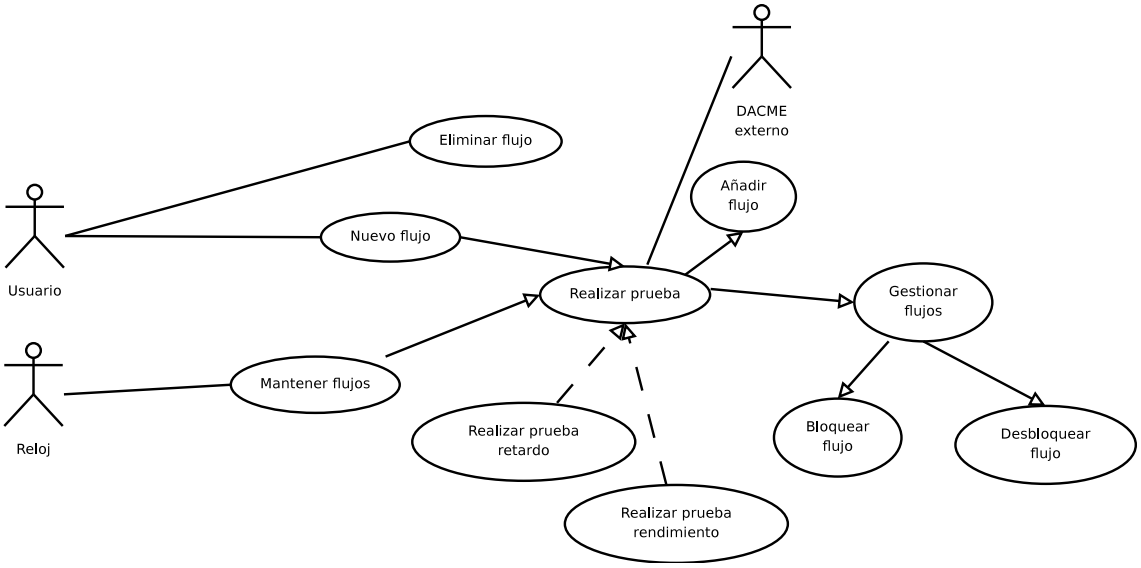

Figura 7.5: Diagrama de casos de uso de DACME

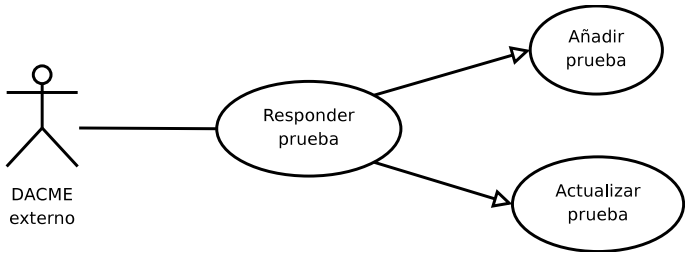

Figura 7.6: Diagrama de casos de uso de DACME

7.2. Análisis del sistema

En el entorno de DACME hay tres actores principales que interactúan con nuestro sistema: en primer lugar los usuarios son aquellos que pueden demandar a DACME dar de alta nuevos flujos o dar de baja flujos ya existentes; el segundo actor que aparece es el reloj que se encargará de activar las tareas de mantenimiento de flujos periódicas, y por último aparece la figura de otro actor, *DACME externo*, que hace referencia a agentes DACME existentes en otras máquinas y con los que se realizan las pruebas de ancho de banda y retardo.

La mayoría de casos de uso que aparecen en las figuras 7.5 y 7.6 son auto explicativos, siendo que el único que tiene más dificultad es "*Realizar prueba*", que detallaremos a continuación.

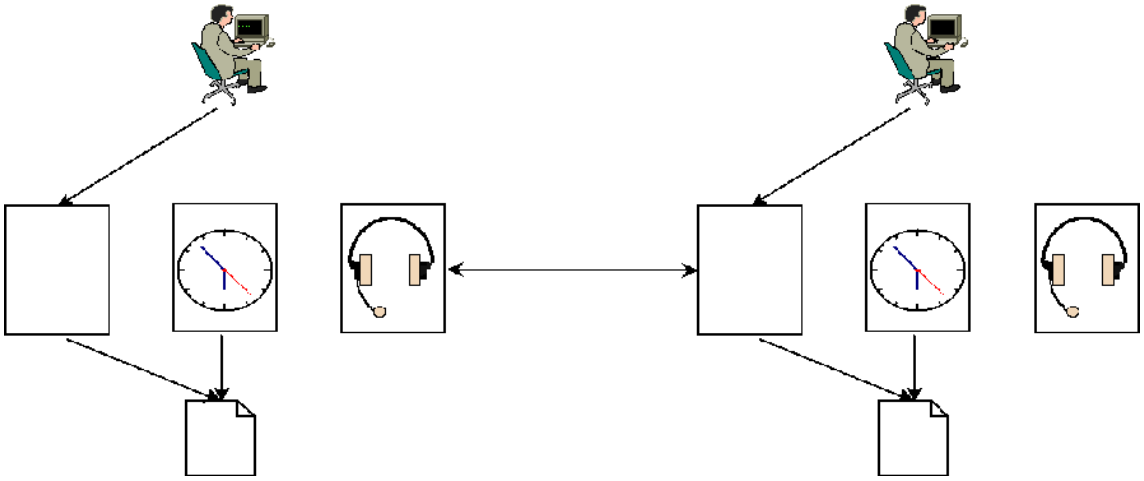

Figura 7.7: Esquematización del funcionamiento de DACME

Realizar prueba

Llamado por Nuevo flujo, mantener flujos.

Llama a Añadir flujos, Gestionar flujos.

Descripción

1. Llegada de la llamada de testeo de flujo con las características demandadas.
2. Realización del test de retardo.
3. Realización del test de rendimiento.
4. Llamada a "*Añadir Flujo*" con las características demandadas.
5. **SI** se ha añadido correctamente el flujo
   a)Llamada a "Gestionar flujos" con los resultados de los tests realizados.

7.3. Diseño de DACME

Como bien se puede observar en la parte de análisis, DACME tiene dos partes claramente diferenciadas: en primer lugar se encuentra la parte encargada de la realización de las pruebas, y en segundo lugar la parte de la recepción de las pruebas desde otros agentes.

Además, dentro de la parte de realización de pruebas, hay dos partes independientes entre si pero que comparten datos comunes: una de las partes es la encargada de la recepción de peticiones por parte de los usuarios, y la otra es la encargada de la activación periódica de los tests de los flujos existentes.

Por todo esto se decidió que el agente DACME dispondría de tres hilos de ejecución, de los cuales solamente dos de ellos tienen acceso a la lista de flujos.

Todo esto se puede observar más claramente en la figura 7.7, donde se ve claramente la comunicación existente entre dos agentes DACME, así como la división existente en los tres hilos de ejecución.

Operation type	IP	Send Port	Destination Port	Category	Bitrate (kbps)	Packet Size	Delay	Jitter

Figura 7.8: Formato del mensaje de comunicación entre los clientes y DACME

7.3.1. Tipos de datos

Para la realización de DACME se definieron varios tipos nuevos de datos, así como diversas listas para poder gestionarlos correctamente y evitar problemas de concurrencia entre los hilos que acceden a los datos.

Dacme flow Es una estructura de datos que agrupa las características de un único flujo. Se agrupan en una lista, y la concurrencia entre hilos se controla al utilizar las funciones específicas para el manejo de esta lista.

Dacme probe Es una estructura de datos que almacena la información asociada a un conjunto de paquetes "probe" de DACME. Aquí se almacenan cosas como número de secuencia, número de "probes" del mismo conjunto recibidos y tiempo de caducidad del conjunto de "probes".

7.3.2. Mensajes DACME

Para la comunicación con otros clientes DACME o con las aplicaciones que demandan o cancelan los flujos se han definido unos mensajes estándar que se proceden a detallar:

7.3.2.1. Mensajes cliente - DACME

Estos mensajes, que emplean el protocolo TCP son con los que se notifica a DACME de que queremos crear un nuevo flujo, dar de baja un flujo creado previamente, o conseguir un listado de todos los flujos que la instancia de DACME contactada posee y en que estado se encuentra (bloqueado o sin bloquear).

Para una mayor facilidad en la comunicación entre diferentes implementaciones de los clientes (diferentes lenguajes de programación) toda la comunicación se realiza en ASCII, facilitando además la tarea de depuración de los mensajes. El tamaño de estos mensajes no es un problema ya que la comunicación (una petición de alta/baja de un flujo de datos) se realiza normalmente entre procesos de la misma máquina.

En la figura 7.8 se puede observar el tipo de mensaje empleado.

7.3.2.2. Mensajes DACME - DACME

Los mensajes de comunicación entre clientes DACME son los que sirven para realizar las mediciones de ancho de banda, retardo y jitter, por lo que estos mensajes se envían por UDP.

Figura 7.9: Formato de los mensajes PROBE de DACME

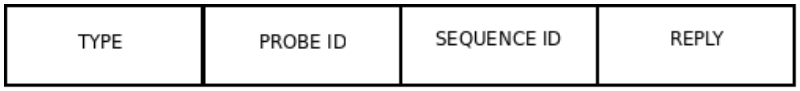

Figura 7.10: Formato de los mensajes RESPONSE de DACME

Dado que la implementación de DACME es en C/C++, la comunicación no se realiza con otros clientes desarrollados en otro lenguaje de programación y en este caso el tamaño del mensaje es crítico para intentar saturar lo menos posible la red, el envío de los datos se realiza en binario.

En la figura 7.9 se puede observar un mensaje de **PROBE**, es decir, los mensajes que envía DACME a otro proceso DACME para que se puedan realizar las mediciones correspondientes.

El primer campo del mensaje es el tipo, es decir, si es un PROBE o un RESPONSE, el segundo campo "*mode*" indica el tipo de mensaje de prueba para poder saber si es un mensaje de medición de ancho de banda, de retardo o de jitter. Los campos "Probe ID", "Packet ID" y "Sequence ID" son campos de control para poder contestar correctamente y llevar las estadísticas necesarias.

En "Packet size" se almacena el tamaño total del mensaje, y por último en "payload" se introducen bytes aleatorios para que el mensaje tenga el tamaño correspondiente.

En la figura 7.10 se puede observar un mensaje de **RESPONSE**, estos mensajes son los que envía un proceso DACME a otro para comunicar el resultado de las pruebas realizadas al recibir uno o varios mensajes **PROBE**.

La estructura es muy parecida a la de los mensajes PROBE, sólo que ahora no necesitamos tanta información. En este caso "type" será para marcar que es un paquete RESPONSE, "Probe ID" y "Sequence ID" son campos de control que nos marcan a qué prueba estamos contestando y por último en "Reply" se almacena la respuesta numérica que esperamos (ya sea ancho de banda, retardo o jitter estimados).

7.4. Gestión de flujos

DACME no gestiona directamente los flujos que se registran en él, sino que delega las funciones de filtrado de paquetes y aplicación de la categoría de QoS en un nivel inferior.

En este caso el nivel inferior empleado para este filtrado ha sido el uso de la

herramienta "*iptables*". Como se pudo apreciar en el capítulo 2, la asignación de categorías de QoS a un flujo determinado es una tarea trivial; por ello, en esta implementación de DACME se ha creado un módulo encargado de formar y ejecutar órdenes iptables para la aplicación de QoS a estos flujos.

Además, si no se cumplen las condiciones de servicio exigidas, DACME debe bloquear el tráfico saliente y también para esta tarea hace uso de iptables.

7.5. Validación de la implementación

Para comprobar el correcto funcionamiento de DACME se realizaron varias pruebas.

La primera prueba que se realizó fue la de asignación de prioridades. Se comprobó que efectivamente el mecanismo empleado para asignar las prioridades a cada flujo funcionaba correctamente, tal y como se hizo en el capítulo 5.

La segunda prueba realizada fue la prueba de estabilidad. Para ello se configuró un conjunto de pruebas cuya duración fue de tres días en los que se iban dando de alta y de baja diversos flujos (utilizando el entorno descrito en la sección 4). Durante esta prueba, y al final de la misma, se comprobó que:

- Todos los flujos que se daban de baja se eliminaban del sistema.
- El consumo de memoria no aumentase más que el estrictamente necesario.
- No existiesen pérdidas de memoria.
- Al realizar las pruebas periódicas, las reglas de QoS se insertaban y se eliminaban correctamente.

Por estas pruebas se puede concluir que la implementación del sistema es lo suficientemente robusta para un uso continuado del programa sin sufrir ningún efecto colateral por su uso.

La última de las pruebas que se realizó es la comprobación del funcionamiento del correcto bloqueo/desbloqueo de flujos cuando las condiciones de transmisión no se cumplen.

En la figura 7.11, se observa una de las pruebas realizadas, en este caso se crearon 4 flujos en máquinas distintas, con el mismo destino que se encontraba a 4 saltos de cada uno de ellos.

Los flujos creados son:

- En el segundo 0, tráfico de Voz con una tasa de bits de 500 Kbit/s durante 85 s.

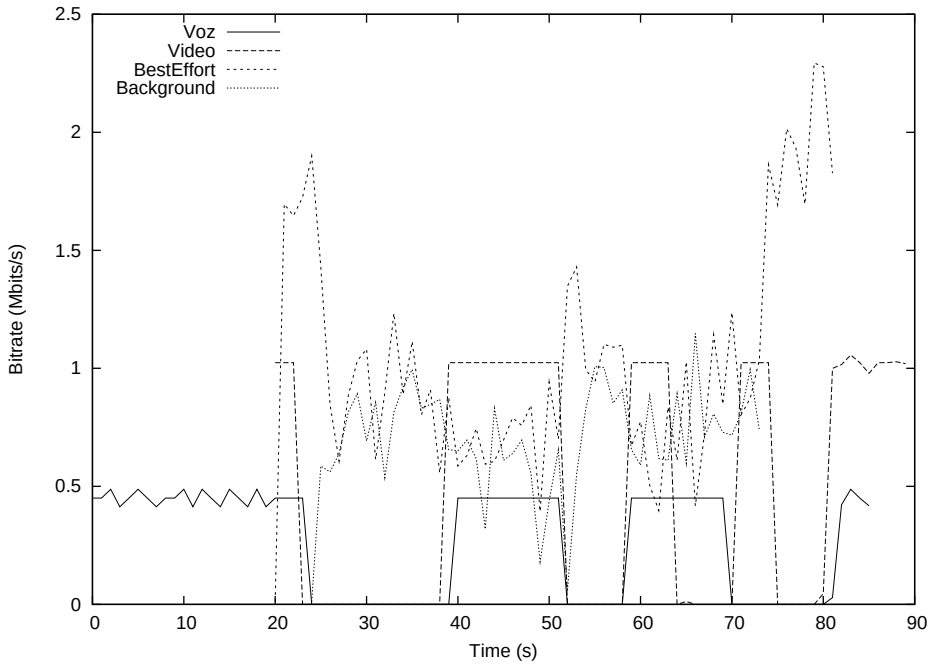

Figura 7.11: Prueba de ejecución con DACME

- En el segundo 20, tráfico de Vídeo de 1 Mbit/s durante 70 s.

- También en el segundo 20, tráfico de Best Effort durante 60 s.

- Por último tráfico de Background durante 50 s, comenzando en el segundo 24.

En la gráfica se aprecia como, con la llegada de los dos flujos de baja prioridad que no son controlados por DACME, las características de la red varían de forma significativa, y como DACME actúa consecuentemente bloqueando y desbloqueando los flujos que hay registrados.

A partir del segundo 20 (instante en que se activa el tráfico Best Effort), cuando se produce el test periódico, se puede apreciar cómo se detecta que las condiciones de la red no son las demandadas, y cómo DACME actúa consecuentemente bloqueando los flujos de la categoría de Voz y Vídeo.

Cuando se activa el tráfico de Background se verifica que los flujos de Voz y Vídeo pasan por varios momentos en los que las condiciones de la red sí que permitían el envío de los datos con las condiciones requeridas, por lo que DACME vuelve a activar los flujos.

Cuando en el segundo 80 finaliza el tráfico sin QoS, ambos tipos de tráfico (Voz y Vídeo) recuperan el rendimiento inicial al disponer de más recursos del canal.

	Tiempo activo (s)	**Bitrate medio (Mbit/s)**	**Desviación típica (Mbit/s)**	**% de desviación típica**
SIN DACME	80	1,44	0,468	32,32 %
CON DACME	70	1,52	0,067	4,52 %

Cuadro 7.1: Datos objetivos del empleo de DACME

7.6. Análisis de efectividad real de DACME

Tras comprobar que DACME tiene un comportamiento dentro de lo esperado se procedió a analizar las mejoras que puede conllevar su uso en este caso se analiza el rendimiento de DACME desde dos frentes distintos: centrándonos en los datos numéricos y centrándonos en los datos visuales.

7.6.1. Datos numéricos

En la tabla 7.1 se pueden observar los datos obtenidos en un escenario en el que se emplean cuatro flujos de 1,5 Mbit/s cada uno. Siendo el número de saltos variable. El primer flujo debe atravesar 7 saltos, el segundo 6, el tercero 5 y el último 4. Además los últimos 4 saltos son compartidos por todos los flujos por lo que en un momento existe un cuello de botella en el que DACME debe actuar.

Naturalmente, al no emplear DACME, el número de segundos activo de cada flujo es el total (80 segundos) sin embargo al emplear DACME, como este bloquea flujos el número de segundos que el flujo está activo es, naturalmente, menor.

El bitrate medio alcanzado cuando los flujos están activos el bitrate medio conseguido por los flujos es mayor con DACME, y además la desviación típica es sensiblemente menor (pasando de un 32 % a un 4,5 %) consiguiendo de esta manera una gran estabilidad en los flujos. Esta estabilidad significa calidad de servicio, que es lo que deseamos obtener al emplear un método de control de acceso.

7.6.2. Datos visuales

En el trabajo original de DACME realizado mediante simulación se obtuvieron gráficas de cual debería ser el comportamiento óptimo de DACME en este caso. Tal y como se puede observar en la figura 7.12, DACME realiza extremadamente bien su función de control de acceso, permitiendo la entrada solamente a los flujos que tienen suficiente ancho de banda como para usar el medio cumpliendo todas las restricciones.

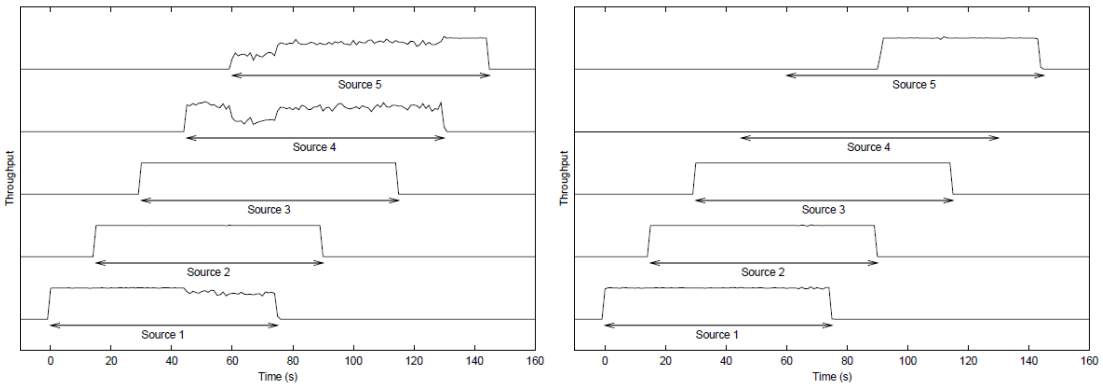

Figura 7.12: Resultados de DACME obtenidos en la simulación

A izquierda se encuentran cinco flujos de datos sin emplear ninguno de ellos DACME, cuando el flujo "Source 4" comienza a transmitir se ve afectado "Source 1" y cuando comienza a transmitir "Source 5" se ven todavía más perjudicados tanto "Source 1" como "Source 4". Sin embargo a la derecha, en el escenario con DACME activo se observa como en ningún momento "Source 4" llega a activarse ya que en la red no hay suficiente ancho de banda disponible para no molestar a otros flujos y "Source 5" no se activa hasta que "Source 1" ha finalizado consiguiendo así mantener un nivel de calidad muy alto en todos los flujos que se permiten.

Sin embargo, si observamos los resultados que se obtienen con la implementación actual (figura 7.13) se puede observar que, cuando no se encuentran los flujos controlados por DACME (a la izquierda), el comportamiento en un canal saturado es totalmente errático. Sin embargo, cuando se emplea DACME, los flujos aceptados tienen un comportamiento excelente (tal y como constatan los datos numéricos expuestos anteriormente), aunque estos flujos tienen una gran variabilidad en lo referente a "activación / desactivación"; es decir, no se consigue mantener los flujos estables a lo largo del tiempo, sino que, al poseer el medio unas condiciones tan cambiantes, se activan o desactivan los diferentes flujos de una manera intermitente impidiendo al usuario realizar una videoconferencia sin cortes.

7.6.3. Conclusiones

El uso de DACME como control de admisión ha conseguido que, cuando los flujos estén activos, posean ciertas garantías de ancho de banda ya que la variabilidad es menor, tal y como se puede observar tanto en los datos visuales como en los datos numéricos.

En general se puede considerar que DACME es un paso adelante para conseguir QoS en entornos MANET por ser un programa computacionalmente no intensivo (ahorro de baterías) y que proporciona unos buenos resultados.

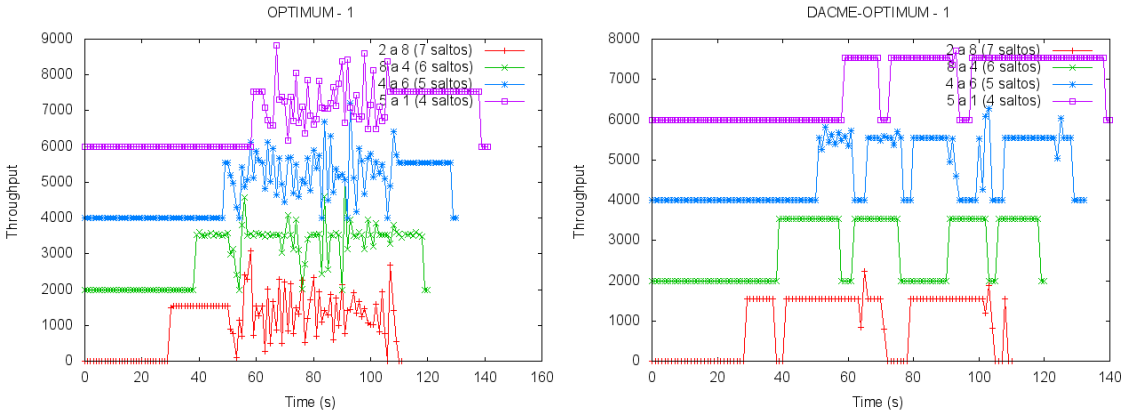

Figura 7.13: Resultados de DACME obtenidos en el testbed

Sin embargo, debido a la naturaleza tan cambiante del entorno inalámbrico, DACME ve mermada su capacidad para mantener los flujos siempre activos, por lo que se deben plantear ciertas mejoras que consigan un comportamiento del programa más cercano al obtenido en simulación.

7.6.4. Modificaciones propuestas para DACME

Al ver que los resultados de DACME eran buenos numéricamente pero que, sin embargo, gráficamente no son tan buenos, se proponen una serie de mejoras para estudiar si se consigue mantener una mayor estabilidad en los flujos.

Las dos modificaciones propuestas son las siguientes:

7.6.4.1. Eliminación del primer paquete de medición de retardo

Al realizar la medición de retardo, tal y como se ha explicado anteriormente, se lanza un primer paquete que no computa en el cálculo del retardo para disparar el protocolo de encaminamiento en caso de que sea proactivo, pero que en caso de pérdida asume que no hay conectividad con el nodo destino y, por lo tanto, bloquea el flujo.

Como nuestras pruebas se están realizando únicamente con encaminamiento estático o encaminamiento proactivo (OLSR) no es necesario el empleo de este paquete que, en caso de pérdida, lo único que provoca es que se bloquee el flujo incorrectamente.

7.6.4.2. Introducción de histéresis en las pruebas cíclicas

Dado que se detectó en varios momentos que las condiciones del canal varían rápidamente de un segundo a otro, produciéndose conexiones y desconexiones muy

rápidas se decidió (aún a costa de perder un poco de velocidad de reacción) tomar la decisión de bloquear o no el flujo sólo si la prueba anterior había dado el mismo resultado. En caso de no ser así se mantiene el flujo tal y como está.

Con esta modificación se pretende dotar al flujo de una mayor estabilidad, tanto para evitar que en una prueba puntual la coincidencia con otros flujos DACME que estén haciendo también sus pruebas nos lleve a malas estimaciones del ancho de banda, como para evitar activarnos en un momento en el que las condiciones del canal han mejorado levemente pero que sin embargo son totalmente transitorias.

Capítulo 8

Desarrollo de interfaces de uso para el sistema implementado

DACME, al ser un sistema que debe ser ejecutado en segundo plano y sin la intervención del usuario, no dispone de una interfaz gráfica o por comandos para el alta/baja de flujos. Por ello, y por dar soporte a las aplicaciones que de forma nativa (y transparente para el usuario) requieran flujos de tráfico con determinados requisitos de QoS, la comunicación con DACME se realiza por TCP siguiendo unas determinadas directivas.

8.1. Comunicación con DACME

Como DACME ha sido pensado para ofrecer la máxima compatibilidad con todos los lenguajes de programación que soporten el envío de paquetes TCP, todos los mensajes de comunicación con DACME son en texto plano (ASCII).

Al agente DACME se le pueden demandar tres funcionalidades:

1. Dar de alta (registrar) un nuevo flujo con determinadas características de QoS.
2. Dar de baja un flujo previamente registrado en el agente.
3. Proporcionar un listado de todos los flujos que DACME tiene registrados en el sistema.

Así siendo, el primer byte de cada mensaje es uno de estos dígitos, indicando la operación a realizar por parte de DACME.

Tanto en el caso 1 como en el 2 el cliente envía, además, las características del flujo que quiere dar de alta/baja como parte del mensaje. Estas características son:

- IP destino

- Puerto fuente
- Puerto destino
- Ancho de banda requerido
- Tamaño de paquete
- Retardo demandado
- Categoría del tráfico

En el caso 3 solamente se envía este dígito, pero DACME contesta con una lista de flujos separados por el signo "-". La primera parte de este mensaje de contestación es el número de flujos devueltos.

8.2. Soporte a aplicaciones nativas

Aunque DACME proporciona a los flujos que lo soliciten características determinadas de QoS, es posible que las propias aplicaciones requieran flujos controlados por DACME sin que el usuario lo sepa.

Dado que la comunicación con DACME es a través de TCP, cualquier lenguaje de programación que disponga de esta pila de protocolos implementada puede interactuar con DACME. Actualmente se han creado interfaces para que los programadores puedan hacer uso de los servicios que proporciona DACME de una forma sencilla para los lenguajes C/C++ y Java.

8.2.1. Interfaz C/C++

```
dacme_flow f;

init_flow(&f);
f.port = port;
f.category = category;
f.bitrate = bitrate;
f.delay = delay;
f.packet_size = numbytes;
strcpy(f.ip, destination_ip);

sock = dacme_socket(&f);
```

Figura 8.1: Código necesario para utilizar DACME en un programa C/C++

En la figura 8.1 se exponen los pasos necesarios para conseguir un flujo DACME con determinadas características.

En primer lugar se debe crear un *dacme flow,* que es una estructura que agrupa todas las características que se pueden demandar de DACME. Tras esto se debe inicializar el flujo (con la orden *init_flow*), y posteriormente rellenar todas las características que se necesiten.

Una vez completado el flujo se crea un *dacme_socket* empleando el flujo anteriormente rellenado. Este socket resultante se maneja igualmente que un socket UDP normal.

Para dar de baja el flujo en DACME se debe utilizar la orden *dacme_release* empleando como parámetro el *dacme flow* creado anteriormente.

8.2.2. Interfaz Java

```
Dacme_flow f = new Dacme_flow(ip, source_port, port, delay, category, bit

Dacme d = new Dacme(local_ip);
DatagramSocket sock = d.dacme_socket(f);
```

Figura 8.2: Código necesario para la utilización de DACME en Java

En Java se sigue una metodología parecida a la empleada en C/C++, En primer lugar se debe crear un *Dacme_flow* con las características requeridas. Tras esto se crea el objeto *Dacme* que será el encargado de conectar con el agente DACME deseado (para ello hay que indicar la IP de la máquina donde se encuentra el agente).

Tras esto se pueden emplear los métodos de la clase *Dacme* para dar de alta, baja o listar los sockets del agente DACME deseado.

Cuando se da de alta un nuevo flujo el resultado para el programador es un socket UDP estándar de Java que, de una forma transparente es gestionado por DACME para garantizar los requisitos de QoS que se han demandado.

8.3. Soporte a aplicaciones no nativas

Empleando la interfaz descrita anteriormente en el punto 8.2.2 se realizó una aplicación gráfica para que los usuarios puedan dar de alta en DACME sus flujos de forma manual.

La aplicación cuenta de una ventana principal que podemos observar en la figura 8.3; en esta ventana se pueden encontrar varios botones que están vinculados a las tres opciones disponibles en DACME.

El primer botón que encontramos es *"Connect"*, y se utiliza para realizar una conexión con el agente DACME deseado. Si la conexión es correcta se listan los flujos existentes en el agente, y en la parte baja se actualiza el estado de "*Currently connected to:*".

En segundo lugar encontramos el botón "*Create flow*"; este botón abre un nuevo diálogo en el que se pueden especificar las características deseadas para un nuevo flujo.

En tercer lugar, el botón "*Refresh*" sirve para, una vez conectado a un agente DACME, actualizar el listado de flujos del agente.

Por último, en cuarto lugar, "Delete flow" sirve para dar de baja un flujo seleccionado.

En la figura 8.4 se muestra el diálogo que aparece al pulsar el botón "*Create flow*". En este diálogo que aparece con cierta información predefinida se pueden definir todas las características del flujo.

Una vez pulsado el botón "Create" pueden ocurrir dos cosas: que el flujo se registre correctamente, y, por tanto, volveremos a la ventana principal y podremos ver el estado del flujo (*running* o *bloqued*); que el flujo no se haya podido crear, situación en que se nos mostrará un mensaje de error como el que podemos ver en la figura 8.5, donde además de decir que le flujo no ha sido creado se muestra la posible causa por la que no se ha podido crear.

8.4. Integración con Castadiva

Como se ha visto anteriormente en la sección 4.2, Castadiva es un emulador de redes MANET con el que se puede probar el código de diferentes aplicaciones en este entorno empleando código real.

Para poder probar el correcto funcionamiento de DACME y poder realizar todas las pruebas descritas anteriormente se ha integrado con Castadiva, pero naturalmente solamente para los flujos que Castadiva soporta. En este caso se ha implementado la opción de usar DACME al usar flujos UDP. Para activarlo simplemente hay que poner a "*true*" la casilla de la fila correspondiente al flujo que queremos que DACME controle.

Adicionalmente DACME debe de ser lanzado manualmente en los nodos fuente y destino para un comportamiento correcto.

Como se puede observar en la figura 4.12, cuando se activa la opción de DACME se puede introducir un "*Delay*"; este campo hace referencia al retardo máximo que estamos dispuestos a permitir que tenga la aplicación. Si este valor lo dejamos en 0 DACME interpretará que el retardo que exigimos para el flujo puede ser infinito

y no realizará las pruebas concernientes a retardo (por lo que solamente bloqueará por ancho de banda).

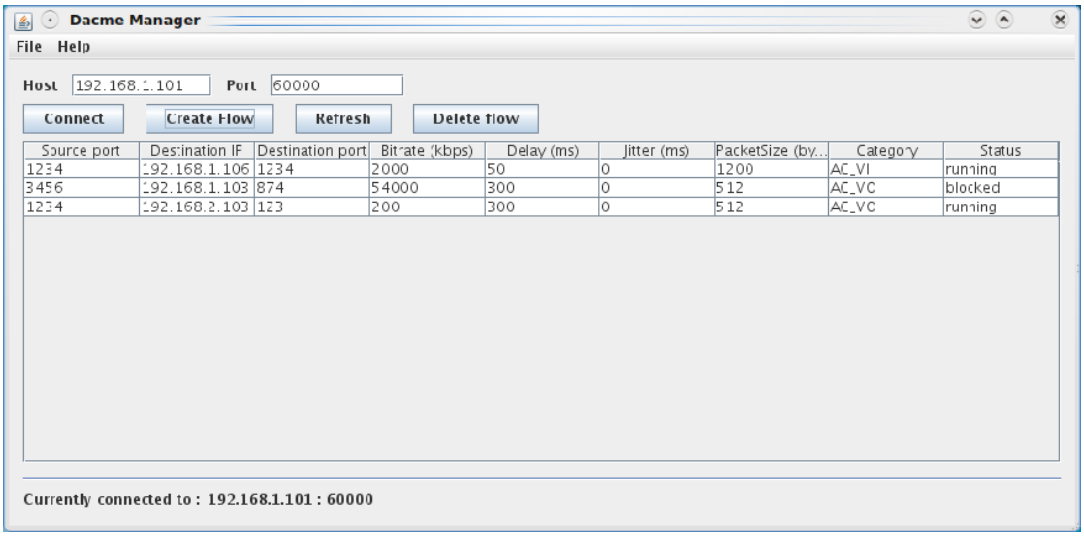

Figura 8.3: Pantalla principal de DACME Manager

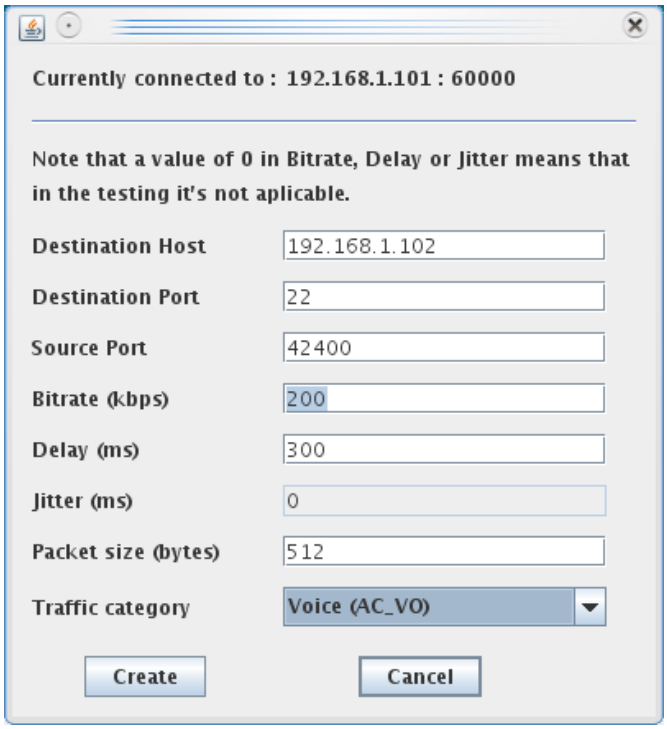

Figura 8.4: Dialogo de características del nuevo flujo

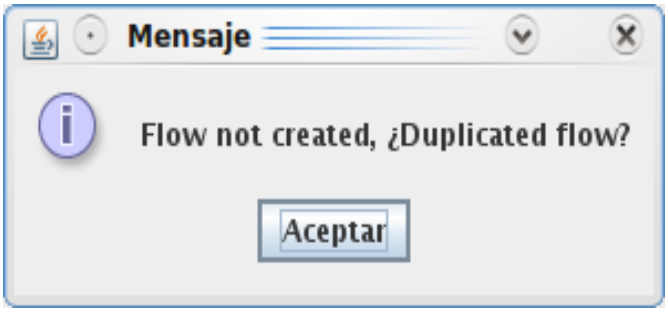

Figura 8.5: Mensaje de error

Capítulo 9

Conclusiones y trabajo futuro

La temática de esta tesina se ha centrado en el área de la QoS en redes ad-hoc multisalto. El hecho de que se tratase de un área relativamente joven ha permitido experimentar con tecnologías y entornos novedosos, así como aportar nuevos datos científicos de elevada relevancia en un ámbito dónde otros investigadores del área no habían logrado resultados aún: el entorno real.

La labor realizada en el proyecto final de carrera se puede dividir esencialmente en dos partes: el desarrollo de un entorno de pruebas real y el desarrollo de un sistema de control de admisión distribuido.

En la primera parte se ha preparado un entorno ad-hoc con soporte a QoS mediante la activación del IEEE 802.11e en las tarjetas inalámbricas de los terminales implicados, lo que ha requerido realizar mejoras a los drivers existentes. En este entorno se ha realizado un estudio de prestaciones del estándar IEEE 802.11e, algo pionero en este área científica.

En general los resultados obtenidos muestran que en este tipo de redes no hay que ser demasiado optimistas en cuanto a prestaciones, ya que los resultados obtenidos muestran una pérdida de efectividad por parte del IEEE 802.11e en la diferenciación de tráfico para un numero de saltos típico (4). Las pruebas realizadas muestran, además, que el ancho de banda consumido por las categorías más bajas es relativamente alto. Así siendo, aunque es posible la realización de una videoconferencia en tiempo real, la realización de llamadas VoIP conllevaría un retardo demasiado elevado para que la llamada se efectuase con una calidad considerada aceptable debido al elevado retardo percibido.

Para asegurarnos de estas conclusiones se distribuyó el testbed por toda la planta de un edificio consiguiendo así un entorno multisalto real (no se emula la visibilidad) con el que se obtuvieron datos muy interesantes que confirman nuestras conclusiones anteriores y que, además, nos permiten detectar las carencias del algoritmo de encaminamiento para MANETs más importante actualmente, así como conocer las

características que un buen algoritmo de encaminamiento de entornos MANET debería de poseer para poder proporcionar a los usuarios una buena calidad de servicio.

Gracias a las necesidades de creación de este entorno se ha mejorado enormemente una herramienta diseñada previamente en el Grupo de Redes de Computadores (Castadiva), que la hacen muy atractiva para la realización de grandes cantidades de pruebas de entornos MANET.

La segunda parte del proyecto ha consistido en la implementación del sistema de control de admisión propuesto por Calafate et al. [8]. Este sistema, denominado DACME, permite mejorar el soporte a QoS en MANETs mediante control de admisión distribuido basado en paquetes de prueba. En el proyecto se ha realizado la implementación de DACME para un entorno GNU/Linux, así como una interfaz de programación y una interfaz de usuario bastante intuitiva y sencilla de utilizar. Se ha realizado también un análisis de prestaciones de DACME en el entorno de pruebas real desarrollado, lo que ha permitido concluir que se ha conseguido un gestor de admisión con soporte a calidad de servicio suficientemente robusto como para que se comience a usar en otro entorno distinto al académico.

El trabajo futuro a realizar se centra principalmente en la ampliación y mejoras de la funcionalidad de DACME y DACME Manager.

9.1. Publicaciones

La labor realizada en este trabajo ha dado lugar a diversas publicaciones científicas relacionas con el entorno de pruebas IEEE 802.11e, y que se pasan a detallar:

9.1.1. Internacionales

- "Deploying a real IEEE 802.11e testbed to validate simulation results", Alvaro Torres, Carlos T. Calafate, Juan-Carlos Cano, Pietro Manzoni, 34th Annual IEEE Conference on Local Computer Networks (LCN), Zurich, Switzerland. October 20-23, 2009. - **CORE A**.

- ”Assessing the IEEE 802.11e QoS effectiveness in multi-hop indoor scenarios”, Alvaro Torres, Castos T. Calafate, Juan-Carlos Cano, Pietro Manzoni, Ad Hoc Networking (2010), doi:10.1016/j.adhoc.2010.07.011 **- REVISTA, Indexada en el 1er tercio del JCR.**

9.1.2. Nacionales

- "Experimenting with the IEEE 802.11e technology in a real testbed", A. Torres, Carlos T. Calafate, Juan Carlos Cano, Pietro Manzoni. XX Jornadas de

Paralelismo, La Coruña, Spain. 16-18 September, 2009.

- "On the feasibility of real-time videoconferencing using QoS-enabled MANETs", Alvaro Torres, Carlos T. Calafate, Juan Carlos Cano, Pietro Manzoni. WMDCT (CEDI 2010), Valencia, Spain. September, 2010.
- ”Extending an emulation platform for automatized and distributed evaluation of QoS in MANETs”, Wannes Vossen, Alvaro Torres, Jorge Hortelano, Carlos T. Calafate, Juan-Carlos Cano, Pietro Manzoni. XXI Jornadas de Paralelismo (CEDI 2010), Valencia, Spain. September, 2010.

Bibliografía

[1] Network Time Protocol. http://datatracker.ietf.org/wg/ntp/charter/. Acessed: May 3, 2010.

[2] Ralink Technology Corporation. Available at: http://www.ralinktech.com/. Accessed: January 30, 2009.

[3] The netfilter.org iptables project. Available at: http://www.netfilter.org/. Accessed: January 28, 2009.

[4] IEEE P802.11 - Task Group G, Draft 6.1 for IEEE 802.11g. http://grouper.ieee.org/groups/802/11/, 2003.

[5] Alvaro Torres, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni. Deploying a real IEEE 802.11e testbed to validate simulation results. In *34th IEEE Conference on Local Computer Networks (LCN)*, Zurich, Switzerland, 2009.

[6] A. Banchs, A. Azcorra, C. Garcia, and R. Cuevas. Applications and challenges of the 802.11e EDCA mechanism: an experimental study. *IEEE Network*, 19(4):52–58, 2005.

[7] Carlos T. Calafate, Juan-Carlos Cano, Pietro Manzoni, and Manuel P. Malumbres. Achieving enhanced performance in MANETs using IEEE 802.11e. In *XV Jornadas de Paralelismo*, Almeria, Spain, September 2004.

[8] Carlos T. Calafate, Manuel P. Malumbres, Jose Oliver, Juan Carlos Cano, and Pietro Manzoni. QoS Support in MANETs: a Modular Architecture Based on the IEEE 802.11e Technology. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(5):678–692, May 2009.

[9] Carlos T. Calafate, Pietro Manzoni, and Manuel P. Malumbres. Assessing the effectiveness of IEEE 802.11e in multi-hop mobile network environments. In *12th IEEE International Symposium on Modeling, Analysis, and Simulation of*

Computer and Telecommunication Systems (MASCOTS'04), Volendam, Netherlands, October 2004.

[10] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. Ad hoc on-demand distance vector (AODV) routing. Request for Comments 3561, MANET Working Group, http://www.ietf.org/rfc/rfc3561.txt, July 2003. Work in progress.

[11] Dmitri D. Perkins and Herman D. Hughes. A survey on quality-of-service support for mobile ad hoc networks. *Wireless Communications and Mobile Computing Journal*, 2(5):503–513, 2002.

[12] IEEE 802.11 WG. International Standard for Information Technology - Telecom. and Information exchange between systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications, ISO/IEC 8802-11:1999(E) IEEE Std. 802.11, 1999.

[13] IEEE 802.11 WG. 802.11e IEEE Standard for Information technology- Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements, 2005.

[14] Jorge Hortelano, Juan-Carlos Cano, Carlos T. Calafate, and Pietro Manzoni. Testing applications in manet environments through emulation. *EURASIP Journal on Wireless Communications and Networking, vol. 2009, Article ID 406979, 20 pages, 2009. doi:10.1155/2009/406979.*

[15] S. Blake. An Architecture for Differentiated Services. IETF RFC 2475, December 1998.

[16] S. Xu and T. Saadawi. Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks? *IEEE Communications Magazine*, 39(6):130–137, June 2001.

[17] Vasilios A. Siris and George Stamatakis. Optimal CWmin selection for achieving proportional fairness in multi-rate 802.11e WLANs: test-bed implementation and evaluation. In *1st international workshop on Wireless network testbeds, experimental evaluation & characterization*, pages 41–48, New York, NY, USA, 2006. ACM Press.

[18] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). Request for Comments 3626, MANET Working Group, http://www.ietf.org/rfc/rfc3626.txt, October 2003. Work in progress.