# Tesis Magíster Redes de Datos

**"QoS en redes wireless con IPv6"**


Autor
Matías Robles

Director
Ing. Luis Marrone



Facultad de Informática
Universidad Nacional de La Plata
2008

*...A mis padres*

Agradezco a:

Ing. Luis Marrone por el apoyo recibido en el desarrollo de esta Tesis

Facultad de Informática de la U.N.L.P. por la formación desinteresada.

... y a mis seres queridos por la paciencia en todos estos años.

# Índice

# Figuras

# Prefacio

El objetivo de este trabajo es analizar y proponer técnicas de Calidad de Servicio en redes wireless usando el protocolo IPv6. Estas técnicas están empezando a tener una gran importancia dentro del ámbito de las redes de datos. Esto se debe, principalmente, a las nuevas aplicaciones que hacen uso de dichas redes. Por su parte, las redes inalámbricas han tenido en los últimos años una incursión masiva en las redes hogareñas y empresariales y ese crecimiento continúa en alza. Pero, a pesar de incrementar el ancho de banda disponible en las sucesivas especificaciones, su rendimiento real no alcanza para cubrir todas las necesidades funcionales de esas aplicaciones. Por último, IPv6 se presenta como el sucesor de IPv4 y entre sus mejoras existen algunas que están relacionadas con la Calidad de Servicio. Es por esto, que ante el avance de estas técnicas y tecnologías, se abre una excelente posibilidad para integrarlas.

Cada vez más las redes de datos se están convirtiendo en redes multimediales transportando voz y video, además de los datos tradicionales, lo que les permite brindar más servicios a los usuarios. Estos diferentes tipos de tráficos requieren un especial tratamiento por parte de la red y son las técnicas de Calidad de Servicio las encargadas de cumplir con los respectivos requerimientos para que los usuarios puedan acceder en forma adecuada a esos servicios.

Obviamente, las redes inalámbricas también se encuentran dentro de este escenario pero, debido a su baja eficiencia por sus características de funcionamiento, el soporte de las técnicas de Calidad de Servicio cobra un especial interés en esta clase de redes. Para cubrir esta deficiencia, la IEEE desarrolló el estándar 802.11e que permite aplicar Calidad de Servicio a las redes inalámbricas.

De acuerdo a lo expresado anteriormente, este trabajo propone la utilización del protocolo IPv6 mediante la redefinición del campo Flow Label, que se encuentra en la cabecera de dicho protocolo, para que las aplicaciones puedan indicar sus requerimientos de Calidad de Servicio a la red. A partir del valor en este campo, los paquetes son asignados a las distintas colas definidas por el estándar 802.11e.

Con el fin de probar la propuesta se define una topología de una red inalámbrica sobre la que se generan tráficos con diferentes requerimientos. Todos estos esquemas son validados utilizando un simulador, el NS-2, implementado en la Universidad de Berkeley. El código del simulador debe ser modificado para poder aceptar las modificaciones planteadas en el trabajo.

En el Capítulo 1 se describen las características básicas de IPv6 y de los demás protocolos usados por éste. Se brinda una breve explicación sobre su funcionamiento.

En el siguiente capítulo, Capítulo 2, se explica el funcionamiento de las redes inalámbricas, específicamente las definidas por el estándar 802.11 de la IEEE.

En el Capítulo 3, se realiza una introducción a las diferentes técnicas de Calidad de Servicio y se explica el funcionamiento del estándar 802.11e, que define como hace una red 802.11 para brindar Calidad de Servicio.

En el último capítulo, el Capítulo 4, se detalla como se redefine el campo Flow Label de la cabecera IPv6, se indican las modificaciones que se realizaron al simulador y, finalmente, utilizando diferentes gráficos, se muestra el resultado de todas las pruebas realizadas.

Por último se realizan las conclusiones del trabajo y se indican cuales son los trabajos futuros posibles.

Además, este trabajo contiene dos apéndices:

Apéndice A, se explica como se instala, configura y utiliza el simulador NS-2 que es la herramienta utilizada para realizar todas las pruebas de esta tesis.

Apéndice B, se incluyen todos los scripts utilizados en las diferentes simulaciones. Además se encuentran los scripts usados para obtener los resultados.

En el último apartado se encuentra la bibliografía consultada, con una breve explicación de lo que se puede consultar en cada uno de los libros o artículos.

# Capítulo 1
# IPv6

## 1 - Introducción

Desde que fue publicado en 1981, el protocolo IPv4 no ha sufrido grandes modificaciones. En más de 20 años de uso, ha demostrado ser flexible, robusto y poderoso. Sin embargo, ha comenzado a mostrar ciertas limitaciones para adecuarse al funcionamiento de las redes actuales y sus nuevas demandas. El crecimiento exponencial que ha tenido Internet en esta última década y, el advenimiento de nuevas tecnologías, han provocado la aparición de ciertas condiciones que el diseño original del protocolo no anticipó (obviamente, no se equivocaron los diseñadores de IPv4, sino que la tecnología ha avanzado mucho más rápido de lo esperado), y que algunas de ellas se enumeran a continuación:

- Escasez de direcciones IPv4 libres para otorgar
- Imposibilidad de los routers del backbone de Internet de mantener tablas de ruteo extremadamente largas
- Inexistencia de una manera simple de configuración de direcciones
- Carencia de un buen método para el envío de tráfico en tiempo real (conocido como Quality of Service - QoS)
- Falta de un mecanismo de seguridad en la capa de red.

Pensando en solucionar estos problemas, y tratando de anticipar los nuevos avances tecnológicos, es que se ha diseñado un nuevo protocolo: IPv6. Las siguientes son sus características más sobresalientes:

- Formato de cabecera simplificado
- Espacio de direcciones más grande (128 bits)
- Direccionamiento e infraestructura de ruteo eficiente y jerárquica
- Configuración de direcciones con y sin estado
- Seguridad intrínseca en el núcleo del protocolo
- Mejor soporte para la Calidad de Servicio
- Nuevo protocolo para la interacción entre nodos vecinos
- Extensibilidad
- Multicast (envío de un mismo paquete a un grupo de receptores)
- Anycast (envío de un paquete a un receptor dentro de un grupo)
- Posibilidad de enviar paquetes con más de 65.535 bytes (jumbogramas)
- Renumeración y multi-homing, que facilita el cambio de proveedor de servicios de Internet
- Características de movilidad

## 2 - Cabecera IPv6

El formato de la cabecera IPv6 es el siguiente:

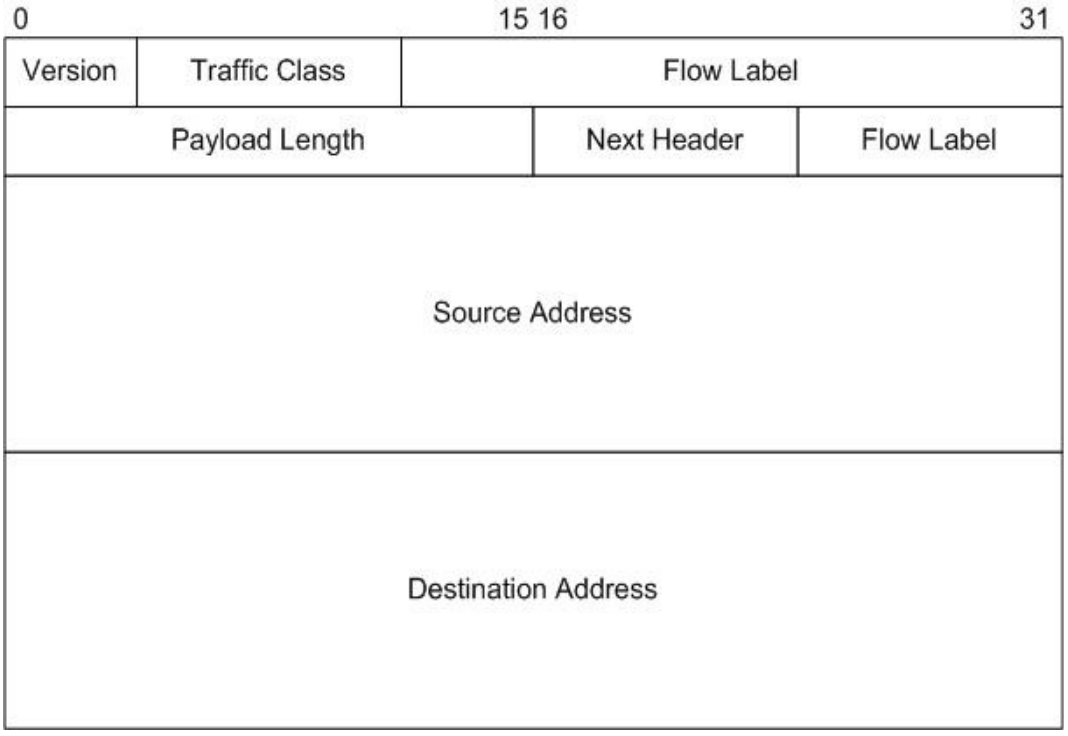



Figura 1.1 - Formato cabecera IPv6

La siguiente lista describe la función de cada campo en la cabecera:

- Version (4 bits): indica la versión del protocolo
- Traffic Class (8 bits): indica la clase o prioridad del paquete IPv6. Reemplaza al campo Type of Service de IPv4
- Flow Label (20 bits): indica que el paquete pertenece a un mismo flujo de tráfico entre un origen y un destino, requiriendo un manejo especial por parte de los routers intermedios.
- Payload Length (16 bits): indica la longitud de los datos después de la cabecera IPv6.
- Next Header (8 bits): indica cual es la cabecera de extensión siguiente, si existe, o el protocolo de capa superior (TCP o UDP). Reemplaza al campo Protocol Type de IPv4.
- Hop Limit (8 bits): indica la cantidad de routers por los que un paquete IPv6 puede pasar antes de ser descartado. Reemplaza al campo Time-to-Live (TTL) de IPv4.
- Source Address (128 bits): indica la dirección origen del emisor del paquete.
- Destination Address (128 bits): indica la dirección del nodo destino del paquete. (Nota: este campo puede no contener la dirección IPv6 del último destino si la cabecera de extensión Routing Header está presente)

El tamaño de las direcciones es de 128 bits (4 veces más grande que en IPv4), pero la longitud de la cabecera IPv6 es de 40 bytes, el doble del tamaño de la cabecera IPv4 sin las opciones. Esto se debe a que se eliminaron varios campos, de los cuales algunos se pasaron como cabeceras de extensión. Por ejemplo, en IPv4 existe el campo Options que es opcional y de longitud variable. Por esto, la cabecera de IPv4 contiene el campo Header Length para saber su longitud exacta. Este campo no es necesario en IPv6 porque el tamaño de la cabecera es fijo.

Un paquete IPv6 puede llevar cero, una o más cabeceras de extensión. Éstas se encuentran a continuación de la cabecera IPv6 y son las siguientes:

- Hop-by-Hop Option Header
- Routing Header
- Fragment Header
- Destination Header
- Authentication Header
- Encrypted Security Payload Header

Cada cabecera es identificada por el campo Next Header de la cabecera anterior. A excepción de la cabecera Hop-by-Hop Option, que debe ser examinada y procesada por cada nodo a lo largo del camino del paquete, las demás, solamente, son procesadas por el nodo destino, respetando estrictamente el orden en el que aparecen en el paquete. Un nodo destino no puede recorrer las cabeceras de extensión buscando un cabecera en particular.

La figura siguiente muestra cómo se forman los paquetes cuando no tienen ninguna cabecera de extensión, cuando tienen una y cuando tienen más de una. Además se indica el valor del campo Next-Header.

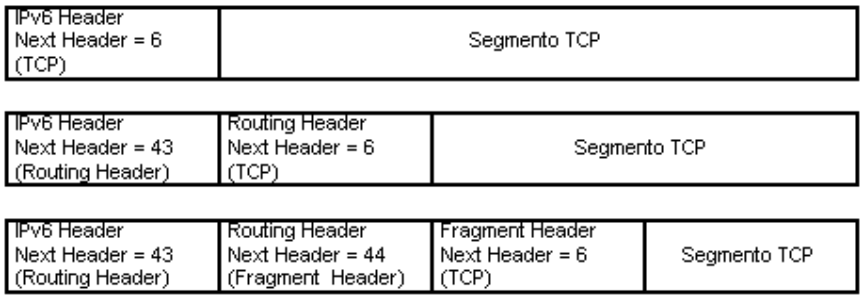



Figura 1.2 - Cabeceras de extensión

## 3 - Direcciones

Las direcciones son representadas como una serie de campos de 16 bits, hexadecimales, separados por dos puntos (:) con el siguiente formato x:x:x:x:x:x:x:x. Un doble dos puntos (::), se permite uno solo por dirección, puede ser usado para comprimir sucesivos campos hexadecimales de ceros.

En IPv6 no existen las direcciones broadcast, su función es sustituida por las direcciones multicast. Existe una dirección de loopback (::1) similar a la de IPv4 y se agrega un nuevo tipo de dirección: la dirección no especificada(::) que indica la ausencia de una dirección IPv6. Esta dirección no debe ser asignada a ninguna interface ni debe ser usada como dirección destino en un paquete IPv6.

Las direcciones se asignan a las interfaces, lo que permite que tengan más de una dirección asignada, y no a los nodos.

En IPv6, el concepto de máscara de red es reemplazado por el concepto de prefijo. Éste es un número decimal que indica cuantos bits contiguos de más alto orden son usados para identificar la porción de red de la dirección. Una dirección IPv6 está compuesta de la siguiente manera:

*dirección IPv6 / prefijo*

### 3.1 Tipos de Direcciones

Existen 3 tipos de direcciones:

a) Unicast

Una dirección unicast es una dirección para una sola interface. Un paquete enviado a una dirección unicast es entregado, solamente, a la interface indicada por esa dirección.

Las direcciones unicast IPv6 son similares a las direcciones IPv4 con CIDR (Classless Inter-Domain Routing). Los siguientes son algunos tipos de direcciones unicast:

*Global Address*

Estas direcciones son utilizadas para el tráfico IPv6 a través de la Internet IPv6. Son globalmente únicas y ruteables y se corresponden con las direcciones unicast públicas utilizadas en IPv4 para comunicarse en Internet. Representan la parte más importante de la arquitectura de direcciones de IPv6. La figura siguiente muestra su estructura:

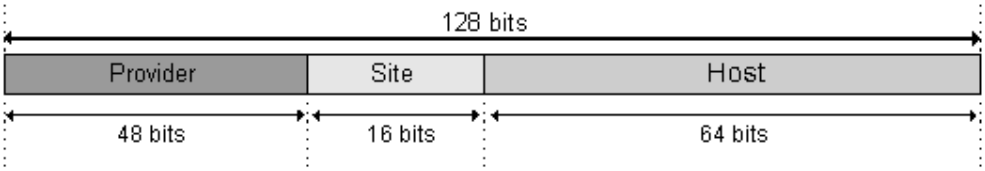



Figura 1.3 - Estructura dirección Unicast Global

- Provider: representa el prefijo de 48 bits cedido a una organización por algún proveedor autorizado.

- Site: con un /48 cedido a una organización, ésta puede manejar hasta 65.535 subredes diferentes. El sitio usa estos bits para subnetting.

- Host: esta parte, que representa los 64 bits de más bajo orden de la dirección, se llama *Interface ID*. Identifica una interface en un enlace. Debe ser único en ese enlace.

*Unique-Local Address*

En la especificación original de la arquitectura de direcciones de IPv6 se describe una dirección unicast llamada Site-Local Address. El prefijo de estas direcciones comenzaba con FEC0::/10. Su alcance era dentro de un mismo sitio u organización y tenían un funcionamiento similar al de las direcciones de redes privadas de IPv4 (10.0.0.0/8, 172.16.0.0/12 y 192.168.0.0/16). No debían salir del ámbito de las redes privadas.

Sin embargo, su uso fue prohibido debido a que causaban problemas en las aplicaciones y en el ruteo. En su reemplazo surgieron las direcciones Unique-Local. En la Figura 1.4 se muestra su formato y a continuación se explican cada uno de los elementos que la componen:

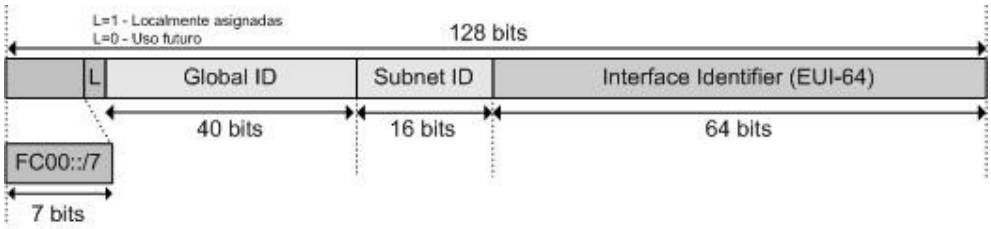



Figura 1.4 - Estructura dirección Unique-Local

- Prefijo: FEC0::/7 es utilizado para identificar una dirección unicast Local IPv6
- L: indica la política de asignación. Sólo está definido el valor 1, que implica asignada localmente, y genera el prefijo FD00::/8.
- Global ID: asegura la unicidad de la dirección al crear un prefijo globalmente único. Se genera en forma aleatoria. No debería ser asignado de manera secuencial.
- Subnet ID: permite que el administrador de la red pueda definir un plan de direccionamiento jerárquico. Permite definir una subred dentro del sitio.
- Interface Identifier: idem Interface Id de la dirección Global Address

Estas direcciones no deben ser ruteadas fuera de un sitio. El valor del campo Global ID, al ser generado en forma aleatoria, debería ser único.

*Link-Local Address*

Estas direcciones, identificadas por el FP = 1111 1110 10, son usadas para la autoconfiguración de direcciones, en funciones del protocolo Neighbor Discovery y cuando no existe un router en el link.

Cuando en una interface se habilita IPv6, la dirección de link-local es la primera dirección que se autoconfigura. Un nodo no puede utilizar IPv6 si no tiene una dirección de este tipo asignada.

El alcance de estas direcciones es el link. Un router nunca debería reenviar un paquete con una dirección de link-local, como dirección origen o destino, más allá del link.

El prefijo para una dirección de link-local es FE80::/10.

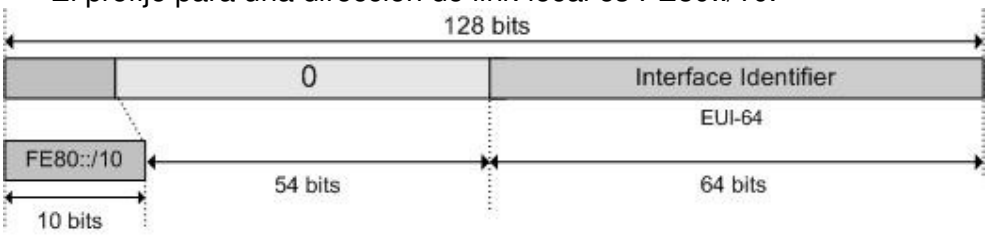



Figura 1.5 - Estructura dirección Link-Local

El identificador de interface (Interface Identifier) que tiene un tamaño de 64 bits se obtiene a partir de las direcciones MAC de 48 bits. El proceso es simple y consiste en insertar los bytes FFFE (hexadecimal) entre los bytes 3 y 4 de la dirección MAC y modificar un bit específico.

El identificador que se obtiene se conoce como EUI-64 (Extended Unique Identifier de 64 bits), de los cuales los primeros 24 bits indican el fabricante y son asignados por la IEEE, y los restantes 40 bits son asignados internamente por dichas compañías.

Para obtener el identificador de interfaz se debe complementar el bit U/L (Universal/Local). Este bit se utiliza para indicar si la dirección es local o universal y es el bit 7 del primer byte del identificador. Como se obtiene de una dirección MAC, el identificador obtenido es globalmente único, por lo que el bit 7 queda igual a 1. La finalidad de complementar el bit es, puramente, con fines administrativos. Si un administrador ingresa direcciones IPv6 manualmente, le es más simple ingresarlas como fec0::1, fec0::2, etc. y no tener que ingresar un 1 en la posición del bit U/L.

Este proceso es usado para obtener el identificador de interface de cualquiera de las direcciones unicast explicadas anteriormente. Si no existiese una dirección MAC en la interface (por ejemplo, las interfaces seriales), el identificador debe ser obtenido mediante algún otro procedimiento o ingresado manualmente por el administrador.

El siguiente gráfico muestra como se forman las direcciones EUI-64:

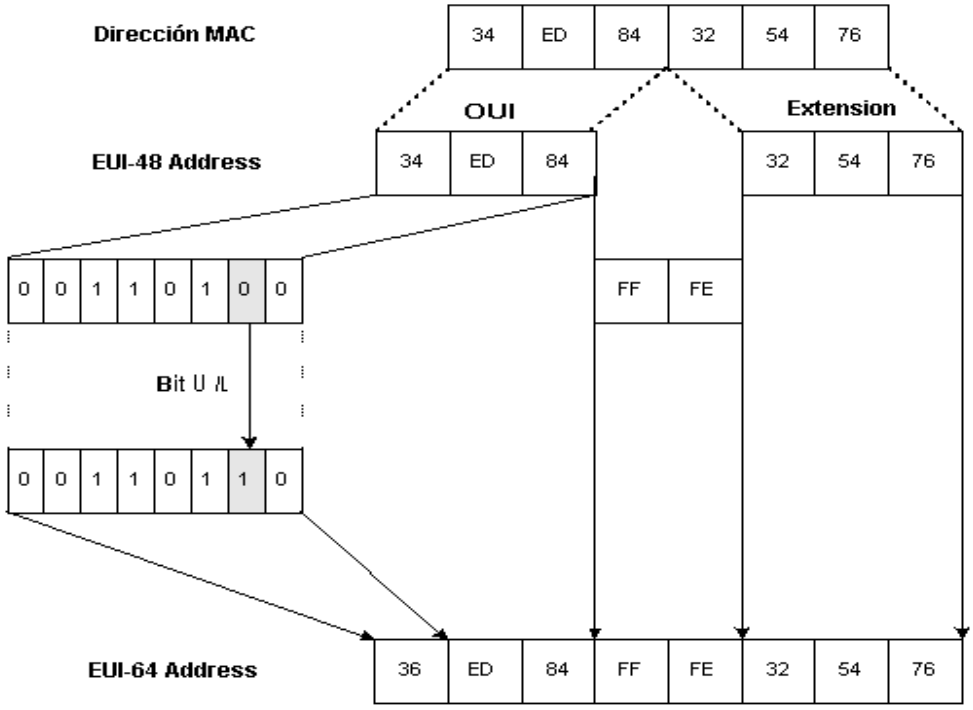



Figura 1.6 - Identificador de Interface EUI-64

La diferencia principal de estas direcciones es el ámbito, o alcance, que tienen. El siguiente gráfico muestra como se relacionan.

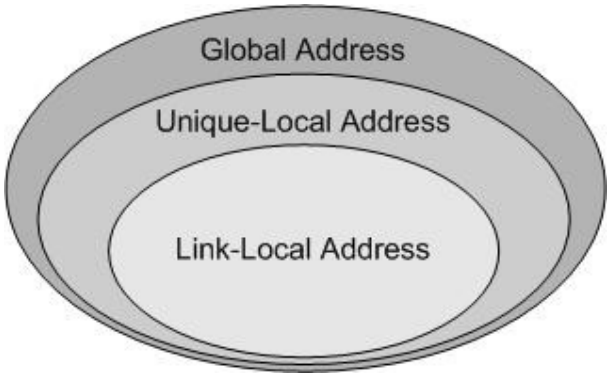



Figura 1.7 - Ámbito direcciones unicast

b) Anycast Address

La idea detrás de este tipo de dirección es proveer redundancia y balanceo de carga en situaciones donde múltiples hosts brindan el mismo servicio. Una misma dirección anycast se asigna a un grupo de interfaces, de tal manera, que un paquete enviado a esa dirección será entregado a un único miembro de ese grupo. Al nodo, perteneciente a ese grupo, que se encuentre más cercano de acuerdo a los protocolos de ruteo le será enviado el paquete. Esto significa que se deben agregar múltiples entradas en las tablas de ruteo.

Estas direcciones son tomadas del espacio de direcciones unicast, es decir, son sintácticamente indistinguibles una de las otras. Cuando se asignan a una interface se debe indicar que la dirección es de tipo anycast.

Existen varias cuestiones a tener en cuenta cuando se las utiliza. Por ejemplo, si un emisor envía varios paquetes a una dirección anycast, los paquetes pueden arribar a diferentes destinos. Si hay una serie de solicitudes y respuestas o el paquete tiene que ser fragmentado, podrían producirse problemas. En la actualidad este tipo de direcciones se encuentran es su etapa experimental.

c) Multicast Address

Identifica a un conjunto de interfaces, de tal modo, que un paquete enviado a una dirección multicast es entregado a todas las interfaces del grupo. Se identifican por el FP (Format Prefix) = 1111 1111, por lo cual, comienzan con el prefijo FF00::/8.

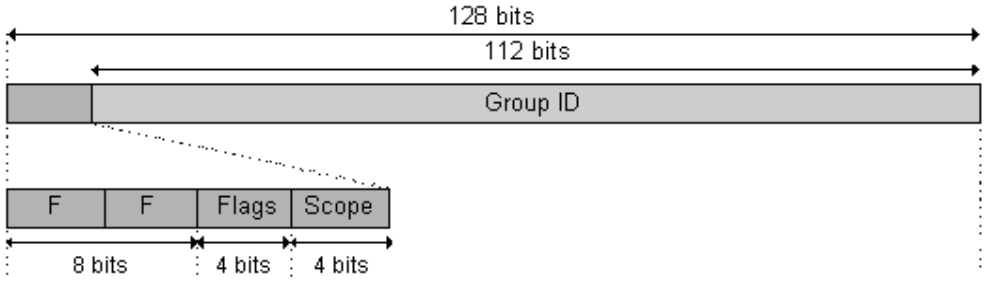



Figura 1.8 - Estructura dirección multicast

El campo Flags indica si una dirección multicast es permanente (0) o si es temporal (1).

El campo Scope limita el alcance de un grupo multicast.

Entre las direcciones multicast asignadas en forma permanente se encuentran las siguientes:

- FF02::1/8 (dirección multicast de todos los nodos del link-local)
- FF02::2/8 (dirección multicast de todos los routers del link-local).

Existe otro tipo de dirección multicast, utilizada por el protocolo Neighbor Discovery, que es la *dirección multicast de nodo solicitado*. Esta dirección permite, a los nodos, un eficiente método de consulta durante el proceso de resolución de direcciones.

El prefijo de este tipo de direcciones es FF02:0:0:0:0:1:FFxx:xxxx/104, donde los últimos 24 bits son los últimos 24 bits de la dirección unicast o anycast que se está intentando resolver. Son utilizadas en los mensajes Neighbor Solicitation del Neighbor Discovery. En la figura 1.9 se muestra la estructura de este tipo de direcciones.

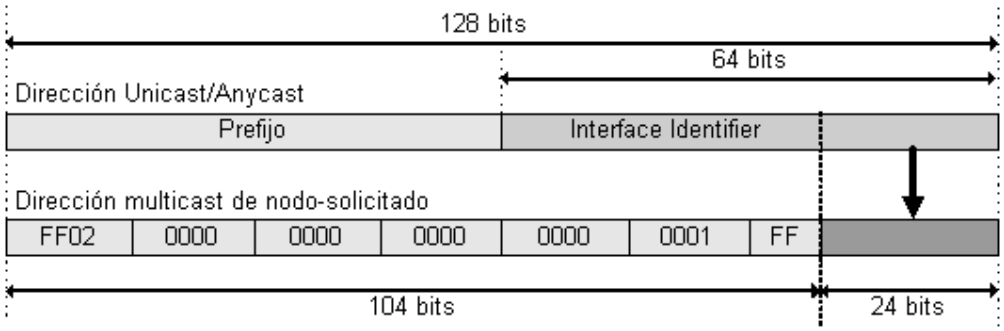



Figura 1.9 - Estructura dirección multicast de nodo solicitado

Todos los nodos (hosts y routers) se deben unir a las siguientes direcciones multicast:

- FF02::1/8
- FF02:0:0:0:0:1:FFxx:xxxx

Además, los routers deben aceptar los mensajes enviados a la dirección FF02::2/8.

Una interface se puede unir a más de una dirección multicast.

Al igual que sucede en IPv4, una dirección multicast IPv6 se debe mapear a una dirección multicast de Capa 2. Estas direcciones contienen el valor 3333 en sus primeros 16 bits (que se usan para identificar una dirección multicast de capa 2 obtenida a partir de una dirección IPv6) y los restantes 32 bits se copian de los últimos 32 bits de la dirección multicast IPv6. Este proceso se muestra en el siguiente gráfico.

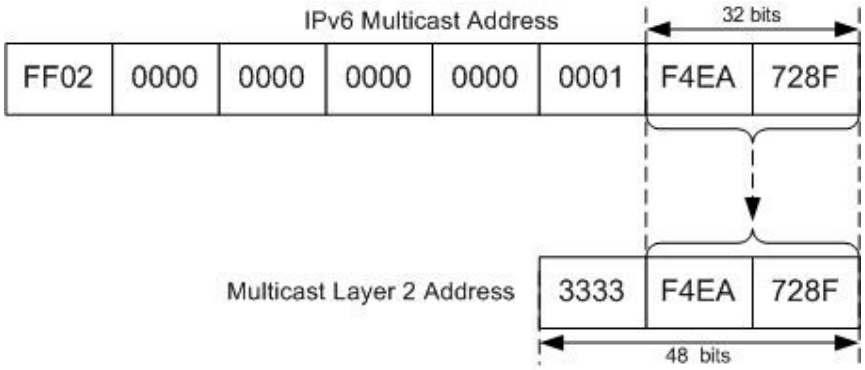



Figura 1.10 - Mapeo direcciones multicast IPv6 en direcciones multicast de capa 2

### 3.2 Tiempo de vida de las direcciones

Las direcciones en IPv6 son asignadas a las interfaces por un período de tiempo determinado, que puede ser infinito. Una dirección puede estar en uno de los siguientes estados:

- Tentativo: la dirección está en el proceso de verificación de su unicidad. Un nodo no debe aceptar paquetes que tienen como dirección destino una dirección en este estado.
- Válido: una dirección a la que se le ha probado su unicidad. Este estado cubre los estados preferido y desaconsejado.
- Preferido: indica el período de tiempo en el que una dirección puede ser usada en forma segura para enviar y recibir tráfico.
- Desaconsejado: el tiempo de vida preferido ha expirado pero la dirección todavía es válida. No se aconseja su uso para establecer nuevas comunicaciones, pero las existentes pueden continuar usándola.
- Inválido: el tiempo de vida válido expiró y no se puede enviar ni recibir tráfico utilizando esa dirección.

La siguiente figura muestra como van cambiando los estados de una dirección a través del tiempo:

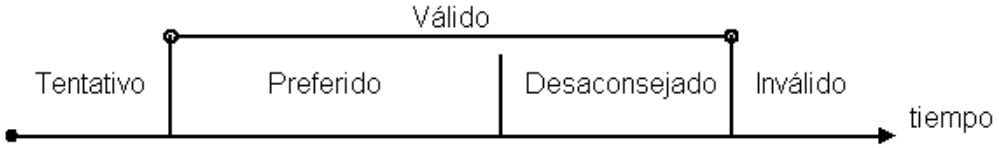



Figura 1.11 - Tiempo de vida de las direcciones

## 4 - ICMPv6

ICMP (Internet Control Message Protocol) en IPv6 cumple las mismas funciones que ICMP en IPv4, es decir, genera mensajes de error (como Destination Unreachable) y mensaje de información (como Echo Request).

Al igual que en IPv4, ICMPv6 debe ser parte integral de IPv6. Esto implica que se debe incorporar a cualquier implementación del protocolo.

Los mensajes de ICMPv6 se agrupan en dos tipos o clases: mensajes de error y mensajes informativos.

En IPv6, ICMPv6 tiene funciones adicionales. Por ejemplo, el proceso Neighbor Discovery, el PMTU (Path Maximum Transmisión Unit) y el protocolo Multicast Listener Discovery lo utilizan para realizar su trabajo.

Un valor decimal de 58 en el campo Next Header de la cabecera IPv6 identifica a un paquete ICMPv6. Un paquete ICMPv6 se encuentra después de todas las cabeceras de extensión de IPv6.

## 5 - Neighbor Discovery

El Neighbor Discovery (ND) es un protocolo que corresponde a una combinación de protocolos de IPv4: el ARP (Address Resolution Protocol), el ICMP Router Discovery y el ICMP Redirect.

El ND posibilita que los nodos (hosts y routers) en un mismo link, anuncien su existencia a sus vecinos y aprendan acerca de ellos. Resuelve un

conjunto de problemas relacionados a la interacción entre nodos en el mismo link. En general, realiza tres funciones principales:

- Provee un mecanismo de resolución de direcciones (que reemplaza al ARP de IPv4)
- Permite a los hosts descubrir cuáles son los routers vecinos que están presentes en el link y provee mecanismos para permitirles obtener cierta información de configuración de ellos
- Mediante el Neighbor Unreachability Detection (NUD) un host puede determinar cuando un vecino se vuelve inaccesible

El ND utiliza el protocolo ICMPv6 para su funcionamiento y define cinco diferentes tipos de paquetes:

- *Router Solicitation*: enviados por los hosts para solicitarle a los routers que generen un mensaje de anuncio de router (Router Advertisement) inmediatamente.
- *Router Advertisement:* los routers anuncian su presencia, junto con varios parámetros del link y de Internet, para que los nodos se autoconfiguren.
- *Neighbor Solicitation:* enviado por un nodo para determinar la dirección de link-layer de un vecino, para verificar que una dirección es única en el link o al ejecutar el Neighbor Unreachability Detection.
- *Neighbor Advertisement:* es en respuesta a una solicitud de vecino. Un nodo, también, puede enviar un anuncio no solicitado (por ejemplo, para anunciar el cambio de su dirección de link-layer).
- *Redirect:* usado por los routers para informar a los hosts de un mejor primer salto a un destino o que éste es un vecino.

### 5.1 - Resolución de Direcciones

El proceso de resolución de direcciones consiste en el intercambio de mensajes Neighbor Solicitation y Neighbor Advertisement para resolver la dirección de link-layer de un nodo vecino. Sólo se ejecuta para las direcciones unicast, no debe hacerse con las direcciones multicast.

Un nodo envía un Neighbor Solicitation a la dirección multicast de nodo-solicitado, la cual se deduce de la dirección IPv6 que se está intentando conocer su dirección de link-layer. El mensaje incluye, como opciones, la dirección de link-layer del nodo emisor (para que el destino sepa a donde contestar y no tenga que ejecutar este proceso nuevamente), y la dirección IPv6 del nodo consultado. Al enviarse la solicitud a la dirección multicast de nodo solicitado, únicamente los nodos que hayan mapeado a esta dirección serán interrumpidos para procesar el mensaje. En IPv4, el protocolo ARP hace este trabajo pero, al utilizar direcciones broadcast todos los nodos en el enlace son interrumpidos.

El nodo, que tiene asignada la dirección IPv6 consultada, le contesta con un mensaje Neighbor Advertisement en el cual incluye su dirección de link-layer

como una opción. Ambos nodos deben actualizar sus caches con la información recibida en los mensajes intercambiados.

El siguiente gráfico muestra como se desarrolla el proceso.

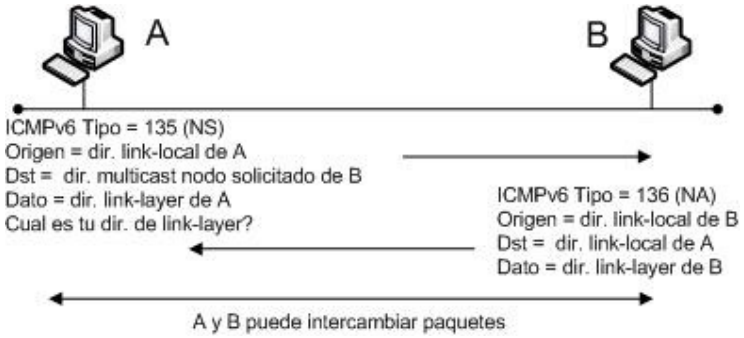



Figura 1.12 - Resolución de direcciones

### 5.2 - Neighbor Unreachability Detection (NUD)

Un nodo vecino es accesible si existe una confirmación, reciente, de que los paquetes IPv6 que se le enviaron los ha recibido y procesado. No garantiza que se tenga acceso al nodo destino porque el vecino puede ser un router, que podría no ser el destinatario final.

Un modo de confirmar la accesibilidad es mediante el envío de un mensaje Neighbor Solicitation a la dirección unicast del vecino y la posterior recepción de un mensaje Neighbor Advertisement (con el flag Solicited seteado a 1 ya que es un anuncio solicitado). Si un anuncio es no solicitado, no puede tomarse como una prueba de que el vecino es alcanzable.

Existe otro método de realizar esta confirmación y es mediante la información intercambiada entre los protocolos de capas superiores que indica que la comunicación, que usa la dirección que se quiere saber si es accesible como primer salto, está progresando. Por ejemplo, en TCP se pueden utilizar los acuses de recibo ya que si hay acceso al destino final también lo hay hasta el primer salto (TCP le debe indicar esto al módulo IP).

### 5.3 - Router Discovery

Router Discovery es el proceso a través del cual los nodos intentan descubrir que routers se encuentran en su link. Es similar al ICMP Router Discovery de IPv4.

Los routers envían mensajes Router Advertisement periódicamente a la dirección multicast de todos los nodos para anunciar su presencia en el enlace. El tiempo entre anuncio es configurable. Los hosts los utilizan para construir su una lista de routers default. Además incluyen otra información como los prefijos de red que los hosts los utilizan para configurar direcciones, el tiempo de vida de los prefijos, el MTU del enlace, el hop-limit, etc.

Si el espacio entre dos anuncios consecutivos es demasiado grande un nodo debería esperar demasiado tiempo para poder autoconfigurarse. Es por esto que un nodo puede solicitar un Router Advertisement envían un mensaje Router Solicitation.

El nodo envía el mensaje Router Solicitation a la dirección FF02::2 que es la dirección multicast de todos los routers y son los únicos que escuchan esa dirección. Le envía su dirección de enlace para que el router no tenga que resolverla al momento de contestarle. Todos los routers habilitados en el enlace le contestarán con un Router Advertisement. En este caso, a diferencia de los anuncios no solicitados, la contestación del router es enviada directamente al host que la solicitó. El siguiente gráfico muestra como es el proceso.

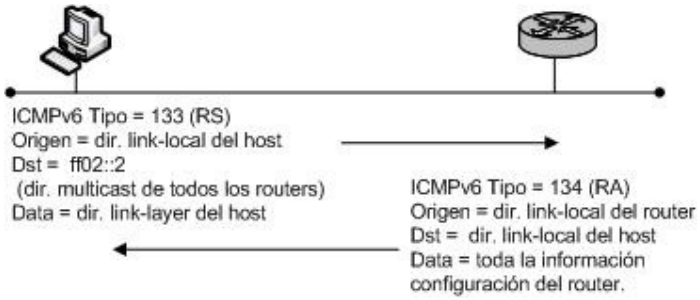



Figura 1.13 - Router Discovery

### 5.4 - Duplicate Address Detection (DAD)

Para asegurar que todas las direcciones sean únicas en un link, los nodos debe ejecutar el proceso Duplicate Address Detection (Detección de Direcciones Duplicadas) antes de asignar una dirección a una interface. Esa dirección permanece en estado tentativo mientras dura el proceso. Este procedimiento debe ser ejecutado sobre todas las direcciones unicast, independientemente de si son obtenidas por autoconfiguración con o sin estado.

El proceso DAD envía un mensaje Neighbor Solicitation a la dirección no especificada (::) porque la dirección que está siendo consultada no puede ser utilizada hasta que se compruebe que es única en el link. Para asegurarse que todos los nodos en el enlace recibieron el mensaje, el host lo puede enviar dos o más veces (es configurable).

Pasado un determinado tiempo sin recibir una respuesta desde que envió la solicitud, el nodo asume que la dirección está libre y la puede utilizar. A partir de este momento, la dirección pasa al estado de válida. En caso contrario, el nodo que la tiene asignada le responderá con un Neighbor Advertisement a la dirección multicast de todos los nodos. En esta situación, el nodo no podrá utilizar esa dirección.

En la figura 1.14 se muestra el proceso. El Host C quiere saber si una dirección IPv6 está duplicada en el enlace. Como esa dirección ya la tiene el Host A asignada no puede ser utilizada por C.

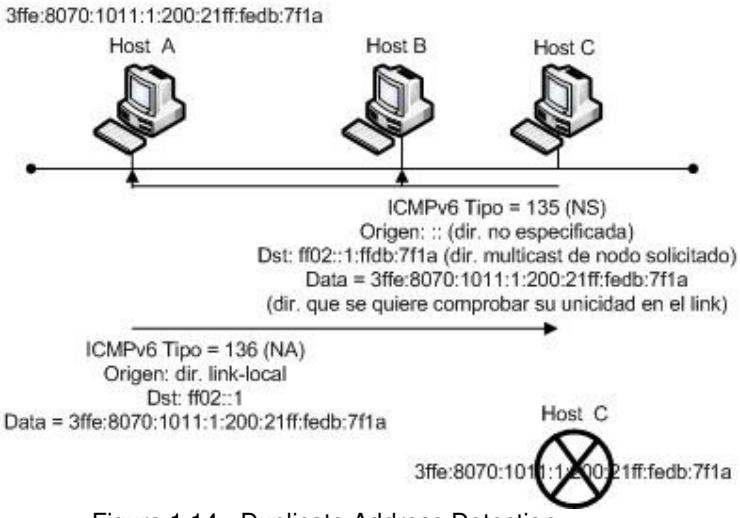



Figura 1.14 - Duplicate Address Detection

### 5.5 - Redirect

Este proceso es similar al que se define para IPv4. Se utiliza para que un router le indique a un host que existe un mejor camino para llegar a un destino determinado. Sólo se necesita un mensaje, el mensaje Redirect, para realizar todo el proceso.

A continuación se muestra como el Router B le indica al Host X que para comunicarse con el Host Y existe un camino más corto a través del Router A.

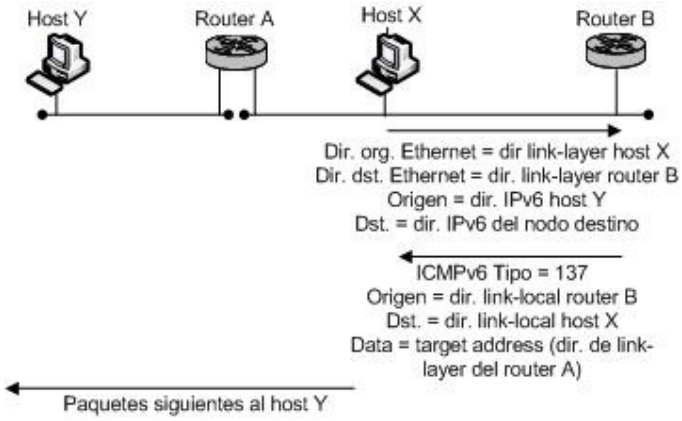



Figura 1.15 - Redirect

### 5.6 - Estructura de un host

Uno de los principios en los cuales el diseño de IPv6 está basado es que los hosts deben trabajar correctamente aún si tienen una visión muy limitada de la red. Durante el inicio, un host se debe autoconfigurar, y luego debe aprender un mínimo de información acerca de los destinos con los cuales intercambiará datos. Esta información está almacenada en memoria en un conjunto de pequeñas estructuras llamadas *cache,* y es válida por un período de tiempo limitado. Estas estructuras son arreglos de registro y cada registro es referido como una entrada.

Se definen cuatro tipos de caches diferentes:

- Neighbor Cache: contiene una entrada por cada vecino a los cuales el nodo le ha enviado tráfico recientemente.
- Destination Cache: contiene una entrada por cada destino a los cuales el nodo le ha enviado tráfico recientemente. La diferencia con la Neighbor Cache es que contiene entradas tanto para destinos on-link como off-link.
- Prefix List: contiene una entrada por cada prefijo on-link, y es utilizada para determinar si una dirección es on-link o no.
- Default Router List: contiene una entrada por cada router que puede ser utilizado como router default.

Los estados de las entradas en la Neighbor Cache pueden ser uno de los siguientes:

- Incomplete: la entrada ha sido creada, pero la dirección de link-layer no ha sido determinada todavía porque el proceso de resolución de direcciones está en progreso
- Reachable: se sabe que la entrada ha sido accedida recientemente.
- Stale: no se sabe si la entrada ha sido accedida recientemente, pero hasta que no se le envíe nuevo tráfico al vecino, ningún intento para comprobar su accesibilidad debería ser realizado.
- Delay: no se sabe si la entrada ha sido accedida recientemente, y se ha enviado tráfico al vecino. En este estado, los mensajes Neighbor Solicitation (probe) son retenidos para permitirle a los protocolos de capa superior confirmar la accesibilidad del vecino.
- Probe: la accesibilidad al vecino es muy incierta, y un mensaje probe ha sido enviado.

## 6 - Autoconfiguración de Direcciones Sin Estado

Un host realiza varios pasos para decidir como autoconfigurar sus direcciones en IPv6. El proceso de autoconfiguración incluye crear una dirección de link-local y verificar su unicidad en el link, determinar que información debe ser autoconfigurada (direcciones, otra información a través del DHCPv6 o ambas). En el caso de las direcciones se debe determinar si serán obtenidas a través del mecanismo con estado, sin estado o ambos.

La autoconfiguración de direcciones sin estado no requiere de una configuración manual de los hosts. Esto permite a los hosts generar sus propias direcciones usando una combinación de información disponible localmente e información anunciada por los routers. Estos anuncian prefijos que indican la/s subredes asociadas con un link, mientras que los hosts generan identificadores de interfaces que identifican únicamente a una interface en un link (o en un alcance mayor). Una dirección es formada por una combinación de los dos. En ausencia de un router, un host solamente puede formar una dirección de link-local en forma automática.

El siguiente gráfico muestra el proceso de autoconfiguración de direcciones sin estado:

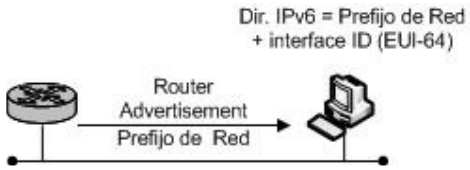



Figura 1.16 - Configuración de direcciones sin estado

Nota: existe otra forma de autoconfiguración, autoconfiguración con estado: DHCPv6.

## 7 - IPv6 en las capas superiores

Todos los protocolos de capa superior que incluyan las direcciones de la cabecera IPv4 para calcular su checksum deberán ser modificados para usar direcciones IPv6 de 128 bits.

En el protocolo UDP, a diferencia de IPv4 donde es opcional, el cálculo del checksum es obligatorio ya que la cabecera IPv6 no contiene este campo.

## 8 - Descubrir el Path MTU (Maximum Transmission Unit)

Como los routers no manejan la fragmentación, ésta es realizada por el nodo origen de un paquete. El proceso de descubrir el path MTU permite conocer el mínimo MTU existente entre el nodo origen, quien ejecuta éste procedimiento antes de enviar el primer paquete, y el nodo destino. Conociendo este valor, los emisores pueden fragmentar los paquetes para enviarlos por ese path.

## 9 - Mecanismos de transición

La migración de IPv4 a IPv6 en un solo día es imposible debido al inmenso tamaño de Internet y al número de usuarios de IPv4. Por esto, no existe un día especial, en el cual, IPv4 se "a*pagará"* y se "*encenderá"* IPv6, sino que será un proceso largo y paulatino, nodo por nodo. No hay un coordinador global ni un orden específico de actualización (por ejemplo, no es necesario actualizar los routers de borde de un sitio antes que los nodos internos).

Ambos protocolos convivirán durante mucho tiempo. La integración y la coexistencia con IPv4 es un requisito para permitir la transición gradual hacia IPv6. Existen un conjunto de mecanismos que pueden implementar los nodos IPv6 con el fin de ser compatibles con los nodos IPv4.

Los siguientes puntos describen algunas de las soluciones disponibles para posibilitar la convivencia de ambos protocolos aunque no sean compatibles. Cada una tiene un conjunto de atributos que son específicos para resolver un problema determinado.

Existen 3 técnicas de transición: Dual-Stack, túneles y mecanismos de translación. Pueden ser usadas en combinación unas con otras.

### 9.1 - Dual-Stack

Éste es el mecanismo de transición más simple. Consiste en proveer a los hosts y los routers un soporte completo para los protocolos IPv4 e IPv6. Cada nodo es configurado con ambos protocolos, por lo cual, pueden interactuar con nodos IPv4 usando mensajes IPv4 y con nodos IPv6 usando paquetes IPv6, sin necesidad de realizar costosos procesos de encapsulación o translación. Para esto deben configurarse con direcciones específicas de cada protocolo. Son llamados "nodo IPv4/IPv6".

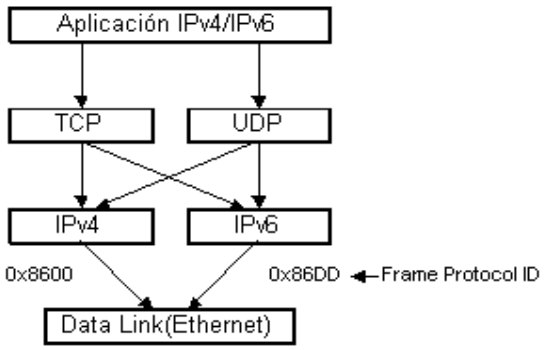



Figura 1.17 - Nodo Dual-Stack

Más que un mecanismo de transición es un mecanismo de integración.

### 9.2 - Túneles

Mientras esté siendo desarrollada, la infraestructura de ruteo para IPv6 estará basada en la provista por IPv4. La técnica del túnel es un proceso en el que la información de un protocolo es encapsulada en un paquete de otro protocolo, lo que se conoce como encapsulación. Permite que redes IPv6 aisladas se puedan comunicar sin necesidad de actualizar la estructura de ruteo IPv4 entre ellas

A continuación se muestra un ejemplo de túnel. Los nodos extremos del túnel deben ser *dual-stack.*

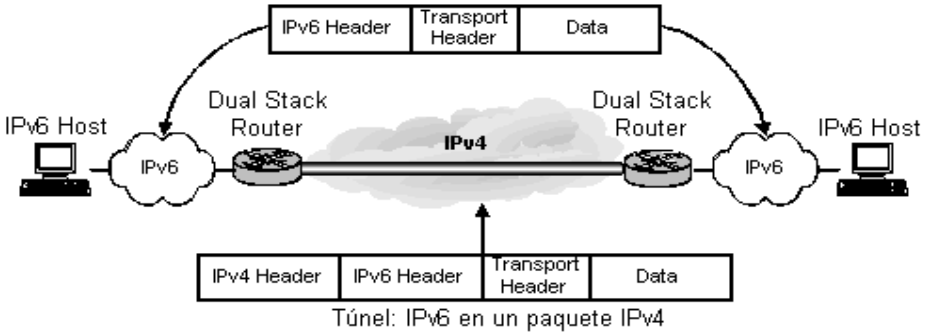



Figura 1.18 - Túnel IPv6 en IPv4

La figura 1.18 muestra como se componen los paquetes IPv6, encapsulados por un router dual-stack, para enviarlos por una red IPv4. El router origen del túnel le agrega una cabecera IPv4, en la cual, las direcciones origen y destino son las correspondientes a las de inicio y fin del túnel. El router donde finaliza el túnel es el encargado de desencapsular el paquete IPv6 (eliminando la cabecera IPv4) y retransmitirlo hacia el destino final.

Los túneles pueden ser configurados o automáticos. La diferencia que presentan es si deben ser configurados manualmente o no.

### 9.2.1 - Túneles Configurados

Esta clase de túneles punto-a-punto necesitan configuración manual por parte de los administradores de la red y el punto final del túnel es determinado por la información de configuración ingresada en el nodo que encapsula. Obviamente, ambos extremos del túnel deben ser configurados. Todos los túneles manuales son bidireccionales. Por lo tanto, un nodo debe mantener información por cada túnel que se le configura.

Estos túneles son creados, en general, para comunicar en forma permanente dos redes IPv6 aisladas usando la infraestructura de ruteo IPv4. Los dos extremos del túnel debe ser dual-stack y, aunque esos extremos pueden ser un host y un router, usualmente son dos routers.

### 9.2.2 - Túneles Automáticos

Los túneles automáticos permiten a los nodos enviar tráfico IPv6 por una red IPv4 sin tener que preconfigurar, manualmente, un túnel.

Al utilizar direcciones IPv6 compatibles con IPv4, el nodo encapsulador puede determinar la dirección IPv4 del final del túnel automáticamente de la dirección IPv6, y por lo tanto, no se necesita configuración manual de los extremos del túnel que, obviamente, deben ser dual-stack.

El siguiente gráfico muestra como se forman estas direcciones, donde los últimos 32 bits corresponden a la dirección IPv4 del nodo destino.

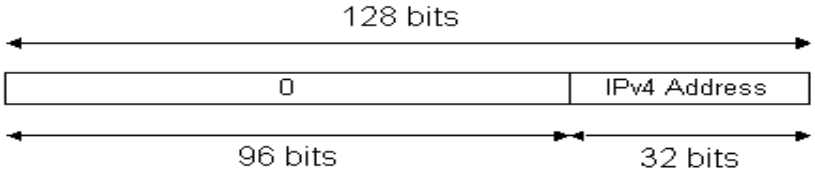



Figura 1.19 - Estructura dirección IPv4-compatible IPv6

Una dirección IPv4-compatible es globalmente única siempre que la dirección IPv4 no pertenezca al espacio de direcciones privadas de IPv4.

En el gráfico a continuación se muestra como un host que soporta este tipo de direcciones, al enviar un mensaje a la dirección ::A319:021E, el nodo encapsulador determina la dirección IPv4, 163.25.2.30, automáticamente de la dirección IPv6.

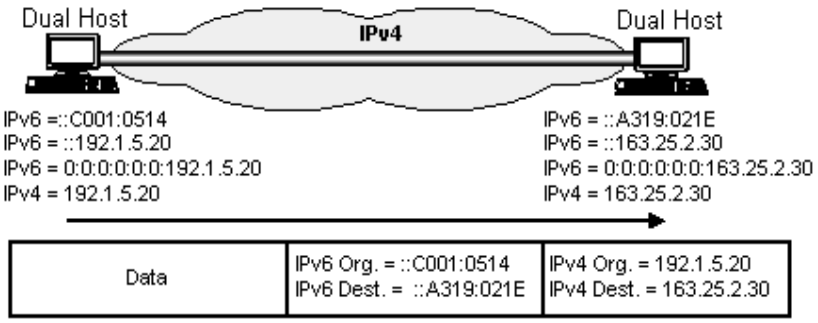



Figura 1.20 - Túneles Automáticos

### 9.2.3 - Túneles 6to4

Este método es un tipo de túnel automático *router-to-router*. Comienzan con el prefijo 2002::/16, y los siguientes 32 bits son ocupados por la dirección IPv4 del router. Esto permite construir un prefijo /48, con lo cual, una organización dispone de los siguientes 16 bits para administrar localmente.

Para poder comunicarse por Internet, la dirección IPv4 debe ser única globalmente.

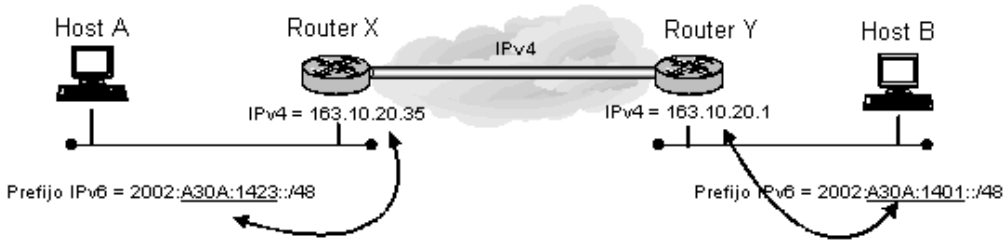



Figura 1.21 - Túnel 6to4

Un router utiliza la dirección IPv4 embebida en la dirección destino IPv6 para determinar el otro extremo del túnel.

Para establecer este tipo de túneles cada dominio IPv6 requiere un router dual-stack que automáticamente construya el túnel. Estos routers reciben el nombre de *6to4 Relay Router*. Se recomienda que cada router tenga una sola dirección 6to4 asignada a su interface externa. Dentro del sitio se puede utilizar un protocolo de ruteo IPv6, fuera de éste se continúan utilizando los protocolos de ruteo IPv4. De esta forma, un número arbitrario de sitios 6to4 pueden comunicarse entre ellos sin necesidad de configurar túneles manuales.

Difieren de los túneles automáticos, explicados en el punto anterior, en que este método permite formar prefijos de red y direcciones para un único host (aunque se recomienda que sea utilizado para configurar prefijos), mientras que el anterior, únicamente permite formar direcciones asignables a un nodo en particular.

### 9.2.4 - 6over4

Esta alternativa, definida en la RFC 2529, permite que nodos IPv6 aislados se conecten, en un mismo link, utilizando direcciones multicast IPv4 mediante túneles automáticos. El túnel establecido es como un link virtual entre los nodos 6over4 en una red IPv4. Las direcciones IPv4 multicast son usadas para ejecutar el proceso Neighbor Discovery. No ha tenido mucho éxito y su utilización se desaconseja.

### 9.2.5 - Tunnel Broker

En esta solución, un nodo dual-stack, ubicado en un red IPv4, se conecta a un servidor web, ingresa información de autenticidad y recibe un script que al ejecutarlo establece un túnel IPv6-en-IPv4 al servidor tunnel broker. Éste puede ser visto como un ISP IPv6 virtual.

A diferencia de la solución 6to4, el Tunnel Broker es adecuado para pequeños sitios IPv6 aislados o hosts IPv6 en una red IPv4 que quieren comunicarse con una red IPv6.

## 9.3 - Mecanismo de translación

Para que un nodo en una red IPv6 se puede comunicar con un nodo remoto en una red IPv4 debe usar los mecanismos de translación. Dentro de estos, se encuentra NAT-PT, Network Address Translation - Port Translation, que realiza un mapeo de direcciones IPv6 en direcciones IPv4 modificando la cabecera de los paquetes. Este proceso es similar al NAT tradicional realizado entre direcciones públicas y privadas del protocolo IPv4.

# Capítulo 2

# Redes wireless: el estándar 802.11 y sus enmiendas

En estos últimos años se está extendiendo, a un ritmo exponencial, el uso de las redes wireless, o inalámbricas, en todos los ámbitos de nuestra vida cotidiana. Son implementadas en los más diversos lugares y pueden ser utilizadas por las más diversas aplicaciones.

Este tipo de redes deben ser consideradas como una extensión de las redes tradicionales y no como su reemplazo. Su principal ventaja es que permiten una movilidad limitada de los usuarios de la red sin que pierdan conectividad. Además, posibilitan ampliar el alcance de las redes a lugares donde sería imposible llegar con las redes cableadas.

Existen varias tecnologías de redes inalámbricas y todas ellas con distintas prestaciones. Este trabajo está basado en el estándar 802.11 de la IEEE y sus derivados: 802.11a, 802.11b y 802.11g; basándose principalmente en el estándar 802.11b.

## 1 - Historia

En el año 1997, después de 7 años de trabajo, la IEEE lanzó el estándar 802.11, que es parte de su familia de estándares 802. Esto significa que su arquitectura es similar a las demás redes definidas dentro de dicha familia, especialmente al estándar 802.3, lo que ha llevado a que mucha gente llame a las redes 802.11: Wireless Ethernet. Como todos los estándares de la IEEE 802, el 802.11 se ocupa de las dos capas inferiores del modelo OSI, la capa física y la de enlace. Los protocolos de la capa de red, tipo IP o IPX, deberían correr en una red de este tipo de igual manera a como lo hacen sobre una red Ethernet.

Los productos basados en el primer documento fueron lanzados al mercado en el año 1997. En su definición original el estándar incluía tres capas físicas de radio diferentes. Una capa infrarroja (IR), que nunca fue muy desarrollada y fue eliminada en las siguientes definiciones del estándar, y dos técnicas de radio de 'spread spectrum' que operaban en la banda de frecuencias de 2,4 GHz, conocida con el nombre ISM (Industrial, Scientific and Medical). Esta banda está habilitada, por distintas agencias internacionales de regulación como la FCC (Estados Unidos), ETSI (Europa) y MKK (Japón), para ser utilizada sin ningún tipo de licencia.

Las velocidades de transferencia definidas inicialmente eran de 1 y 2 Mbits por segundo. Éstas eran velocidades muy bajas para lograr que las redes inalámbricas fuesen ampliamente aceptadas por el mercado. A esto se le

sumaba el problema de la seguridad. Las redes wireless eran muy vulnerables a todo tipo de ataque. Para solucionar ambas problemáticas, la IEEE decidió desarrollar enmiendas al estándar original. Para el primer problema, dos nuevos documentos vieron la luz en 1999, el 802.11a y 802.11b. Para el de la seguridad, la solución vendría años más tarde con el documento 802.11i.

Tanto la enmienda 802.11a como la 802.11b sólo afectan la capa física del estándar original permitiendo velocidades de transferencias más altas. La arquitectura del protocolo y los servicios continúan iguales a los definidos en el documento original.

## 2 - Introducción

Antes de empezar a ver las capas que constituyen un dispositivo compatible con el estándar 802.11 es necesario hacer una introducción a algunos conceptos que se incorporan en esta clase de redes. A pesar de tener un funcionamiento similar a la 802.3, 802.11 agrega una importante cantidad de nuevas características.

Al ser miembro de la familia IEEE 802, la 802.11 pertenece al conjunto de las redes de área local (LAN). Todas estas especificaciones se ocupan de las dos capas inferiores del modelo OSI, la capa física y la capa de enlace, lo que se puede apreciar en el siguiente gráfico.

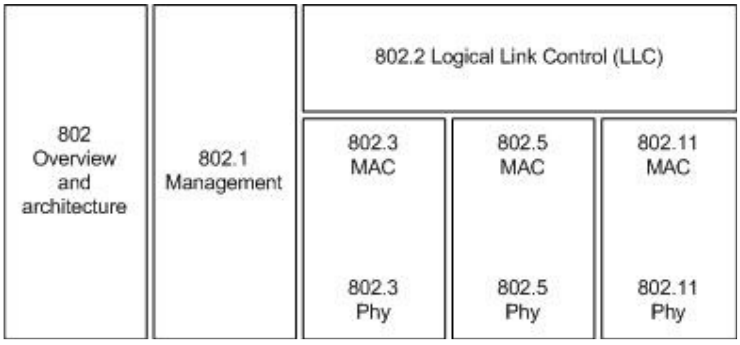



Figura 2.1 - Familia de estándares 802

Todas las redes 802 tienen una capa física y una capa MAC. Esta última es un conjunto de reglas que indican como acceder al medio para enviar datos, pero los detalles de la transmisión pertenecen a la capa física. Ésta se encarga de poner la información en el medio de transmisión.

Un grupo de estaciones conectadas entre sí en forma inalámbrica se lo conoce como 'basic service set' (BSS, Conjunto de servicios básicos) y al área que abarca ese BSS se la conoce como 'basic service area' (BSA, área de servicio básico). Existen dos tipos diferentes de BSS: Independent Basic Service Set (IBSS) e Infrastructure Basic Service Set.

En un IBSS, las estaciones se comunican directamente entre si, lo cual implica que dos estaciones deben estar dentro del mismo área para establecer una conexión entre ellas. El control de la red se realiza entre todas las

estaciones que conforman la red. No existe un controlador central. Este tipo de redes, generalmente, se crean con un fin específico y durante un periodo de tiempo determinado. Por ejemplo, varias personas que quieran intercambiar información pueden formar una red IBSS. Al finalizar el encuentro, la IBSS se disuelve. A las redes de este tipo se las conoce informalmente como redes *ad-hoc*.

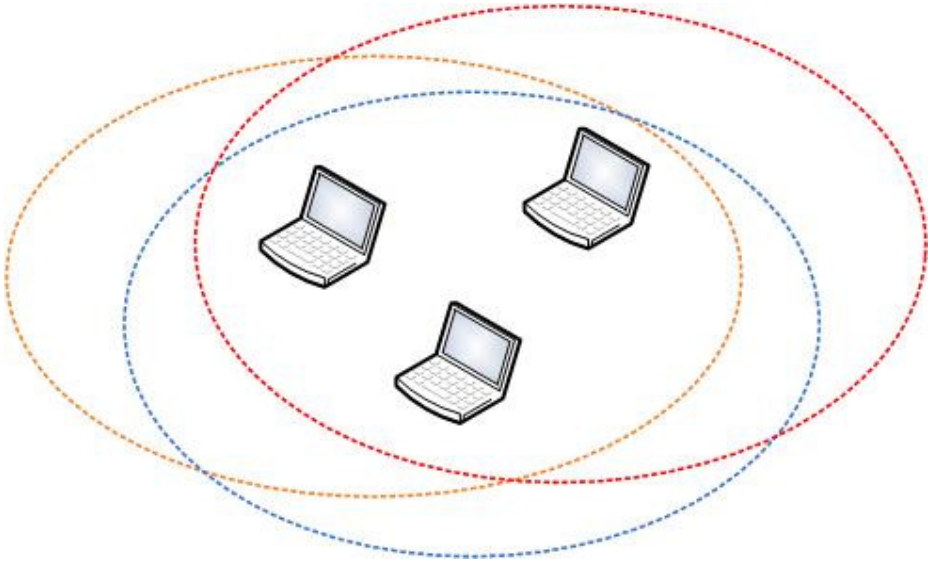


Figura 2.2 - Red IBSS (Independent Basic Service Set) o ad-hoc

En cambio, una red Infraestructure BSS se distingue de una IBSS por el uso de un dispositivo especial llamado access point (AP, Punto de Acceso). Este dispositivo es un intermediario en toda conexión entre dos estaciones cualquiera, incluyendo las comunicaciones entre dos estaciones wireless pertenecientes al mismo BSS. En este caso, el nodo wireless origen transmite la trama al access point, que a su vez la retransmite al nodo wireless destino. Además, funciona como un intermediario entre los dispositivos en el medio cableado y los que se encuentran en el medio inalámbrico. El área de una red infraestructura está definida por el alcance del access point, lo que significa que todas las estaciones deben estar dentro su área de cobertura, no existiendo alguna restricción con respecto a la distancia entre las estaciones que componen un mismo BSS. Aunque dos estaciones no puedan comunicarse directamente porque están fuera del alcance, si lo podrán hacer a través del access point al que están asociados.

Antes de poder utilizar un access point las estaciones deben asociarse al mismo. Este proceso es siempre iniciado por las estaciones, nunca por el access point, y un access point puede denegarle el acceso a una estación. Una estación sólo puede estar asociada a un access point por vez. El estándar no limita la cantidad de estaciones que pueden asociarse a un access point (esto puede ser limitado por las distintas implementaciones). Al no existir un access point, la asociación no es necesaria en una red IBSS.

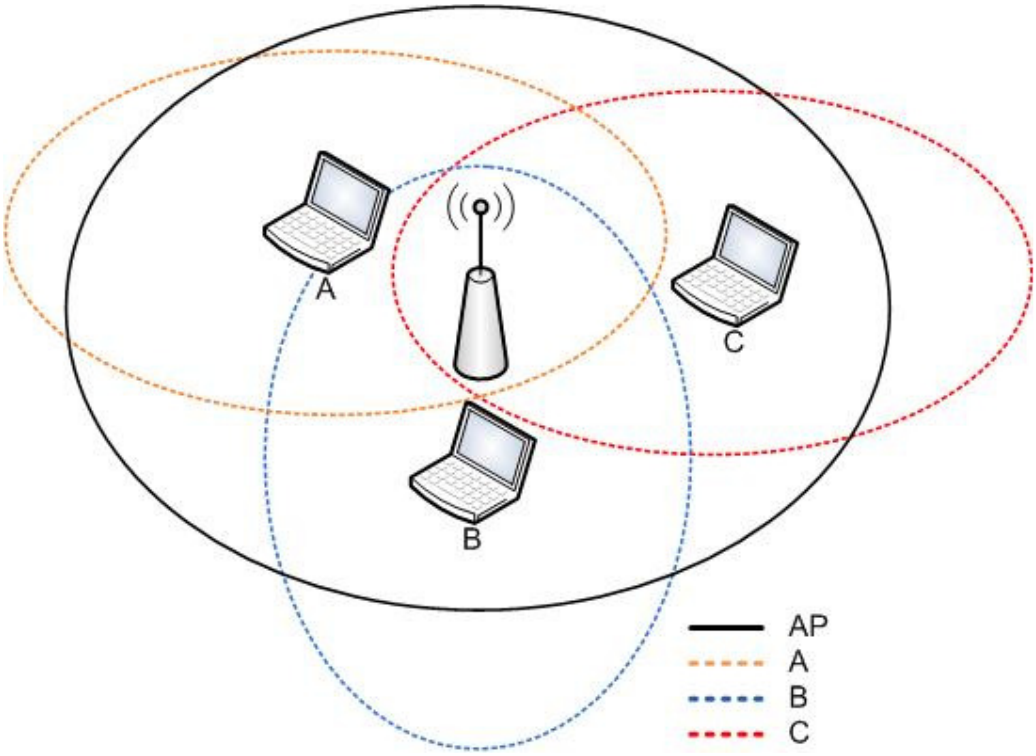



Figura 2.3 - Red Infraestructure Basic Service Set

Para cubrir un área mayor a la que puede ser abarcada por un único BSS, se pueden encadenar varios BSS por medio de una red de backbone llamada Distribution System (DS, Sistema de Distribución). Este conjunto de BSSs unidos se lo conoce como Extended Service Set (ESS, Conjunto de Servicios Extendidos). Con el fin de que los usuarios se puedan mover sin perder conectividad con la red entre los distintos BSSs, estos deben tener un cierto grado de superposición. Cuando una estación necesita cambiarse de access point, antes de asociarse al nuevo access point, debe desasociarse del que estaba asociado.

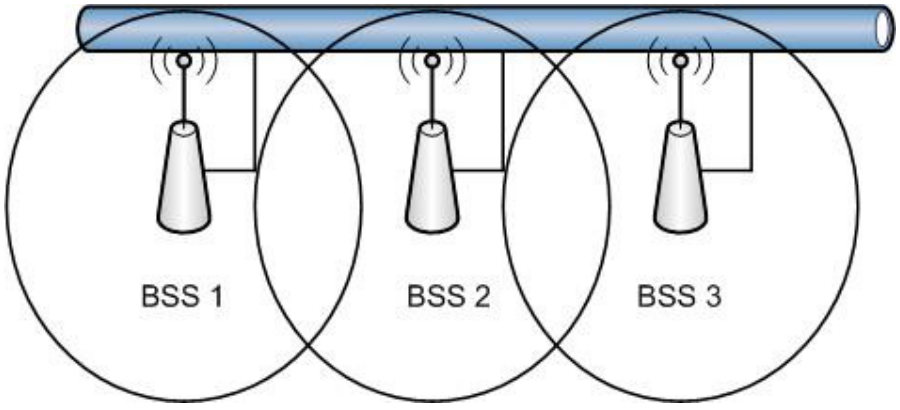



Figure 2.4 - Extended Basic Service Set (EBSS)

Las estaciones dentro del ESS se pueden comunicar con las demás estaciones sin importar en que BSS se encuentran. Si dos estaciones en

diferentes BSS se quieren comunicar entre si, cada estación debe enviar sus tramas al access point al que está asociado, que a su vez las reenviará, a través del DS, al access point al cual está asociado el nodo destino.

El movimiento de una estación desde un ESS a otro no está soportado en el estándar 802.11. En realidad, el movimiento está permitido pero no en forma transparente como sucede internamente en un ESS. Al producirse el movimiento, las conexiones de las capas superiores serán interrumpidas. Por ejemplo, si los protocolos de capa superior son TCP/IP es necesario tener soporte para Mobile IP con el fin de asegurar que las conexiones activas no se interrumpan.

## 3 - Capa Física

Como se dijo más arriba, la 802.11 original define tasas de transferencia de 1 y 2 Mbps por segundo usando ondas de radio con dos técnicas de spread spectrum: Frequency Hopping Spread Spectrum (FHSS) y Direct Sequence Spread Spectrum (DSSS). Éstas son dos técnicas de señalización diferentes y no son capaces de interoperar una con otra.

Con FHSS, la banda de 2,4 GHz es dividida en 75 subcanales de 1 MHz cada uno. El emisor y el receptor se van moviendo constantemente entre los distintos subcanales en forma simultánea y coordinada, transmitiendo una ráfaga corta de información en cada subcanal. Esta técnica es muy simple pero no permite velocidades mayores a las 2 Mbits/seg (restricción impuesta por la FCC que limita el ancho de banda de los subcanales a 1MHz). Una ventaja de estos sistemas es que son muy baratos de construir.

En cambio, la técnica de DSSS divide la banda de los 2,4 GHz en 14 subcanales de 22 MHz cada uno. Los canales adyacentes se superponen parcialmente, dejando como máximo solamente 3 canales sin hacerlo. El canal 1 se suporpone con los canales 2, 3, 4 y 5. El canal 2 lo hace con 3, 4, 5 y 6. Y así sucesivamente. Solo los canales 1, 6 y 11 no se superponen. Por lo cual, si se quieren instalar varias redes inalámbricas, el problema de la interferencia entre los canales debe ser tenido en cuenta. Los datos son enviados usando uno de estos subcanales sin necesidad de saltar a otro. A diferencia de FHSS, este método requiere un proceso más sofisticado de procesamiento de las señales, lo que requiere un hardware más específico. La ventaja es que permite mayores velocidades de transferencia que FHSS.

La modificación realizada por la 802.11b consiste, básicamente, en agregar dos nuevas velocidades de transferencia: 5,5 Mbps y 11 Mbps. Para lograr esto, se debe utilizar la técnica de DSSS. FHSS no puede lograr estas velocidades sin violar las restricciones impuestas por la FCC. Este nuevo método se conoce como High-Rate Direct Sequence Spread Spectrum (HR-DSSS). Los dispositivos 802.11b deben tener la capacidad de interoperar con los dispositivos 802.11.

El estándar 802.11a incluye unas cuantas modificaciones. En primer lugar, ya no utiliza la banda 2,4 GHz para funcionar sino que utiliza un rango

ubicado en los 5 GHz que también es una banda no licenciada. Pero ésta no es la única innovación incluida en la enmienda. También se agregó una nueva técnica de señalización conocida como OFDM (Orthogonal Frequency División Multiplexing) que permite alcanzar velocidades de transferencia de hasta 54 Mbps/seg. Lógicamente, no existe interoperabilidad posible entre los dispositivos 802.11a y 802.11 u 802.11b.

A continuación se muestra un gráfico donde se pueden observar las distintas capas físicas:

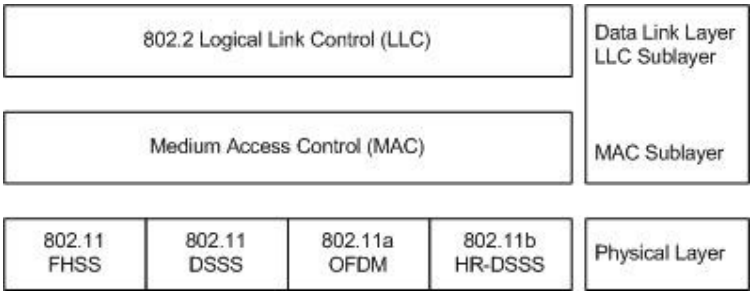



Figura 2.5 - Capa física 802.11

En la figura 2.6 se muestra la arquitectura lógica de una red 802.11. Como se puede observar, la capa física está dividida en dos subcapas: la Physical Layer Convergence Procedure (PLCP) y la Physical Medium Dependent (PMD). La PLCP es la capa que hace de *pegamento* entre las tramas de la capa MAC y la transmisión de las mismas usando ondas de radio. Además, agrega campos en las cabeceras de las tramas antes de ser transmitidas, formando lo que se conoce como PLCP Protocol Data Unit (PPDU). La PMD es la responsable de transmitir cualquier información que recibe de la PLCP. Otra función que se encuentra en la capa física es la 'clear channel assessment', que sirve para indicar a la capa MAC cuando se ha detectada una señal en el medio.

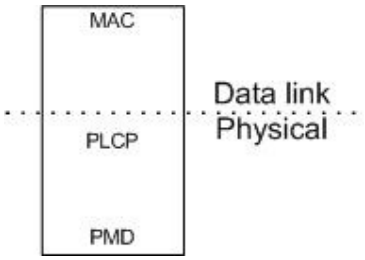



Figura 2.6 - Arquitectura lógica de una red 802.11

Años más tarde, en el 2003, la IEEE lanzó el estándar 802.11g que puede alcanzar hasta las 54 Mbits/seg. Funciona en el mismo rango de frecuencias que lo hace el estándar 802.11b y son compatibles entre si. Un nodo 802.11g puede interactuar con un nodo 802.11b. Este último determina la velocidad máxima a la que se puede realizar la conexión. Obviamente, en este caso no puede ser superior a los 11 Mbits/seg.

## 4 - Capa de enlace

Al igual que 802.3 (Ethernet) u 802.5 (Token Ring), la capa de enlace de las redes wireless está compuesta por dos subcapas: Logical Link Control (LLC, Control lógico del enlace) y Media Access Control (MAC, Control de acceso al medio). La capa LLC es similar a la que usan las redes LAN 802, lo que permite que los protocolos de la capa superior utilicen las redes 802.11 de la misma forma que lo hacen con las otras redes definidas en la familia 802.

Esta capa se ubica sobre la capa física y es la que decide cuándo se pueden transmitir los datos que recibe de las capas superiores. Además, provee las operaciones de entramado, control de errores, fragmentación, etc. El estándar 802.11 adapta exitosamente el funcionamiento de las redes Ethernet.

Para controlar el acceso al medio, 802.11 utiliza un esquema similar al que se utiliza en Ethernet. Pero, a diferencia de las redes Ethernet, que son capaces de detectar las colisiones, en una red 802.11 se las trata de evitar porque los dispositivos tienen dificultad para detectarlas o les resulta imposible hacerlo. El método se conoce como Carrier Sense with Multiple Access with Collision Avoidance (CSMA/CA, Acceso Múltiple con Detección de Portadora con Evitación de Colisiones). Esto significa que una red 802.11 usa un esquema de acceso distribuido, es decir, nadie tiene el control total de la red, cada estación usa el mismo método de acceso para obtener el medio y así poder transmitir.

En una red tipo Ethernet es muy simple detectar cuando se produce una colisión. La estación transmisora escucha el medio mientras transmite, lo que le permite detectar si se produce una colisión y tomar las medidas apropiadas. Como esto es muy difícil de realizar en una red 802.11 es necesario contar con otro mecanismo que permita asegurar al emisor que los paquetes que envía llegan correctamente al destino. Es por esto que se debe enviar un paquete de confirmación por cada paquete unicast que se envía al medio. Esto produce un overhead en la red con la consecuente baja del rendimiento de la misma. Por ejemplo, el intercambio de un paquete entre dos estaciones inalámbricas requiere el intercambio de un total de 4 paquetes. El siguiente gráfico muestra la secuencia.

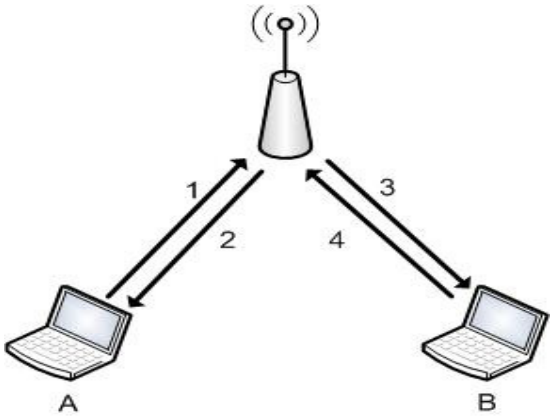



Figura 2.7 - Intercambio de paquetes entre dos estaciones wireless

Cuando la estación A le quiere enviar un mensaje a la estación B, como se explicó anteriormente, lo debe hacer a través del access point. En primer lugar, la estación A le envía el mensaje al access point que le contesta con un paquete de confirmación, un paquete Ack, en el paso 2. Esta secuencia es una operación atómica, es decir, que no puede ser interrumpida. Si alguno de los dos pasos falla, el mensaje se considera perdido y deberá ser retransmitido. Luego, el access point reenvía el paquete a la estación B quien debe confirmar la recepción del paquete, lo que hace en el paso 4 con otro mensaje ACK.

### 4.1 - Problema del Nodo Oculto

En una red cableada, cuando un nodo envía un mensaje al medio, éste se encarga de que la señal generada sea recibida por todas las estaciones. Una estación es capaz de comunicarse con cualquier otra estación de la misma red. Las redes wireless nos enfrentan al problema de que dos estaciones pertenecientes a la misma red, o BSS, no puedan comunicarse entre si. Esto se conoce como problema del nodo oculto (hidden node problem) y se muestra en el siguiente gráfico:

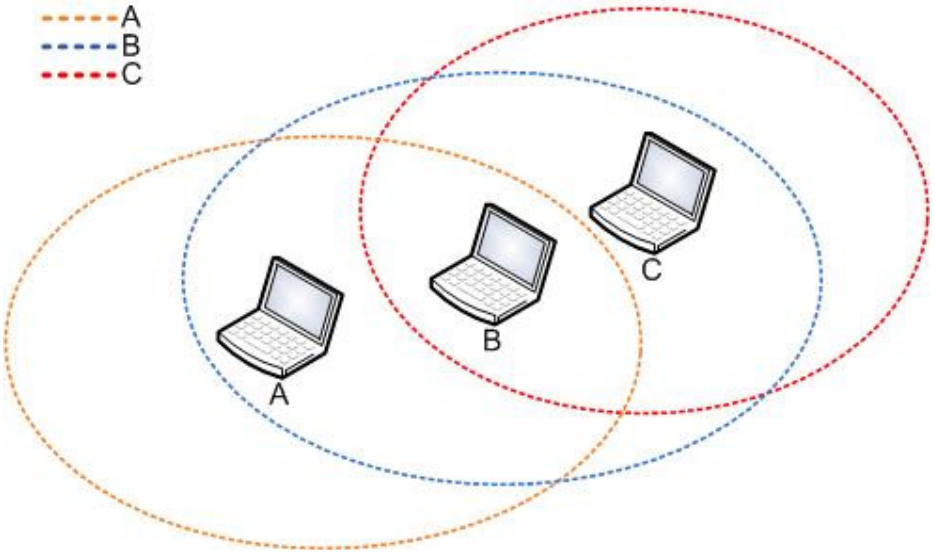



Figura 2.8 - Problema del nodo oculto

Como se puede observar en la figura anterior, la estación B se encuentra en el área de cobertura de la estación A pero no sucede lo mismo con la estación C. Toda transmisión desde A será escuchada por B pero no por C. Algo similar ocurre con toda transmisión originada en C, será recibida por B pero no por A. El problema surge cuando, por ejemplo, A y C realizan una transmisión simultánea a B. Ambos mensajes colisionarán. Esta colisión es local a B, ninguno de los dos nodos transmisores pueden detectarla. Sólo pueden asumirla al no recibir el correspondiente mensaje de confirmación de la estación B. Desde el punto de vista de A, el nodo C es un nodo oculto (lo contrario sucede con el nodo C).

Con el fin de evitar las colisiones se introducen dos nuevos mensajes: Request to Send (RTS, Solicitud para enviar) y Clear to Send (CTS, Limpio

para enviar). Como se puede observar en el siguiente gráfico, cuando una estación desea transmitir (en este ejemplo, la estación A) primero envía un mensaje RTS con el fin de reservar el medio por el tiempo necesario para realizar la transmisión de la trama y recibir la correspondiente confirmación de recepción de parte del receptor. Pero, como se puede observar, el alcance de las transmisiones de la estación A no llega hasta la estación B. Al recibir un mensaje RTS, el access point contesta con un mensaje CTS. Este mensaje, que si es recibido por todas las estaciones dentro de la red, porque es enviado por el acces point que es el que define el área de cobertura de la red, provoca que B no intentará transmitir mientras lo hace A, ni tampoco lo intentará hacer cualquier otra estación asociada al access point. Con el mensaje RTS, la estación A hace una reserva del medio por un tiempo determinado y, con el correspondiente CTS, el access point se encarga de avisarles a todas las estaciones que se encuentran fuera del alcance de A que no realicen una transmisión durante el tiempo indicado por A.

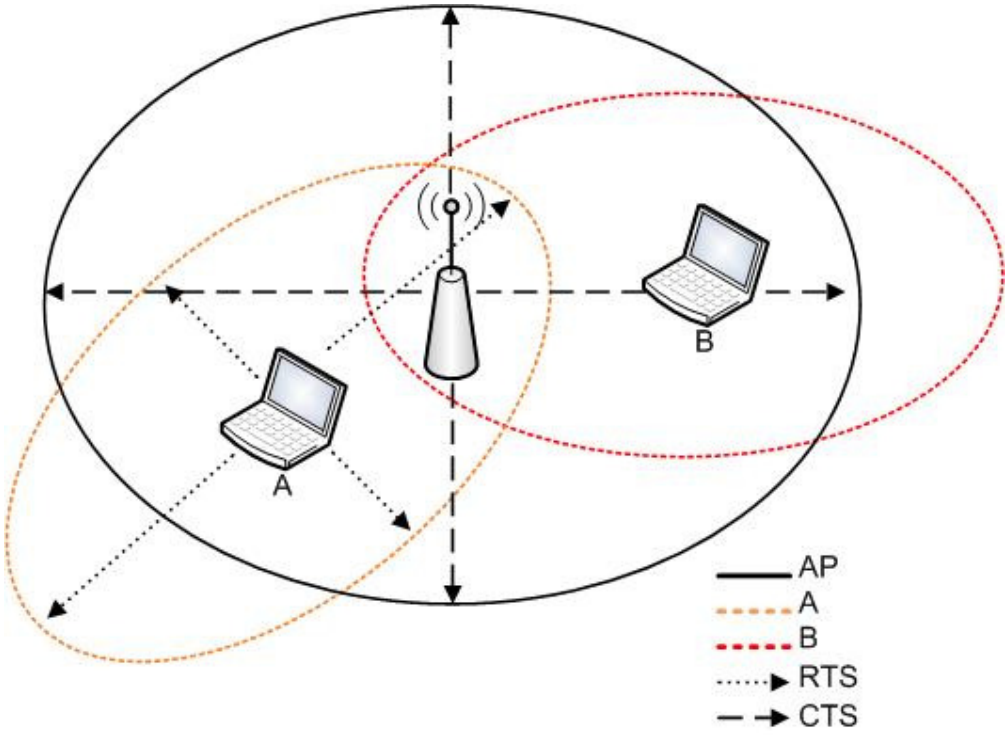



Figura 2.9 - Reserva del medio usando mensajes RTS/CTS

El siguiente gráfico muestra el intercambio completo de mensajes. En total se requiere un total de cuatro mensajes para completar todo el proceso. Si la estación A le quiere enviar un mensaje a B, inicia el proceso enviando un frame RTS. La estación receptora le contesta con un CTS, habilitándolo para enviar el frame, lo que hace en el paso siguiente. Por último, se envía un Ack para confirmar la recepción correcta del frame.

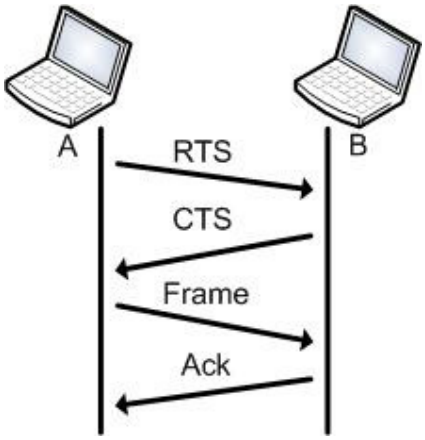



Figura 2.10 - Intercambio completo de paquetes con RTS/CTS

Esta característica es una mejora al método CSMA/CA y, si el dispositivo lo permite, los usuarios pueden habilitarlo cuando lo deseen. Para esto existe un parámetro llamado RTS threshold que indica, que para toda trama que se desea enviar con un tamaño mayor a este umbral, se debe realizar, previamente, el intercambio de tramas RTS/CTS.

Aunque permite solucionar un problema inherente a las redes wireless, el intercambio de estas tramas disminuye el rendimiento de la red. Su habilitación se recomienda en entornos en los cuales existen una gran cantidad de colisiones y se desaconseja para tramas de un tamaño pequeño. Además, su utilización está restringido únicamente a los mensajes unicast. No debería ser usado con paquetes multicast o broadcast.

### 4.2 - Modos de acceso al medio

El acceso al medio es controlado por funciones de coordinación. Una función es similar a la que tiene Ethernet, ninguna estación tiene el control de la red y todas deben competir entre sí para obtener el acceso al medio, y se llama Distributed Coordination Function (DFC, Función de coordinación distribuida). El otro método requiere un access point y recibe el nombre de Point Coordination Function (PCF, Función de punto de coordinación) y, a diferencia de DCF, es un servicio libre de contención, lo que implica que las estaciones no deben competir entre ellas para acceder al medio.

En DCF, el mecanismo de acceso al medio es CSMA/CA. Como sucede en Ethernet, antes de empezar a transmitir una estación debe verificar que el medio no esté ocupado (que no haya otra estación transmitiendo). Si el medio está libre, puede transmitir después de esperar un tiempo determinado. Si el medio está ocupado, o se produce una colisión, se ejecuta un algoritmo de backoff para determinar qué estación es la siguiente en transmitir. Para mejorar el funcionamiento de este método, en circunstancias determinadas, se pueden utilizar dos paquetes adicionales, RTS (Request to Send) y CTS (Clear to Send). DCF debe ser implementado en todas las estaciones 802.11 y se lo puede utilizar en cualquiera de las dos configuraciones de red posible (infraestructura o ad-hoc).

PCF está construido sobre DCF y solamente se puede ejecutar en el modo infraestructura. La función PCF se ejecuta en un nodo especial llamado Point Coordination (PC, Punto de Coordinación), que reside en el access point, y es el que determina qué estación tiene derecho a transmitir. El PC consulta a las estaciones para ver si alguna tiene tramas para transmitir. Al hacer que las estaciones transmitan por turno se crea un método de acceso libre de contención y de colisiones.

Una red en modo infraestructura puede ofrecer únicamente el servicio DCF pero no puede ofrecer solamente el servicio PCF. Si desea ofrecer este último, debe brindar ambos métodos. Es decir, la red puede funcionar siempre en el modo DCF pero, si desea funcionar en PCF, debe alternar entre los dos métodos. Durante un intervalo de tiempo funciona en modo DCF y en el siguiente lo hace en PCF, y así, sucesivamente. Los dos métodos de acceso se van alternando, creando un intervalo con contenciones y luego otro libre de contenciones, en forma sucesiva y alternada. En un mismo BSS pueden existir estaciones funcionando en DCF y en PCF simultáneamente. Una estación utilizando el método PCF tiene prioridad sobre otra que usa DCF.

### 4.3 - Mecanismo de comprobación del medio

Antes de empezar a transmitir, una estación debe comprobar que el medio no esté ocupado. 802.11 define dos tipos de funciones para realizar esta tarea, una forma física y otra virtual. La primera está provista por la capa física y la segunda por la capa MAC.

Debido a lo costoso que era construir hardware que pudiese transmitir y recibir simultáneamente, las estaciones no podían escuchar el medio mientras transmitían, como sucede en Ethernet. Esta limitación no le permitía a las estaciones detectar si se había producido una colisión mientras transmitían. Pero, aun con un hardware que eliminase esta restricción, el método físico tiene algunas limitaciones para determinar si el medio está ocupado, bajo ciertas condiciones, como sucede con el problema del nodo oculto.

El método virtual está provisto por un temporizador llamado Network Allocation Vector (NAV, Vector de Reserva de Red). Su trabajo se basa en hacer una predicción estimada del tiempo necesario para realizar una determinada transmisión. Es decir, se calcula cuánto es el tiempo que le llevará a una estación realizar una transmisión exitosa contemplando todo el intercambio de tramas necesario para realizarla. La mayoría de las tramas 802.11 contienen un campo llamado Duration que se puede utilizar para reservar el medio por un tiempo determinado. El NAV es un valor que indica por cuanto tiempo el medio va a estar ocupado. Cuando las demás estaciones reciben el NAV empiezan a decrementar su valor hasta llegar a 0. Cuando esto sucede pueden asumir que la transmisión concluyó y que el medio está libre.

El siguiente gráfico muestra el funcionamiento del método virtual. Como se puede observar, el emisor hace uso de una trama RTS en la cual envía el valor del NAV. El receptor le contesta con un CTS en el cual también envía el NAV modificado. Al valor original recibido le resta un tiempo llamado SIFS y el

tiempo necesario para enviar el CTS. Las demás estaciones que escuchan el RTS o el CTS suspenden sus transmisiones hasta que el NAV llegue a 0.

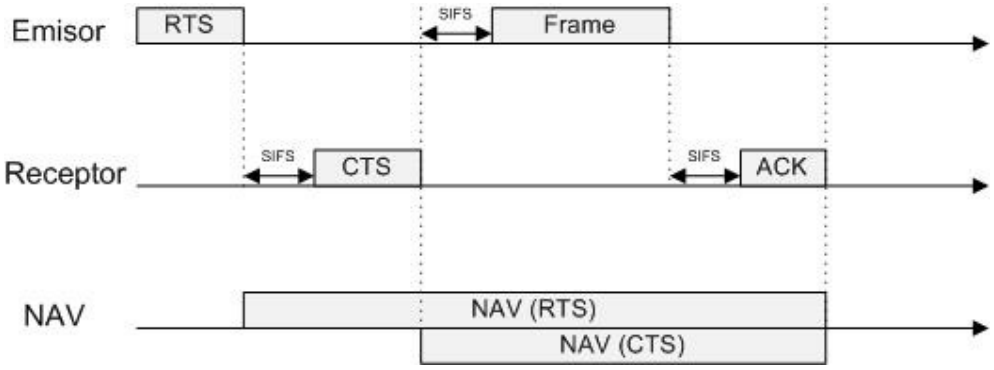



Figura 2.11 - Funcionamiento NAV

El intercambio de los mensajes RTS/CTS es uno de los posibles medios para distribuir la información de reserva del medio. Otra forma de hacerlo es utilizando el campo Duration en los frames de datos.

Los dos métodos se deberían utilizar simultáneamente para verificar si el medio está libre antes de realizar una transmisión.

### 4.4 - Tiempo entre tramas

Con el fin de que las estaciones accedan al medio en forma coordinada existen diferentes intervalos de tiempo entre tramas consecutivas que se llaman Interframe Space (IFS, Espacio entre tramas). En total se definieron 4 diferentes IFS para proveer distintos niveles de prioridad de acceso al medio wireless. Los 4 niveles de acceso son los siguientes, ordenados de menor a mayor por su duración en tiempo:

1. SIFS - Short Interframe Space
2. PIFS - PCF Interframe Space
3. DIFS - DFC Interframe Space
4. EIFS - Extended Interframe Space

A continuación se detalla para qué sirven cada uno de estos parámetros.

1. SIFS

   Éste es el menor tiempo de espaciado entre tramas y es utilizado por las tramas de mayor prioridad, entre las que se encuentran los ACKs, CTS, todos los fragmentos de una trama (excepto el primer fragmento) y toda respuesta a las consultas realizadas por un PCF. El valor de SIFS variará de acuerdo al estándar que se esté utilizando.

2. PIFS

   Este tiempo debe ser utilizado únicamente por estaciones que están operando en un entorno PCF durante el período libre de contención. Al tener una duración menor que DIFS, el tráfico bajo PCF tiene mayor

prioridad que el tráfico con DCF. El tiempo PIFS se calcula con la siguiente fórmula:

$$PIFS = SIFS + slotTime$$

3. DIFS

Todas las estaciones operando con DCF deben esperar un tiempo DIFS, como mínimo, para empezar a transmitir sus tramas de datos y de management. El tiempo DIFS se puede calcular mediante la siguiente fórmula:

$$DIFS = SIFS + 2 * slotTime$$

4. EIFS

Este tiempo debe ser usado bajo DCF. No es un intervalo fijo y se usa solamente cuando la capa física detecta un error en la transmisión de una trama. Este intervalo comienza cuando la capa física le indica a la capa MAC que el medio está libre después de detectar una trama errónea. Es el más largo de los tiempo entre tramas.

En el siguiente gráfico se puede observar una relación entre estos tiempos de espaciado de trama. Una estación que desea transmitir una trama con mayor prioridad deberá esperar un menor tiempo de espaciado entre tramas para ser enviada.

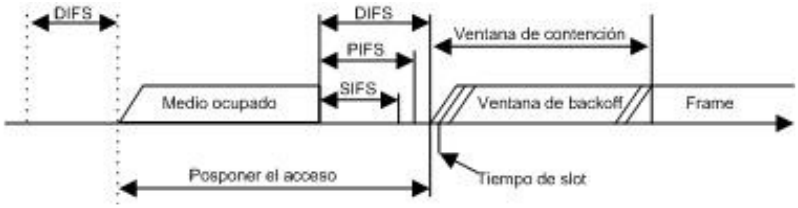



Figura 2.12 - Espaciado entre tramas

El tiempo de slot (slotTime) depende de la capa física.

### 4.5 - CSMA/CA

El protocolo CSMA/CA fue diseñado para reducir la posibilidad de que se produzca una colisión entre dos o más estaciones que desean transmitir simultáneamente y es el método de acceso al medio de DCF. El instante más probable de que ocurra una colisión es cuando el medio se queda libre después de haber estado ocupado por una transmisión. Si varias estaciones están esperando para transmitir lo intentarán hacer en ese instante con el previsible resultado, todas las transmisiones colisionarán. Como CSMA/CA intenta evitar las colisiones, lo que hace es obligar a las estaciones que desean transmitir, después de que cesó la actividad en el medio, a que esperen un tiempo aleatorio antes de hacerlo.

Los pasos que debe seguir una estación cuando quiere transmitir son los siguientes. Lo primero que debe hacer es verificar que el medio esté libre. Si ninguna otra estación está transmitiendo y el medio está libre por un tiempo igual a DIFS (en el caso de que la última trama fuese recibida por la estación

con un valor FCS incorrecto, debe pasar un tiempo EIFS), la estación puede comenzar a transmitir. Pero, si alguno de los mecanismos de comprobación del medio detectan que el mismo está ocupado, la estación debe aplazar su envío hasta que finalice la transmisión actual. Cuando esto sucede, pone en funcionamiento el algoritmo de backoff después de comprobar que el medio está libre por un tiempo igual a DIFS (o EIFS).

El algoritmo de backoff es el encargado de determinar el tiempo que deben diferir su transmisión cada una de las estaciones que están esperando que cese la actividad en el medio. Este tiempo es aleatorio, se llama Backoff Timer (Tiempo de Backoff) e indica la cantidad de tiempo de slots que debe esperar una estación para poder empezar a transmitir. Obviamente, si dos o más estaciones eligen el mismo tiempo de backoff, sus transmisiones colisionarán. La fórmula para calcular el Backoff Timer es la siguiente:

Backoff Timer = rand (0..CW) * slotTime

CW (Contention Window) es un parámetro que toma un valor dentro de un rango entre $CW_{min}$ y $CW_{max}$. Estos son dos valores definidos en la capa física e indican el valor mínimo y máximo de la ventana de contención. En el primer intento de transmisión, el valor de CW es igual a $CW_{min}$. Por cada intento fallido de retransmisión de una trama se incrementa el tamaño de la ventana. Este incremento se da en valores de potencia de 2 menos 1 (7, 15, 31, 63, etc) hasta alcanzar un valor máximo, $CW_{max}$. El tamaño de la ventana vuelve a su valor original cuando la transmisión que disparó el algoritmo es exitosa o porque se agotaron todos los intentos de transmisión, lo que genera un error que se reporta a las capas superiores. Cuanto más grande sea el valor de CW existen menos posibilidades de que varias estaciones elijan el mismo Backoff Timer simultáneamente y colisionen.

Todas las estaciones que necesitan ejecutar el algoritmo de backoff elegirán al azar un tiempo de backoff determinado dentro de su ventana. Cada vez que pasa un tiempo de slot y no se detecta actividad en el medio, la estación debe decrementar su tiempo de backoff en un slot. Cuando llega a cero, la estación puede transmitir. En definitiva, la estación que elija el menor tiempo de backoff ganará y tendrá acceso al medio más rápidamente. Las demás estaciones, al detectar actividad en el medio, deberán suspender el algoritmo. Para esto es necesario que mientras se ejecuta el algoritmo de backoff las estaciones ejecuten los mecanismos de comprobación de medio descriptos anteriormente. Cuando la estación que ganó el medio terminó de transmitir y pasó un tiempo DIFS, las demás estaciones reanudarán el proceso de backoff interrumpido previamente. El valor del tiempo de backoff, a partir del cual las estaciones continúan decrementando, es el mismo que tenían al momento de suspender el algoritmo de backoff.

Como se explicó anteriormente, la recepción de una trama de confirmación (trama ACK) es la única indicación de que una trama unicast fue recibida correctamente por el receptor. Si no se recibe la trama ACK, el emisor considera que hubo un error en la transmisión. Antes realizar la retransmisión de la trama se debe ejecutar el algoritmo de backoff.

En la figura 2.13 se puede observar el funcionamiento del algoritmo de backoff.

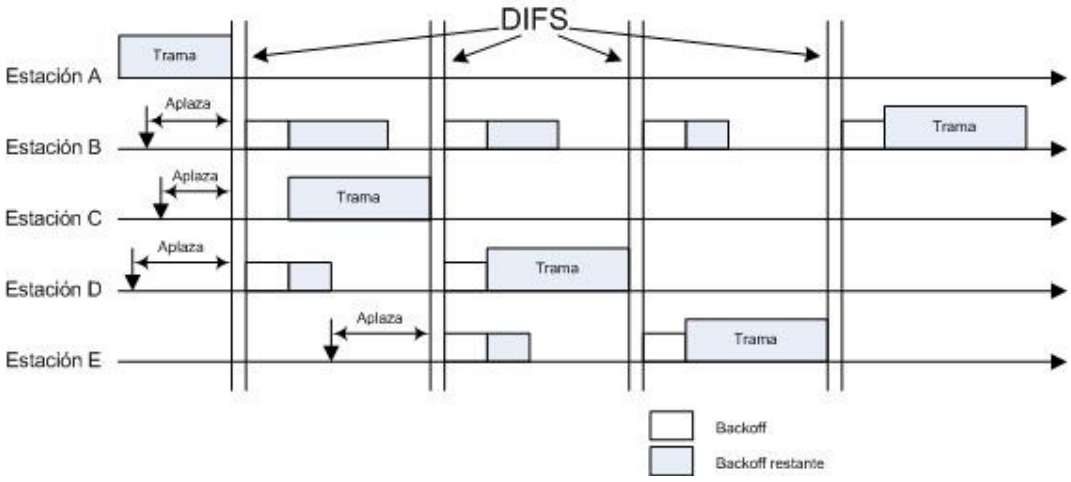



Figura 2.13 - Algoritmo de backoff

Como se puede observar en el gráfico, mientras la estación A está transmitiendo, las estaciones B, C y D intentan empezar sus respectivas transmisiones. Al detectar que el medio no está libre deben posponer dichos intentos. Después que A termina de transmitir su trama, todos deben esperar que pase, como mínimo, un tiempo igual a DIFS para poder transmitir. Las estaciones que tuvieron que aplazar sus transmisiones deben ejecutar al algoritmo de backoff para evitar colisionar entre si. Todas eligen un tiempo de backoff y la que elija el menor valor será la primera en transmitir. La estación C es la que selecciona el menor tiempo, por lo tanto, es la que gana el medio y tiene el derecho a transmitir. Mientras C está transmitiendo, la estación E intenta realizar su transmisión. Como detecta que el medio está ocupado la aplaza. Cuando C finaliza, las estaciones esperan un tiempo DIFS para poder acceder al medio. B y D continúan descontando del tiempo de backoff que habían seleccionado en el primer paso pero E debe elegir su propio tiempo de backoff. La estación D es la próxima en obtener el acceso al medio, luego la estación E y, por último, la estación B.

## 5 - Tramas 802.11

El estándar 802.11 define tres tipos de tramas que se pueden categorizar de acuerdo a sus funciones:

- Administración
  Se necesitan varios tipos de tramas de administración (management) para proveer servicios que son muy simples en redes cableadas. Son tramas supervisoras. Por ejemplo, procesos como identificación de la red, autenticación y asociación utilizan este tipo de tramas.

- Control
  Estas tramas asisten a las estaciones en el envío de las tramas de datos. Ambos tipos de tramas trabajan en conjunción para poder enviar la información en forma confiable entre las estaciones. Por ejemplo, se utilizan para lograr la adquisición del medio (tramas RTS/CTS) o para confirmar la recepción de una trama (tramas ACK)

- Datos
  Son las tramas encargadas de llevar información de una estación a otra. Transportan datos de protocolos de capas superiores en el cuerpo de la trama.

Todas las tramas tienen la misma composición. Están divididas en tres partes:

a) MAC Header (Cabecera MAC): incluye información de control, direcciones MAC, etc.
b) Frame body (Cuerpo de la trama): un campo de longitud variable que contiene información específica de la trama
c) Frame check sequence (FCS, Secuencia de chequeo de trama): contiene un campo de 32 bits conocido como Cyclic Redundancy Code (CRC, Código de redundancia cíclica)

## 6 - Funcionamiento de una red 802.11

A pesar de ser conocida como Wireless Ethernet, las redes inalámbricas incluyen una cantidad de características fundamentales que no son necesarias en Ethernet. Es muy simple detectar una red cableada, dónde se encuentra y cuál es su alcance. Sólo se necesita localizar y seguir los cables de la misma. La señal de la red tiene el mismo alcance que los cables. Pero esto no sucede con las redes inalámbricas. Las señales no son guiadas por ningún cable y su alcance no está perfectamente delimitado. Además, si dos o más redes inalámbricas se encuentran superpuestas, cómo puede saber el usuario que red debe utilizar.

Este problema se soluciona asignándole un nombre a la red. Toda red inalámbrica tiene un nombre que se conoce como SSID (Service Set Identifier) y es anunciado al medio mediante una trama de administración llamada Beacon. En una red infraestructura el access point es el encargado de enviar, periódicamente, estas tramas. Además del SSID, los Beacons también llevan información con las distintas características de la red (velocidades de transferencia, seguridad, etc). Cuando un usuario quiere utilizar una determinada red inalámbrica debe indicar el nombre de la red a la que quiere conectarse.

Por motivos de seguridad se puede ocultar la red haciendo que no se envíe el SSID en las tramas Beacons (llamadas redes ocultas). En este caso, la estación debe buscar la red. Para hacerlo, la estación envía al medio tramas Probe Request indicando el nombre de la red que quiere utilizar. Cuando el

access point que tiene definida esta red escucha la trama, le contesta con una trama Probe Response. Esto se conoce como búsqueda activa.

Una vez que se detectó la red, la estación debe autenticarse con el access point. El estándar original define dos modos de autenticación. Una forma es Autenticación Abierta (Open Authentication) y la otra es Autenticación de Clave Compartida (Shared-Key Authentication). En el primer método se le permite el acceso a todo aquel que quiera utilizar la red, no existe una autenticación real (aunque si se realiza un intercambio de paquetes por cada estación que se autentica). En cambio, en el segundo método existe un proceso de desafío-respuesta. Todas las estaciones comparten una misma clave secreta para lograr la autenticación. Esta clave también se la puede utilizar para encriptar en caso que se requiera el envío de datos encriptados. El algoritmo definido en el estándar original se llama Wired Privacy Equivalency (WEP, Privacidad equivalente a las cabledas). Este algoritmo presentó muchas fallas en su funcionamiento lo que obligó a los ingenieros de la IEEE a desarrollar un nuevo mecanismo de seguridad. Años más tarde, la IEEE daría a conocer el estándar 802.11i que especifica nuevos métodos de seguridad para las redes inalámbricas.

En una red infraestructura, después de autenticarse, las estaciones deben asociarse al access point correspondiente. Este paso incluye el intercambio de dos tramas. El access point puede negarle la asociación a una estación. Cada estación sólo puede estar asociada a un único access point a la vez. A partir de este momento, la estación puede intercambiar información con las demás estaciones de la red. Por último, cuando termina de utilizar la red, la estación se debería desasociar del access point, pero no es obligatorio que lo haga.

En una red IBSS, al no existir un access point, el control de la red lo tienen entre todas las estaciones que la componen. Salvo la asociación, los demás pasos son similares a los de una red infraestructura.

## 7 - PCF

Para soportar servicios con tiempo limitado, el estándar IEEE 802.11 define un mecanismo de acceso al medio opcional llamado Point Coordination Function (PCF, Función de Punto de Coordinación). Este método permite que las estaciones tengan prioridad para acceder al medio y están coordinadas por una función especial llamada Point Coordination (PC, Punto Coordinador) que funciona en los access points y tiene el control de la red. Sólo las redes de tipo infraestructura soportan este método de acceso al medio

Cuando se usa PCF, la red wireless no puede funcionar únicamente en el modo libre de contenciones sino que se deben alternar periodos de contención, controlado por DCF, y libres de contención, controlado por PCF. El periodo de contención debe ser lo suficientemente grande para permitir la transferencia de, al menos, una trama de tamaño máximo y su Ack correspondiente.

Cada vez que el PC quiere comenzar un periodo libre de contenciones envía una trama beacon. En estas tramas indica, además de toda la información que anuncian ordinariamente estas trama, la máxima duración del periodo sin contenciones. Todas las estaciones que la reciban guardan en su vector NAV dicha duración y así se evita que accedan al medio las estaciones que trabajan con DCF. Esto es complementado con el espacio de tiempo entre tramas. Las tramas enviadas durante el periodo libre de contenciones están separadas por SIFS y PIFS, ambos valores son menores que DIFS que es el valor utilizado por las estaciones basadas en DCF, por lo que PCF tiene prioridad sobre DCF. El PC genera estos beacons en intervalos regulares de tiempo llamados Target Beacon Transmisión Time (TBTT), con lo cual cada estación sabe cuando llegará el próximo beacon que iniciará el periodo sin contenciones. El valor del TBTT es anunciado en los beacons.

Durante el modo PCF, el PC va asignando el turno para transmitir a las estaciones que tienen datos para enviar. Una estación no puede transmitir hasta que es autorizado por el access point. Siguiendo una lista de estaciones, llamada “polling list”, el access point sabe que estaciones debe consultar y habilitarlas para transmitir. La “polling list” se va formando a medida que las estaciones se asocian al access point y solicitan que se las consulte.

Utilizando un paquete CF-Poll, el access point le asigna el turno para transmitir a una estación de la lista. Para obtener el medio antes que cualquier otra, la estación consultada espera un tiempo SIFS para empezar a transmitir. Si durante un periodo SIFS el access point no recibe nada de la estación que tiene el turno para transmitir debe enviar el siguiente paquete después que pasó un tiempo PIFS desde el final de la última transmisión.

# Capítulo 3

# El protocolo 802.11e

El estándar 802.11e fue desarrollado para corregir los problemas que presentaba el estándar 802.11 al momento de ofrecer Calidad de Servicio a los diferentes tipos de tráficos de red. Como se explicó en el capítulo anterior, las redes wireless tienen un funcionamiento similar al de las redes Ethernet, pero éstas tienen un rendimiento muy superior, motivo por el cual no es sumamente necesario considerar aspectos tales como la Calidad de Servicio.

Tiempo atrás, los usuarios no encontraban muchos servicios disponibles en las redes de datos. Con el transcurso de los años, la oferta de servicios por parte de las redes se ha ido incrementando. Hoy en día, las redes de datos son usadas para acceder a información estática y/o dinámica, transmitir voz y video, realizar compras, etc. Obviamente, estas acciones generan diferente tipo de tráfico que deben recibir distinto tratamiento por parte de la red.

## 1 - Calidad de servicio

Calidad de Servicio (QoS, Quality of Service) es un término usado para definir la capacidad de una red para proveer diferentes niveles de servicio a los distintos tipos de tráfico. Permite que los administradores de una red puedan asignarle a un determinado tráfico prioridad sobre otro y, de esta forma, garantizar que un mínimo nivel de servicio le será provisto. Debido al desarrollo de estos nuevos tipos de aplicaciones (streaming, Voz sobre IP, videoconferencia, etc.), la necesidad de implementar técnicas de calidad de servicio se ha vuelto más evidente.

En los últimos tiempos, la capacidad de las redes se ha incrementado considerablemente, a tal punto que en la actualidad la velocidad de muchas redes se llega a medir en gigabit, lo que permite, que si una red presenta problemas de congestión sea muy posible aumentar su ancho de banda. Ésta es una solución simple que, además de ser costosa, puede introducirnos en un círculo vicioso. Podría suceder que después de incrementar el ancho de banda de la red los protocolos que causaron la congestión original, o nuevos tipos de tráficos, consuman el ancho de banda adquirido lo que nos ubica nuevamente en la misma situación experimentada antes del incremento. Una mejor solución sería analizar los distintos flujos de tráficos de la red con el fin de determinar la importancia de cada protocolo y/o aplicación, y así, poder implementar una estrategia para priorizar la utilización del ancho de banda de acuerdo a las necesidades de cada tráfico y/o aplicación.

Dependiendo del tipo de aplicación son los requerimientos que se precisan. Por ejemplo, FTP no es un protocolo críticamente sensitivo a la congestión de la red. Simplemente, la operación tardará un tiempo mayor en realizarse pero no impide que se ejecute correctamente (salvo que la

congestión sea tan grande que la conexión de timeout y se aborte). En cambio, las aplicaciones de voz o video son particularmente sensitivas a retardos de la red. Si a los paquetes que componen una comunicación de voz les toma demasiado tiempo en llegar al destino, el sonido o el video resultante estarán distorsionados. Aplicando técnicas de Calidad de Servicio se puede proveer un servicio más acorde al tipo de tráfico.

A continuación se indican algunas de las situaciones en las cuales sería conveniente dar Calidad de Servicio:

- para dar prioridad a ciertas aplicaciones de nivel crítico en la red
- para maximizar el uso de la infraestructura de la red
- para proveer una mejor performance a aplicaciones sensitivas al retardo como son las de voz y video
- para responder a cambios en los flujos del tráfico de red

Al aplicar técnicas de Calidad de Servicio, el administrador de la red puede tener control sobre los diferentes parámetros que definen las características de un tráfico en particular, entre los que se encuentran el delay (latencia), jitter (variación en el retardo), packet loss (pérdida de paquetes) y bandwidth (ancho de banda). A continuación se definen cada uno de ellos:

- Delay: es la cantidad de tiempo que tarda un paquete en alcanzar el destino después de ser transmitido desde el emisor. Este periodo de tiempo es conocido como “retardo de fin a fin” (end-to-end delay).
- Jitter: es el cambio de la latencia durante un periodo de tiempo. Indica la variación de tiempo en el arribo entre paquetes debido a las condiciones variables de la red. Por ejemplo, si un paquete tiene 100 milisegundos de latencia y el siguiente paquete tiene una latencia de 130 milisegundos, entonces el jitter es de 30 milisegundos.
- Packet Loss: indica la cantidad máxima de paquetes que puede perder una red. Ésta no puede garantizar que todos los paquetes alcanzarán su destino. En determinados picos de carga, los paquetes serán eliminados por los routers.
- Bandwidth: los distintos tipos de aplicaciones compiten por el limitado ancho de banda. La falta de ancho de banda puede causar retardo, pérdida de paquetes y pobre performance para las aplicaciones.

Si una red estuviese vacía el tráfico de una aplicación debería conseguir cumplir con todos los parámetros anteriores, obtendría el bandwidth necesario, no perdería paquetes y tampoco sufriría delay ni jitter. Pero la realidad es diferente. Existen varias aplicaciones usando la red al mismo tiempo y, por lo tanto, compitiendo por los recursos disponibles.

De los anteriores términos el más difícil de comprender es el Jitter, es por esto que a continuación se muestra un gráfico para ayudar a entender su significado. Los paquetes A y B llegan al destino cada 50 milisegundos pero el paquete C tarda 90 milisegundos, 40 milisegundos más de retardo que los dos paquetes anteriores lo que provoca un jitter de 40 milisegundos.

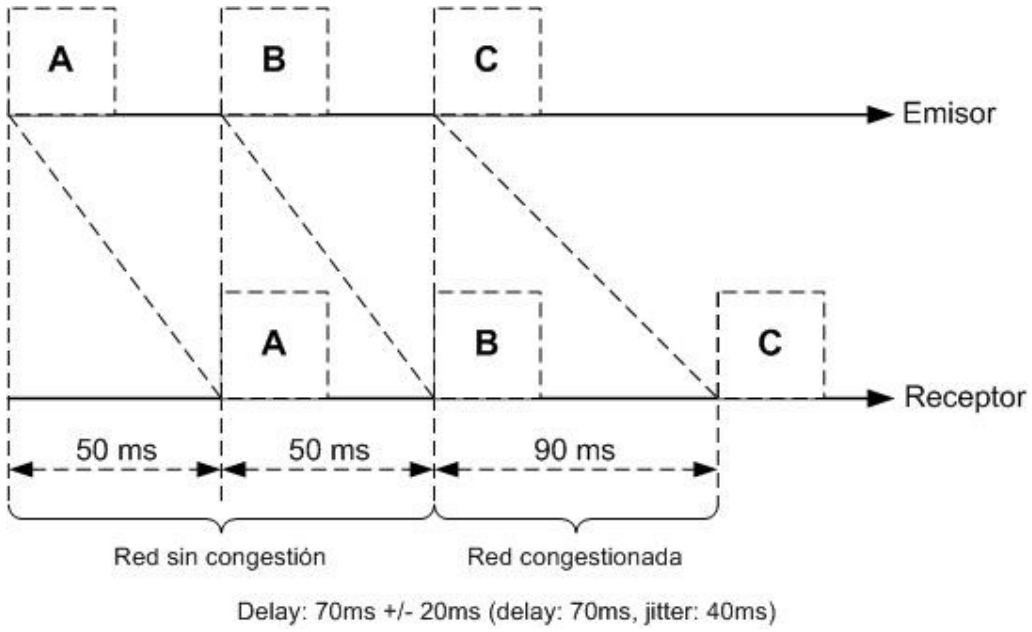



Figura 3.1 - Jitter

## 2 - Modelos de servicios

Las técnicas de Calidad de Servicio pueden ser divididas en tres niveles o modelos de servicios. Estos modelos describen un conjunto de capacidades que provee la red a determinados tráficos desde su origen hasta su destino. Estos niveles son: servicio de mejor esfuerzo, servicios integrados y servicios diferenciados.

### 2.1 - Servicio de mejor esfuerzo

Éste es el nivel que proveen las redes IP y, por consiguiente, es el modelo que se utiliza en Internet. La red hará todo lo posible por enviar cada paquete hasta su destino, pero no da ninguna garantía de que eso suceda. Una aplicación puede enviar todos los datos que desee en cualquier momento sin solicitar permiso o notificar a la red. Determinadas aplicaciones, como FTP o HTTP, pueden utilizar este modelo sin mayores inconvenientes, pero no es un modelo óptimo para otro tipo de aplicaciones.

### 2.2 - Servicios integrados

En este modelo se provee, a cada flujo, un nivel garantizado de servicio mediante la negociación de distintos parámetros de red desde el origen al destino. Para esto, la aplicación debe indicar las características del flujo que inyectará en la red y especificar los requerimientos de recursos para el flujo. Los routers que se encuentran a lo largo del camino, entre el origen y el destino, reservan los recursos de red solicitados antes de que la aplicación empiece a transmitir. Ésta no enviará tráfico hasta que reciba una señal de la red indicándole que puede manejar la carga y proveer la calidad de servicio requerida.

Cuando recibe una solicitud de recursos, la red ejecuta un proceso de control de admisión. Mediante este mecanismo, la red comprueba que está en condiciones de satisfacer los requerimientos solicitados. Si es así, se realiza la reserva de recursos en los routers, que se mantiene hasta que la aplicación termine la transmisión. En caso contrario, la reserva no se puede hacer y se rechaza la conexión.

RSVP (Resource Reservation Protocol) es el protocolo que se encarga de realizar la reserva de los recursos solicitados por la aplicación en forma dinámica. Éste es un protocolo que se desarrolla entre los usuarios y la red, y entre los routers de la red que soportan este protocolo. Tanto la solicitud de reserva de recursos en los routers, como su mantenimiento y cancelación, se hace mediante el intercambio de mensajes de señalización RSVP. En grande entornos esto representaría un considerable tráfico adicional.

Este método tiene la desventaja de que para cada flujo de información que lo requiera es necesario hacer una reserva de recursos en los routers, lo que puede producir que, ante una gran demanda de servicios, un router no pueda satisfacer todos los pedidos. No es una solución escalable, por lo cual no es adecuada para grandes entornos como Internet.

### 2.3 - Servicios diferenciados

Este método fue concebido para superar los problemas de escalabilidad de Servicios Integrados. Los tráficos ya no se tratan individualmente, sino que se agrupan en diferentes clases que reciben distinto tratamiento por parte de los routers. Los routers de borde son los encargados de marcar los paquetes que entran a la red. El procesamiento que reciban los paquetes dentro de la red depende de la clase en la que fueron ubicados.

El marcado consiste en modificar los primeros 6 bits del campo DS (DiffServ) llamado DSCP (DiffServ Code Point). DS suplanta las definiciones de los campos Type of Service de la cabecera IP y Traffic Class de IPv6. Cada uno de los posibles valores de DSCP puede significar una forma diferente de tratar los paquetes por parte de los routers. A cada una de las formas de tratar los paquetes se lo conoce como Per-Hop Behavior (PHB).

Ésta es una solución escalable. El router sólo debe mirar el valor del campo DSCP para decidir como procesar cada paquete. No es necesario mantener un estado por flujo en cada router ni intercambiar tráfico de señalización. La desventaja de este método es que si se agrega una nueva conexión, todos las demás conexiones serán afectadas. Por ejemplo, si hay 10 conexiones atravesando un router y se genera una nueva conexión, el router la aceptará, incluso si sus recursos están saturados, los que, a partir de ahora serán compartidos por las 11 conexiones, introduciendo una posible degradación en la calidad recibida en todas las conexiones. En Servicios Integrados esto no sucede. Una nueva conexión no afecta el rendimiento de las demás conexiones ya establecidas y, si un router no tiene suficientes recursos para satisfacer los requerimientos de aplicación, la conexión se rechaza.

# 3 - Herramientas de Calidad de Servicio

Las herramientas de Calidad de Servicio que tienen disponibles los administradores de redes se encuentran dentro de las siguientes:

- Marcado y clasificación
- Policing y Shaping
- Control de congestión
- Evitar la congestión

## 3.1 - Marcado y clasificación

Un paquete que entra en un router primero debe pasar por los procesos de marcado y clasificación. Esto permite diferenciar entre los distintos tráficos y así poder tratarlos de acuerdo a la clase de tráfico a la que pertenecen. Aunque se suelen usar en forma intercambiable, su función es diferente.

Marcado: este paso se encarga de escribir un campo en un paquete con el propósito de distinguir un tipo de paquete de otro. Modifica el campo DiffServ (DS) que ocupa los primeros 6 bits del campo Type of Service en IPv4 y Traffic Class en IPv6. Este paso se realiza en los routers de borde de la red.

Clasificación: el grupo o clase al que pertenecen cada uno de los paquetes depende del valor con el que fueron marcados. Cada uno de esos agrupamientos recibirá un tratamiento diferente, de acuerdo a las políticas que se hayan especificado para cada de una de las clases.

## 3.2 - Policing and shaping

Estas son dos de las primeras técnicas utilizadas para brindar Calidad de Servicio. El objetivo de estas herramientas es similar, controlar la velocidad a la cual es admitido el tráfico en la red. Aunque la forma de identificar las violaciones de tráfico es similar, la forma en que las resuelven es donde difieren.

Policing: esta técnica, al detectar tráfico excesivo, elimina los paquetes con el fin de mantener los flujos de datos dentro de los límites definidos o los remarca para que tengan menor prioridad. No introduce ningún retardo a los flujos que cumplen con las reglas de policing pero puede provocar más retransmisiones por parte de TCP.

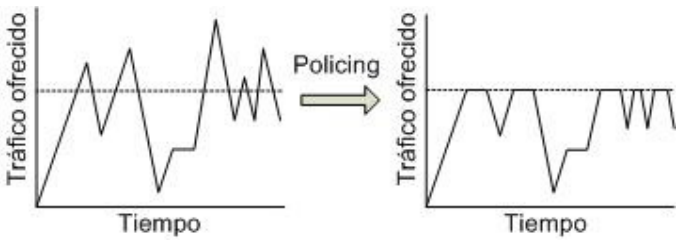



Figura 3.2 - Policing

Shaping: esta técnica, a diferencia del policing, no elimina el tráfico excesivo sino que lo aplaza, intentando que no se sobrepase el límite establecido. Lo que hace es poner en un buffer el tráfico excesivo e intenta transmitirlo más tarde cuando su transmisión no exceda el límite acordado. Implica la existencia de colas y suficiente espacio de memoria para mantener los paquetes pendientes.

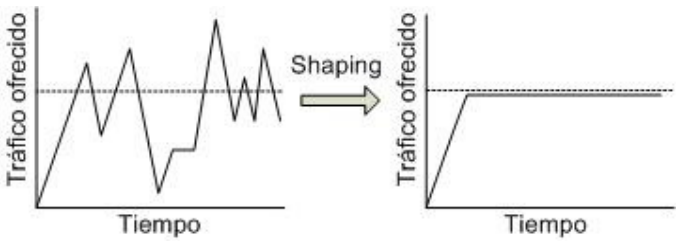



Figura 3.3 - Shaping

### 3.3 - Control de congestión

Existen distintas estrategias de encolado para solucionar el problema que se presenta cuando las aplicaciones, en su conjunto, requieren más ancho de banda del total disponible. Estas estrategias no evitan la congestión pero si permiten que cierto tráfico de red tenga prioridad sobre otro.

Los routers mantienen los paquetes en las colas hasta que tengan suficientes recursos para reenviarlos por el puerto correspondiente, lo que harían inmediatamente si no hubiese congestión. Las colas son usadas para manejar las ráfagas de tráfico que llegan al router más rápido de lo que la interface de salida puede reenviarlas. En un router, las colas son espacio de memoria física lo que implica que su tamaño no es infinito. Sólo pueden contener una cantidad limitada de información (por lo general, el tamaño de las colas es un valor configurable). Los paquetes son ubicados en la cola en el mismo orden en el que arriban. Si la cantidad de paquetes que se reciben sobrepasan la capacidad de la cola, ésta se desborda y los paquetes no se pueden encolar. Se los descarta. Esto se conoce como tail-drop. Obviamente, los protocolos de las capas superiores lo detectarán y retransmitirán los paquetes eliminados. En un entorno TCP/IP corriendo varias aplicaciones esto puede introducir un problema conocido como “TCP global synchronization” que se muestra en la figura 3.4. Si una cola se llena, los paquetes de las distintas conexiones TCP, que arriben a esa cola, serán descartados. TCP interpretará esto como un error de transmisión y disminuirá el tamaño de su “ventana deslizante” para reducir su velocidad de transmisión. A medida que las próximas trasmisiones sean exitosas irá agrandando esa ventana. Si varias conversaciones TCP acontecen simultáneamente, todas sufrirán el mismo problema e intentarán solucionarlo de igual manera, decrementando su correspondiente ventana y luego incrementándola paulatinamente mientras transmiten exitosamente. Esto podría causar que la interface se vuelva a congestionar y comience a eliminar paquetes de nuevo. Esto será interpretado como un error de transmisión por parte de TCP. Nuevamente, las conversaciones TCP reducirán su ventana de transmisión para disminuir su tasa de transferencia causando una fluctuación en el uso de la red.

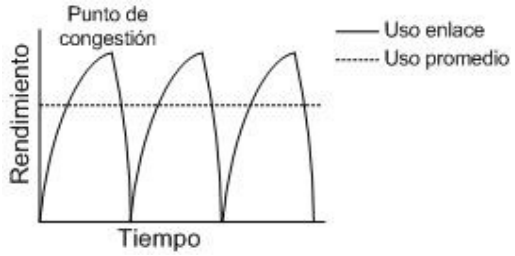



Figura 3.4 - TCP global synchronization

Además del encolado es importante entender el término *scheduling (planificación)*. Éste es el proceso encargado de determinar cuál es el próximo paquete a transmitirse. Si se tienen varias colas con distintas prioridades el scheduler se encarga de seleccionar de qué cola se enviará el siguiente paquete. Hay varios algoritmos que permiten realizar esta decisión:

- Strict Priority (Prioridad estricta): siempre sirve primero las colas con mayor prioridad. Sólo si éstas están vacías atiende a una de menor prioridad, lo que puede producir inanición.
- Round Robin: las colas son servidas en secuencia. No produce inanición pero puede introducir delay en los tráficos sensitivos al tiempo.
- Weighted Fair: las colas son servidas según su importancia. Los paquetes en las colas son "pesados" (weighted), por ejemplo, por el campo IP Precedence, y las colas con mayor importancia son servidas más frecuentemente. No puede garantizar el ancho de banda que algunos tráficos necesitan pero soluciona los problemas de los dos algoritmos anteriores.

Entre las técnicas de encolado más conocidas se encuentran las siguientes:

- First In First Out (FIFO)
- Priority Queuing (PQ)
- Custom Queuing (CQ)
- Fair Queuing (FQ) / Weighted Fair Queuing (WFQ)

### 3.3.1 - First In First Out (FIFO)

Éste es el tipo de encolado más simple. Simplemente, el primer paquete que entre en la cola será el primero en ser reenviado. Ninguna clasificación se realiza con los paquetes. El principal propósito es manejar los paquetes entrantes en una interface, ubicarlos en la cola en el mismo orden en el que fueron recibidos, y alimentar la interface saliente a la velocidad constante que la interface puede manejar. No existen las clases de tráficos, todos los paquetes pertenecen a la misma clase. Si se llenan, los buffers empiezan a descartar paquetes (tail-drop). Y esto lo hace con todos los paquetes sin importar la prioridad de los mismos. Además, un flujo muy agresivo puede provocar inanición al transmitir una cantidad de paquetes que mantenga siempre llena la cola y provoque que los paquetes de las demás flujos sean descartados. Al arribar los paquetes de un flujo en particular, la cola puede

estar vacía, con lo cual se reenviarán rápidamente, prácticamente sin delay, o puede estar casi llena lo que provocará que tengan que esperar un tiempo mayor para ser retransmitidos. Esto puede introducir variación en el jitter de la conexión.

### 3.3.2 - Priority Queuing

Este método, muy fácil de implementar, es una manera estricta de manejar la congestión. Permite definir varias colas y asignarle a cada una un nivel distinto de prioridad. Estas colas son del tipo FIFO y, si se llenan, utilizan el método tail-drop para descartar paquetes. Son procesadas estrictamente por orden de prioridad. Si una cola de mayor prioridad tiene paquetes encolados será procesada hasta que esté vacía, independientemente del estado de las demás colas. Cuando el router termina de procesar una cola de mayor prioridad pasa a la siguiente cola basándose en el nivel de prioridad. Cada vez que se envía un paquete de una cola, el router chequea las colas de mayor prioridad para asegurarse que siguen vacías. Si no es así, el router enviará los paquetes de la cola de mayor prioridad. Este algoritmo es muy bueno en el manejo del tráfico de tiempo real y permite que los tráficos más importantes sean reenviados más rápido. El problema que presenta es que puede producir inanición. Si las colas de mayor prioridad tienen mucho tráfico, los paquetes encolados en las de menor prioridad nunca serán enviados.

### 3.3.3 - Custom Queuing

Para solucionar el problema de inanición presente en el método anterior se introdujo Custom Queuing o Class-Based Queuing. Este método permite definir varias colas, tipo drop-tail, que son atendidas en forma round-robin, asegurándose que todas las colas tengan su oportunidad de transmitir. Cada cola sólo puede enviar una cantidad máxima de bytes, no paquetes, por turno. La prioridad de una cola está dada por la cantidad de bytes que es capaz de enviar por turno. El último paquete siempre es enviado en su totalidad aun si la cantidad total de bytes enviados en el turno supera el máximo permitido para la cola. Aunque no es posible la inanición, sí puede suceder que se le asigne un ancho de banda tan grande a una cola (o más de una) que implique que las colas de menor prioridad no puedan obtener el ancho de banda necesario. Cuando esto sucede, las aplicaciones con datos en estas colas (las de menor prioridad) pueden dar timeout. Esto provoca que las aplicaciones no puedan funcionar en forma apropiada y tiene los mismos efectos que la inanición

### 3.3.4 - Fair Queuing (FQ) / Weighted Fair Queuing (WFQ)

El método Fair Queuing es otro método para crear distintas clases de tráficos. También se lo conoce como encolado por flujo o basado en flujo. Los paquetes entrantes son clasificados en N colas y, a cada cola, se le asigna 1/N del ancho de banda de la interface de salida. El scheduler visita las colas en forma round-robin, salteando las que están vacías. Es simple de implementar, no requiere un mecanismo especial de asignación de ancho de banda. Si se agrega una nueva cola, para crear un nuevo tráfico, el scheduler, automáticamente, ajusta el ancha de banda de las colas de salida a 1/(N + 1).

El problema que presenta esta solución es que si algún tráfico necesita mayor ancho de banda no tiene forma de poder cumplir con ese requerimiento. Otro problema que presenta es que un paquete entero es transmitido por cada ciclo y el tamaño de los paquetes impactará la distribución del ancho de banda. Una cola con paquetes de mayor tamaño conseguirá más ancho de banda que otra con paquetes de menor tamaño aunque las dos envíen la misma cantidad de paquetes.

Con la idea de solucionar este problema surge Weighted Fair Queuing. Ésta es una generalización del método Fair Queuing. En vez de asignar el ancho de banda de salida en forma proporcional a cada clase, lo hace de acuerdo a sus requerimientos de ancho de banda. Para poder diferenciar los distintos requerimientos de cada flujo (o cola) se les asigna un peso (weight). Este peso controla el porcentaje de ancho de banda que recibirá cada flujo. Usando el campo ToS (Type of Service), de la cabecera IP, podemos indicar el peso.

### 3.4 - Evitar la congestión

Los métodos anteriores no intentan evitar la congestión sino que mediante la definición de distintas reglas, priorizan un tráfico sobre otro, es decir, administran la congestión una vez que se produjo. Existen otras técnicas que intenta evitar que se produzca una congestión en la red eliminando paquetes de flujos TCP cuando detecta que se está alcanzando un estado de congestión. El propósito de estos métodos es indicarle a los protocolos end-to-end, como TCP, que en algún punto, la red se está empezando a congestionar. Se busca que TCP disminuya la velocidad de transmisión en un conjunto random de flujos antes de que la congestión se torne severa y se eliminen paquetes de todos los flujos, como lo haría tail-drop. Entre esas técnicas se encuentran:

- Random Early Detection (RED)
- Weighted Random Early Detection (WRED)

#### 3.4.1 - Random Early Detection (RED)

Este mecanismo descarta paquetes, en forma aleatoria, antes de que se produzca la congestión. No espera a que se llenen los buffers para empezar a descartar, como hace tail-drop, con lo cual evita problemas presentes en este método como “TCP global synchronization”. Debe medir constantemente la ocupación de los buffers para empezar a descartar una vez que se superó un umbral predeterminado. El umbral, en el cual RED empieza a descartar paquetes, es un valor configurable. Si el buffer está casi vacío, todos los paquetes entrantes son aceptados y reenviados normalmente. Pero si el tamaño de la cola crece, la probabilidad de eliminar los paquetes entrantes también crece.

A continuación se muestra un gráfico donde se detalla el funcionamiento de este método.

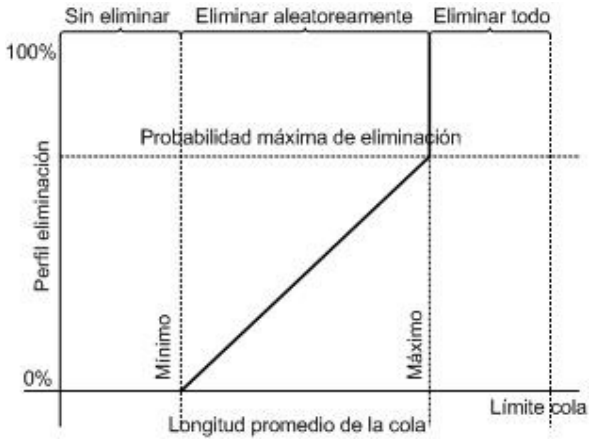



Figura 3.5 - RED

Cuando la longitud de una cola excede el umbral mínimo el router empieza a descartar paquetes en forma aleatoria. Si se alcanza el máximo tamaño de la cola, se descartan todos los paquetes.

### 3.4.2 - Weighted Random Early Detection (WRED)

Ésta es una mejora al método RED. La selección de los paquetes a ser eliminados no es tan aleatoria como en RED, sino que se introduce un grado de influencia en esa selección. Se utiliza un peso, el campo IP Precedence, para determinar que paquetes descartar. En general, WRED descarta primero los paquetes con un valor menor en ese campo. De esta forma, los tráficos con prioridad superior son enviados con una mayor probabilidad que los tráficos menos prioritarios.

A simple vista, las técnicas de evitación de congestión parecen ser la solución a todos los problemas de congestión de una red. Sin embargo, presentan algunas desventajas. En primer lugar, sólo trabajan con conexiones TCP. Por ejemplo, IPX no trabaja con el concepto de “ventana deslizante”, por lo tanto, si se descartan paquetes pertenecientes a este protocolo la retransmisión será a la misma velocidad que antes del descarte. Tanto RED como WRED son ineficientes en una red con protocolos distintos a TCP. Otra desventaja es que los paquetes son eliminados directamente, no los encola.

## 4 - El protocolo 802.11e

En la próxima generación de redes inalámbricas es muy probable que las redes IEEE 802.11 tengan un papel fundamental. Sus principales ventajas son su bajo costo económico y fácil instalación. Actualmente, además de estar implementadas en las universidades, aeropuertos, hospitales y muchos otros sitios, los usuarios han comenzado a mostrar mucho interés en instalar esta clase de redes en sus hogares debido a sus ventajas (no es necesario romper paredes ni desplegar cables por las habitaciones para conectar la red).

A la par de este tipo de redes, el uso de las aplicaciones multimedias se está incrementando en forma exponencial. Hoy en día, los usuarios utilizan las

redes de datos para transportar voz, audio y video. Y las redes inalámbricas no son ajenas a esta tendencia. Pero, a diferencia de las aplicaciones de datos de mejor esfuerzo, las aplicaciones multimedia requieren soporte de Calidad de Servicio como un ancho de banda garantizado o un retardo o jitter acotado.

El protocolo MAC del estándar IEEE 802.11 no provee ninguna forma de diferenciar los distintos tipos de tráficos. Todos son tratados en forma equitativa.

El estándar 802.11e fue desarrollado para enmendar los problemas presentados por el estándar original 802.11 a la hora de brindar Calidad de Servicio. Este último define dos métodos de acceder al medio, DCF (Distributed Coordination Function) y PCF (Point Coordination Function) pero ninguno de los dos mecanismos provee suficiente funcionalidad para satisfacer la Calidad de Servicio demandada por las aplicaciones de multimedia.

### 4.1 - Limitación de Calidad de Servicio en DCF

DCF es el método principal de acceso al medio en las redes inalámbricas IEEE 802.11 y sólo provee un servicio de mejor esfuerzo. Es decir, todas las estaciones compiten por obtener el medio, y el consiguiente permiso para poder transmitir, con igual prioridad. Nadie tiene el control total de la red. Tampoco lo tiene el access point en una red de tipo infraestructura. No hay forma de priorizar un tráfico sobre otro. Su principal ventaja, que es la equidad de acceso al medio entre todas las estaciones, es su mayor debilidad al momento de brindar Calidad de Servicio.

### 4.2 - Limitación de Calidad de Servicio en PCF

Aunque fue desarrollado para soportar aplicaciones con tiempo acotado, este método presenta tres problemas que hacen que la Calidad de Servicio provista sea deficiente:

- define un solo método de asignación del medio de tipo round-robin. Esto no permite manejar los diferentes requerimientos de Calidad de Servicio de las distintas aplicaciones.
- el mecanismo establece la transmisión de una trama beacon cada un tiempo TBTT (Target Beacon Transmission Time) pero, el access point, que es el encargado de enviar dicho trama, debe disputar el acceso al medio con las demás estaciones. Si el medio está ocupado en el momento en el que se vence el TBTT, la emisión del beacon podría ser retrasada. Además, las estaciones están autorizadas a transmitir aun cuando la transmisión finalice después que se venció el tiempo del siguiente TBTT. Si se retrasa la transmisión del beacon, los datos esperando para ser transmitidos en el periodo libre de contención también sufrirán un retardo.
- es muy difícil para el access point controlar el tiempo de transmisión de una estación consultada. Una estación puede transmitir tramas de hasta una longitud máxima de 2304 bytes lo que puede introducir retardos variables en una transmisión. Además, la velocidad de

transmisión puede cambiar de acuerdo a las condiciones variables del canal. Así, el access point no es capaz de predecir de manera segura el tiempo de transmisión de una estación. Esto evita que el access point pueda proveer un jitter y/o retardo garantizado a las estaciones.

El estándar 802.11e define un conjunto de mejoras a la capa MAC del estándar 802.11 para proveer Calidad de Servicio. En 802.11e se implementa una nueva función de la capa MAC que se llama Función de Coordinación Híbrida (HCF, Hybrid Coordination Function), la cual combina las funciones de DCF y PCF con algunas mejoras, mecanismos específicos de Calidad de Servicio y nuevos tipos de tramas. HCF tiene dos modos de operación llamados Acceso al Canal Distribuido Mejorado (EDCA, Enhanced Distributed Channel Access) y Acceso al Canal Controlado HCF (HCCA, HCF Controlled Channel Access) que son compatibles con los métodos heredados del 802.11. El siguiente gráfico muestra como modifica el estándar 802.11e al estándar original.

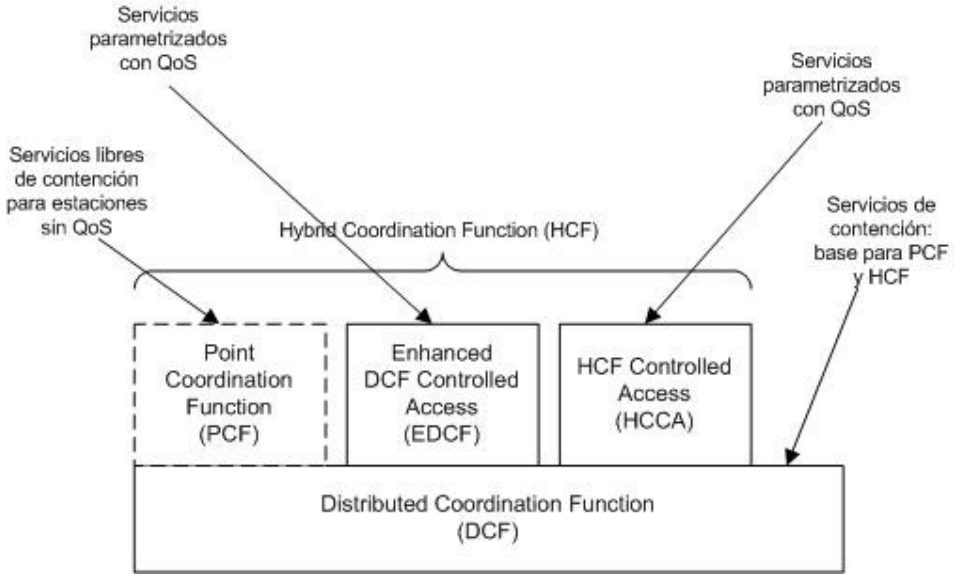



Figura 3.6 - Modelo 802.11e

EDCA se puede utilizar únicamente durante el periodo de contienda, mientras que HCCA puede, teóricamente, operar durante ambos periodos (con y sin contienda) pero el estándar 802.11e recomienda su uso solamente en el periodo sin contienda.

Este estándar se refiere a una estación que soporta Calidad de Servicio como Estación con Calidad de Servicio (QSTA) y al access point como Access Point con Calidad de Servicio (QAP). En un QBSS (QoS Basic Service Set) existe un controlador centralizado llamado Hybrid Coordinator (HC, Coordinador Híbrido) que implementa la secuencia de intercambio de tramas y las reglas de manejo de los MSDU. El HC opera durante los dos periodos. Es el encargado de asignar los TXOPs y de iniciar los intervalos de contención controlada (CCI, Controlled Contention Interval). Normalmente, el HC reside en

un access point convirtiéndolo en un QAP.

Un TXOP (Transmission Opportunity) es una oportunidad de transmisión y está definido por un tiempo de inicio y una duración máxima. Se define como el intervalo de tiempo en el cual una estación tiene derecho a transmitir una serie de tramas. Una estación puede obtener un TXOP en cualquiera de los dos métodos de acceso. Si lo hace durante el periodo de contención de acuerdo a las reglas del mecanismo EDCF es definido como EDCF-TXOP. Si el TXOP es obtenido durante el periodo de acceso al canal controlado se lo define como HCCA-TXOP.

Una QSTA implementa un superconjunto de las funcionalidad de una estación sin QoS, lo que implica que una estación con soporte para Calidad de Servicio debería poder asociarse con un access point sin soporte para Calidad de Servicio.

### 4.3 - EDCA (Enhanced Distributed Channel Access)

EDCA es una extensión para mejorar el mecanismo de acceso DCF y provee acceso con prioridad al medio inalámbrico. Forma parte del modo HCF, no es una función de coordinación separada. Define un mecanismo de categorías de acceso (AC, Access Category) que provee soporte para las prioridades en las estaciones. Cada estación puede tener hasta cuatro AC para ocho prioridades de usuario (UP, User Priorities). Una o más prioridades de usuario son asignadas a una categoría de acceso. Las estaciones acceden al medio basado en la categoría de acceso de la trama a ser transmitida. Cada AC es una variante mejorada de DCF que disputa las oportunidades de transmisión (TXOPs) utilizando los parámetros de EDCA.

Para que una AC, con una mayor prioridad, obtenga acceso al medio con anterioridad a las demás ACs, con menor prioridad, se le asigna un tiempo de backoff menor, lo que implica que la AC con mayor prioridad tenga que esperar menos tiempo. Esto se hace seteando los valores de los parámetros, específicos para cada AC, $CW_{min}[AC]$ y $CW_{max}[AC]$ que no son fijos como en DCF sino que son variables. Estos valores, que definen el rango del cual se obtiene el CW[AC], son menores cuando mas prioritaria es la AC. Además, para lograr una mayor diferenciación entre las distintas ACs se introducen diferentes espacios entre tramas (IFS, Inter-frame space). En lugar de DIFS se utiliza un nuevo IFS llamado AIFS (Arbitration Inter-Frame Space), específico para AC, que indica el tiempo que debe esperar esa AC antes de intentar transmitir o empezar el algoritmo de backoff. La siguiente fórmula define el valor de AIFS:

$$AIFS = SIFS + AIFS[AC] * slotTime$$

Como se puede observar, el valor mínimo que puede tomar AIFS, y que se corresponde con la AC de mayor prioridad, es igual DIFS. Es decir, AIFS es DIFS más algún tiempo de spot que puede ser igual a 0. Lógicamente, cuanto menor sea el valor de AIFS más posibilidades tiene la AC correspondiente de acceder al medio.

La tabla que se muestra a continuación indica los parámetros definidos por el estándar de las CW y AIFS para las diferentes AC.

| AC | $CW_{MIN}$ | $CW_{MAX}$ | AIFS |
|---|---|---|---|
| AC_BK | $CW_{min}$ | $CW_{max}$ | 7 |
| AC_BE | $CW_{min}$ | $CW_{max}$ | 3 |
| AC_VI | $[(CW_{min} +1)/2] – 1$ | $CW_{min}$ | 2 |
| AC_VO | $[(CW_{min} +1)/4] – 1$ | $[(CW_{min} + 1)/2] –1$ | 2 |

Tabla 3.1 - Parámetros de los diferentes AC

En la siguiente tabla se muestran las distintas prioridades definidas en el estándar y su mapeo a las distintas Categorías de Acceso (las prioridades son las mismas que las establecidas en el estándar IEEE 802.1p).

| | Prioridad Usuario | Categorías de Acceso (AC) | Designación (Informativa) |
|---|---|---|---|
| Lowest | 1 | AC_BK | Background |
| | 2 | AC_BK | Background |
| | 0 | AC_BE | Best Effort |
| | 3 | AC_BE | Best Effort |
| | 4 | AC_VI | Video |
| | 5 | AC_VI | Video |
| | 6 | AC_VO | Voice |
| Highest | 7 | AC_VO | Voice |

Tabla 3.2 - Mapeo prioridades a las Categoría de Acceso

Al igual que en DCF, si el medio se detecta libre, la transmisión puede comenzar inmediatamente. Si no es así, la estación suspende su transmisión hasta que la estación que está ocupando el medio lo libere. Cuando esto sucede, la estación espera un tiempo AIFS(AC) para comenzar su procedimiento de backoff. El intervalo de backoff es un número random derivado del rango [0, CW(AC)]. Cada AC dentro de una estación se comporta como una estación virtual. Compite por el acceso al medio inalámbrico e independientemente comienza su tiempo de backoff después de sensar el medio como libre por al menos un tiempo AIFS. Las colisiones entre las distintas ACs dentro de una misma estación son resueltas dentro de la estación para lograr que las ACs con mayor prioridad reciban más rápido el TXOP. Si el tiempo de backoff de dos o más categorías de tráfico alcanza cero en el mismo instante, un planificador interno a la estación evita la colisión virtual dándole el acceso al medio al que tenga mayor prioridad. Una colisión entre distintas ACs se trata como si fuese una colisión externa. También, cuando más prioridad tiene la AC mayor es el intervalo de TXOP, lo que implica que por cada turno puede transmitir más cantidad de paquetes. En la figura 3.7 se puede observar el modelo de referencia de EDCA. Cada cola tiene sus propios parámetros. El estándar permite que se definan más colas. Un parámetro específico para cada cola, que no se detalla en el gráfico, es el TXOP.

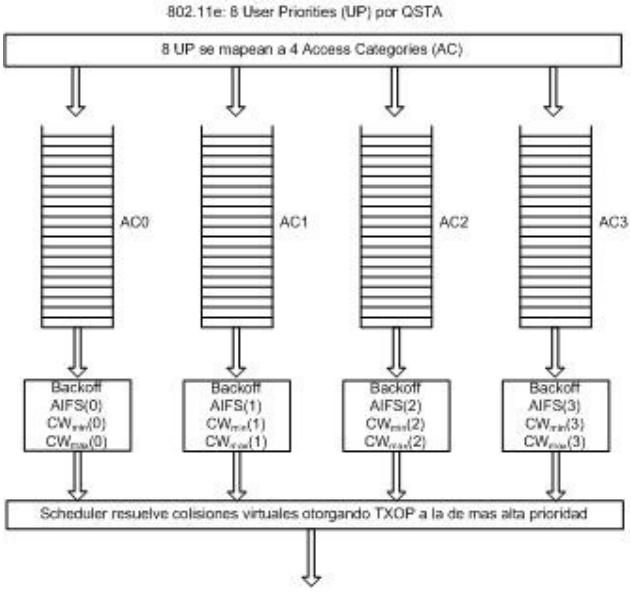



Figura 3.7 - Modelo de referencia EDCA

El acceso con prioridad al medio de EDCA es realizado asignando diferentes CWs y AIFS a cada una de las distintas ACs. Las unidades de datos se envían a través de múltiples instancias de backoff dentro de cada estación. Cada instancia de backoff es parametrizada con valores específicos de la categoría de tráfico.

La duración de los TXOPs es anunciada por los access points, con soporte para Calidad de Servicio, usando los elementos de información de las tramas beacons y probe response. Durante el tiempo de TXOP, la estación mantiene el control, sin interrupciones, del medio y puede transmitir múltiples tramas pertenecientes a la misma AC. Un frame será transmitido únicamente si el tiempo necesario para enviarlo más el correspondiente ACK es menor que el tiempo restante de TXOP que le queda a la estación. Si existen frames pendientes en otras ACs no serán transmitidos en el mismo TXOP.

### 4.4 - HCCA (HCF Controlled Channel Access)

El esquema de consulta de PCF es extendido en 802.11e mediante el uso de la función llamada Hybrid Coordination Function (HCF, Función de coordinación híbrida). En este método existe una funcionalidad provista por un Hybrid Coordinator (HC, Coordinador Híbrido), generalmente ubicado en el access point. Éste difiere del PC, utilizado en PCF, en varios aspectos pero, principalmente, en que puede funcionar en ambos periodos, el de contienda y el libre de contienda, lo que implica que no es obligatorio que utilice el periodo libre de contiendas para transferir datos de Calidad de Servicio. Otra diferencia es que el HCF puede consultar a las estaciones que soportan Calidad de Servicio y otorgarles TXOPs.

El HC obtiene el control del medio wireless cuando necesita enviar tráfico de datos o algún frame específico a las estaciones que soportan Calidad de Servicio. Para lograr esto, el HC espera una menor cantidad de tiempo entre

transmisiones al que tienen que hacerlo el resto de las estaciones usando los procedimientos de EDCA.

Para evitar que una estación sin soporte de Calidad de Servicio intente transmitir en un periodo libre de contenciones (como la estación no soporta Calidad de Servicio no entiende los nuevos parámetros definidos en el estándar 802.11e), el HC puede incluir el tiempo de duración del periodo de contención en los frames beacon para que la estación pueda establecer el valor del NAV y no intente transmitir mientras ese valor sea mayor a cero. Esto causa que la estación suponga que existe un PC en el BSS.

Antes de obtener el medio para comenzar un periodo libre de contienda, o un TXOP en el periodo de contienda, el HC debe sensar el medio. Si está libre por un lapso igual a PIFS, el HC transmitirá el primer frame del intercambio de frames correspondiente indicando la duración del TXOP o del periodo libre de contienda. En este intervalo de tiempo, llamado Fase de Acceso Controlado (CAP, Controlled Access Phase), el medio está bajo el control del HC. Si pasa un tiempo PIFS desde el final del TXOP y el HC no reclama el medio, el CAP finaliza.

El HCF también puede operar como un PC consultando a las estaciones que no tienen la capacidad de soportar Calidad de Servicio; para esto usa los formatos de frames, secuencia de intercambio de frames y todas las demás reglas especificadas para PCF.

### 4.5 - Mejoras 802.11e MAC

Además de proveer las funcionalidades de EDCA y HCCA discutida previamente, el estándar 802.11e contiene otras mejoras en la capa MAC.

#### 4.5.1 - Ráfaga Libre de Contención (Contention Free Burst - CFB)

Esta característica permite que una estación, o el access point, pueda enviar varios frames seguidos sin tener que disputar el acceso al medio. Una vez que un nodo obtiene su TXOP, puede enviar frames continuamente y lo hace mientras le quede tiempo en el TXOP obtenido. Para poder asegurarse el acceso al medio, el tiempo entre frames es SIFS.

A continuación se muestran dos gráficos en el que se ven las diferencias entre una red sin CFB, la figura 3.8, y con CFB en la figura a continuación.

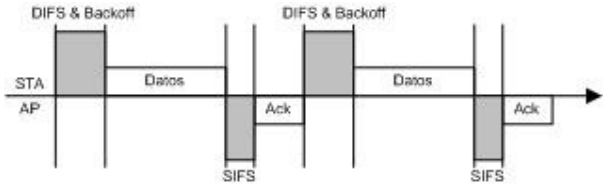



Figura 3.8 - Red sin CFB aplicado

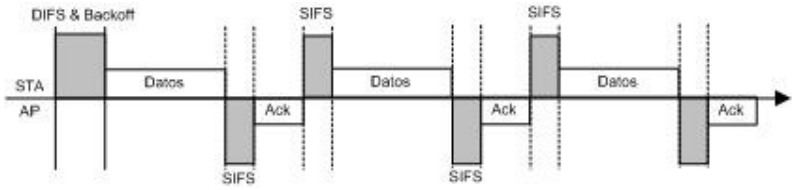



Figura 3.9 - Red con CFB aplicado

### 4.5.2 - Protocolo de enlace directo (Direct Link Protocol - DLP)

En una red infraestructura, según la especificación original del estándar 802.11, si dos estaciones se quieren comunicar entre ellas lo deben hacer a través del access point. DLP permite que dos estaciones se puedan comunicar directamente sin necesidad de usar el access point. Esto sólo puede funcionar si cada estación se encuentra dentro del área de cobertura de la otra.

En el siguiente gráfico se muestra el intercambio de paquetes necesarios para realizar el proceso de establecimiento de enlace directo.

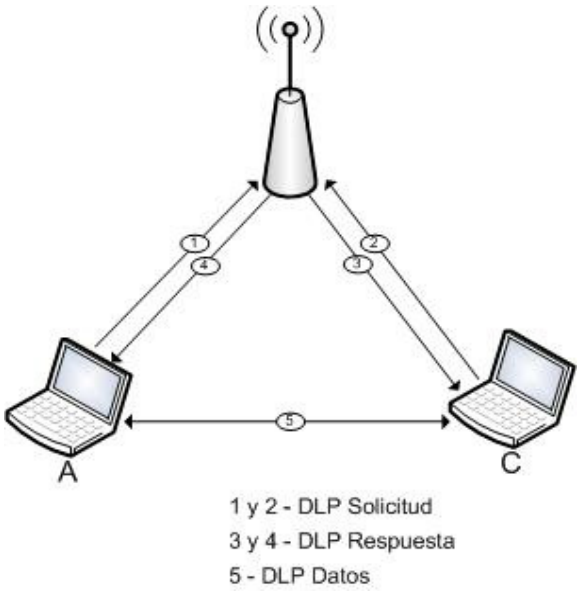



Figura 3.10 - Proceso del protocolo de enlace directo

### 4.5.3 - Nuevas reglas de acuse de recibo (New Acknowledgement Rules)

De acuerdo al estándar 802.11, para cada frame unicast se requiere un frame de control ACK. HCF agrega dos nuevas opciones que son especificadas en el campo de control de QoS de cada frame de datos:

- Sin ACK - con el fin de mejorar la eficiencia de determinadas aplicaciones no se requiere el envío de frames ACKs. Esta característica es muy útil en aplicaciones con una latencia muy baja, pero que pueden tolerar una cierta cantidad de pérdida de paquetes

- Bloque ACK - incrementa la eficiencia mediante el envío de un único ACK para varios frames. Puede ser de dos tipos:

    1. Inmediata: el receptor debe responder inmediatamente con un ACK al recibir una solicitud, de parte del emisor, pidiéndole que le envíe un Bloque ACK. Esto lo puede hacer el emisor después de haber enviado varios frames seguidos en un CFB durante un TXOP.

    2. Demorado: en este caso el receptor, al recibir la solicitud de un Bloque ACK, envía una respuesta indicando que el ACK será demorado. Esto le da más tiempo a los sistemas de baja performance para calcular el ACK.

### 4.5.4 - Piggybacking

Este método permite mejorar la perfomance de la red al enviar los datos "piggybacked" en los frames de consultas y ACKs. Esta función puede mejorar la performance total de la red.

# Capítulo 4

# Propuesta y simulaciones de las modificaciones realizadas

## 1 - Introducción

De lo explicado en los capítulos anteriores se deduce la necesidad de integrar las herramientas provistas por IPv6 con las suministradas en las redes wireless, a través de la especificación IEEE 802.11e, para brindar Calidad de Servicio.

La propuesta de esta tesis es la de proveer Calidad de Servicio en redes wireless usando el campo Flow Label que se encuentra en la cabecera de los paquetes IPv6. Este campo puede contener distintos valores que serán mapeados en alguna de las cuatro colas que se encuentran definidas en el estándar 802.11e.

El campo Flow Label de IPv6 fue diseñado para posibilitar un manejo eficiente de los paquetes que deben recibir un tratamiento especial, específico para el flujo de datos, por parte de los nodos. Al ubicar este campo en la capa de red con el fin de solicitar dicho tratamiento se resuelven algunos problemas que se encuentran en IPv4. Sin este campo, la clasificación del flujo debe basarse en la información contenida en la cabecera de la capa de transporte, por ejemplo los numero de puertos, en conjunto con la ubicada en la cabecera IPv4. Para esto, un nodo debe recorrer los paquetes buscando esta información, lo que implica cierto retardo en el reenvío de los mismos. Además, puede suceder que esa información no esté disponible porque el paquete está encriptado o fragmentado.

Son varios los trabajos realizados para dotar de un significado real al campo Flow Label. Este campo, de un tamaño de 20 bits, se encuentra especificado en el documento original de IPv6 pero su utilización no está determinada en una forma concreta. Al quedar su definición, en cierta forma abierta, es posible modificarlo para plantear nuevas alternativas en la manera de utilizarlo.

Como se explica en el prólogo, este trabajo define una nueva forma de utilizar el campo Flow Label para brindar Calidad de Servicio de una manera consistente en todo el camino que sigue un determinado flujo de datos. A lo largo de este capítulo se explicarán las modificaciones realizadas a dicho campo para lograr tal fin y, una vez detalladas esas modificaciones, se realizarán las pruebas necesarias para demostrarlo. Para esto último, se utiliza el simulador NS-2 de la Universidad de Berkeley. En el Apéndice 1 se explica la forma de instalar este simulador y como funciona. La versión utilizada para realizar las pruebas es la 2.28.

## 2 - Modificación al campo Flow Label

Son varios los parámetros que hay que tener en cuenta al momento de intentar brindar Calidad de Servicio. Entre ellos se encuentran Delay (Retardo), Buffer Requeriment (Requerimiento de Buffer), Bandwidth (Ancho de Banda), Jitter (Variación en el Retardo) y Packet Loss (Pérdida de paquetes). Utilizando el campo Flow Label, con las modificaciones propuestas, se indicarán todos estos parámetros, excepto el parámetro Buffer Requeriment. Si por algún motivo fuese necesario indicarlo explícitamente se podría utilizar la cabecera de extensión Hop-by-Hop Option, que es otra de las nuevas funcionalidades que nos brinda IPv6.

De acuerdo a lo explicado más arriba, es conveniente utilizar un campo de la cabecera IP para indicar qué Calidad de Servicio se desea. Otra ventaja de utilizar este campo es que tanto el medio inalámbrico como el medio cableado pueden utilizar el mismo valor para realizar el mapeo en las distintas colas de prioridades. Aunque los diferentes medios de transmisión tengan distintas maneras de proveer Calidad de Servicio, todos lo hacen a partir del mismo valor.

En la cabecera IPv6 existe un campo llamado Traffic Class que es del mismo tamaño que el campo Type of Service de la cabecera IPv4 y cumple la misma función. Este campo también es utilizado para indicar Calidad de Servicio pero considero que es mejor utilizar el campo Flow Label para hacerlo porque su tamaño más grande (20 bits contra los 8 bits del campo Traffic Class) permite indicar una mayor granularidad al momento de especificar los distintos requerimientos.

El valor indicado en el campo Flow Label es usado para mapearlo en una tabla que contiene los posibles valores que puede tener dicho campo. Esta tabla mantiene una relación entre uno, o varios, valores del campo Flow Label y una de las colas definidas en cada uno de los nodos. Esos valores deben ser universalmente aceptados y definidos en forma consistente en todos los hosts y routers de la red. Por ejemplo, todos los nodos deben estar de acuerdo en que el valor 0 en este campo significa un servicio default de mejor esfuerzo, es decir, no se requiere ningún tratamiento especial por parte de los nodos intermedios.

Al tener 20 bits disponibles la cantidad de flujos que se pueden especificar por cada tupla (IPv6 origen, IPv6 destino) es extremadamente grande ($2^{20}$). Administrar toda esta cantidad posible de flujos puede ser realmente difícil y complicado. Sin dudas, es conveniente dividirlo para poder utilizarlo en una manera más simple.

Como son 4 los parámetros que se necesitan indicar directamente, el campo Flow Label es dividido en 4 partes. Cada una de estas partes indica un parámetro determinado. A continuación se explican cada una de las divisiones realizadas y el tamaño respectivo.

La primera parte es utilizada para indicar el Bandwidth (Ancho de Banda) y tiene un tamaño de 5 bits, lo que permite especificar un ancho de banda de $2^b$ * 32 bits, donde b es el valor decimal del campo Bandwidth y 32 bits es la unidad de solicitud. El valor máximo posible que se puede indicar es $2^{30}$ * 32, que representa 32 Gbits, y es un valor muy superior a lo que cualquier aplicación puede requerir hoy en la actualidad y lo será, sin dudas, por bastante tiempo más. Si en algún momento esa capacidad es sobrepasada y es necesario indicar un valor mayor, la unidad de solicitud podría ser modificada a Kbits, Mbits o lo que fuese necesario, en lugar de bits. También se puede multiplicar por algún número mayor a 32 (64, 128, 256, etc.). Cualquiera de estas dos alternativas se pueden utilizar para lograr el incremento necesario.

La siguiente parte en la que se divide el campo es para indicar el Delay (Retardo) y ocupa 5 bits. El tiempo se indica en milisegundos y está dado por la fórmula $2^b$ * 4. Si se necesita utilizar un valor menor se puede indicar el resultado en microsegundos, por ejemplo.

El Jitter (Variación en el Retardo) se indica en la siguiente parte. Ocupa 5 bits y tiene la misma fórmula que el Delay. También puede ser modificado como en el campo Delay si se necesita indicar un tiempo menor.

Y por último, en la cuarta parte se indica el Packet Loss (Pérdida de Paquetes). El valor de este campo debe ser mayor a cero e indica cuál es el porcentaje máximo de paquetes que no pueden llegar a destino del total de paquetes enviados. Obviamente, cuanto menor es este porcentaje menor es la cantidad de paquetes que se pueden perder.

Existen aplicaciones que necesitan una pérdida de paquetes entre el 0% y el 1%. La forma de indicar esto es utilizando potencias con exponentes negativos. Para hacerlo se destina el primero de los 5 bits. Si el valor del primer bit es 0, el exponente es positivo y, si es 1, es negativo. Los 4 bits restantes sirven para indicar el valor decimal que reemplazará a b en la fórmula de pérdida de paquetes.

Hay que recordar que para indicar que los paquetes no necesitan ningún tratamiento especial por parte de los nodos se usa el valor cero en todo el campo Flow Label, lo que implica que el valor cero en cada uno de los campos está reservado. El único campo que puede indicar un valor cero para su tratamiento es el campo Packet Loss, no se acepta pérdida de paquetes en la transmisión. Para indicar el valor 0 en ese campo se utilizan todos 1. Por ejemplo, si se quiere solicitar que el porcentaje de pérdida de paquetes sea cero, se deben poner todos 1 en el último campo (los últimos 5 bits del campo Flow Label). Obviamente, no tiene sentido solicitar 0 Mbits/s de Bandwidth ó 0 milisegundos de Delay o Jitter, por eso no es definido el valor 0 para estos 3 parámetros.

Para indicar todos los valores posibles se utiliza la siguiente fórmula:

- todos 0s, indica sin Calidad de Servicio
- todos 1s, indica 0. Sin pérdida de paquetes
- 1..b, donde b es el valor de los 4 bits y el primer bit es 0

- $10^{-1}..10^{-b}$, donde b es el valor decimal de los 4 bits y el primer bit es 0, exponente negativo

A continuación se muestra como queda compuesto el campo Flow Label con las divisiones propuestas:

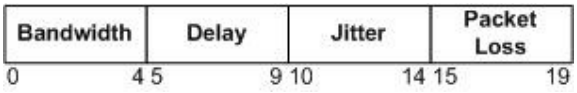



Figura 4.1 - Nuevo diseño campo Flow Label

Como se puede observar en el gráfico anterior, el campo Flow Label está dividido en 4 partes y cada uno de ellos se define de la siguiente manera

- Bandwidth = $2^b * 32$
- Delay = $2^b * 4$
- Jitter = $2^b * 4$
- Packet Loss
  - todos 0 => sin calidad de servicio
  - todos 1 => sin pérdida de paquetes
  - 1..15% de pérdida de paquetes
  - $10^{-1}..10^{-15}$ de pérdida de paquetes

donde b es el equivalente decimal del valor especificado por los 5 bits correspondientes a cada campo.

Una vez determinado el nuevo formato del campo hay que establecer la manera en que se van a mapear todos los posibles valores del campo en las distintas colas que proveen Calidad de Servicio. Es obvio, que el total de combinaciones que se pueden hacer con las 4 partes en las que se dividió el campo Flow Label es muchísimo mayor que la cantidad de colas que constituyen cualquier técnica de Calidad de Servicio.

Como se explicó en el capítulo anterior, el estándar 802.11e define un total de 4 colas (recordar que se pueden definir más colas) por lo que el total de combinaciones posibles es $2^{20}$. Esto significa que cada cola definida podría llegar a representar una cantidad muy grande de valores. Es por esto, que cada implementación está habilitada para realizar el mapeo de la forma que mejor se ajuste a sus necesidades. Lo primordial del mapeo es que, sin importar como se lo define, sea consistente en toda la red.

A continuación se muestra la tabla utilizada para realizar el mapeo del campo Flow Label a una de las 4 colas del estándar 802.11e. Ésta se definió de acuerdo a los valores descriptos en los distinos documentos para cada uno de los parámetros. Sin embargo, estos valores pueden ser redefinidos por las distintas implementaciones según sus necesidades y requerimientos.

Dependiendo del parámetro, la comparación del valor indicado por la aplicación con los correspondientes valores definidos en la tabla puede ser mayor o menor. Por ejemplo, para el caso del bandwitdth, si se quiere solicitar la mayor prioridad, el valor indicado debe ser mayor al valor definido en la primera fila de la columna Bandwidth. En cambio, si se quiere indicar alguno de los otros parámetros (Delay, Jitter o Packet Loss), el valor debe ser menor al establecido en la fila correspondiente a la prioridad que se desea.

| Bandwidth | Delay | Jitter | Packet Loss | 802.11e Queue |
|---|---|---|---|---|
| 5 | 150 | 50 | 1 | 0 |
| 2 | 400 | 100 | 5 | 1 |
| 1 | 1000 | 200 | 10 | 2 |
| 0,5 | ∞ | ∞ | ∞ | 3 |

Tabla 4.1 - Relación parámetros QoS con las colas 802.11e

Los valores en el campo Bandwidth están dados en Megabits por segundo, en milisegundos los campos Delay y Jitter y en porcentaje el campo Packet Loss. En los parámetros Delay, Jitter y Packet Loss se usa el símbolo de infinito (∞) para indicar que todos los valores que no entran en los tres primeros rangos entrarán en éste.

## 3 - Modificaciones al NS-2

Para poder implementar lo propuesto en este trabajo es necesario realizar algunas modificaciones al código fuente del simulador. Estos cambios son necesarios para poder realizar el mapeo del campo Flow Label en una de las 4 colas que proporciona el estándar 802.11e.

### 3.1 - Descripción

Después de instalar el simulador como se indica en el Apéndice 1, se realizaron las modificaciones correspondientes. Dentro del directorio ../ns-2.28/mac/802.11e se encuentran varios archivos, desarrollados en Tcl y C++, que sirven para implementar el funcionamiento del estándar 802.11e en el simulador:

a) priority.tcl
   En este archivo se encuentran los parámetros para cada una de las 4 colas que se definen en el estándar. Para cada una de las colas se indica el CWMax, CWMin, TXOP Limit, PF y AIFS. Modificando algunos de estos parámetros se puede modificar las prioridades de las colas correspondientes.

b) priq.h/priq.cc
   En esta clase se definen la cantidad de colas, 4 según el estándar, y es la encargada de asignar los frames en su correspondiente cola. La prioridad de cada uno de los frames viene indicada en su cabecera. Simplemente define un arreglo de cuatro colas de tipo Drop-Tail. Si se quiere definir una nueva cola se debe indicar acá y los parámetros que la definen en el archivo priority.tcl.

c) d-tail.h/d-tail.cc
En esta clase se redefine el tipo de cola drop-tail que se utiliza en 802.11. Es la encargada de mantener los distintos parámetros de prioridad de cada cola, una vez que la cola ha sido creada.

d) ns-mobilenode_802_11e.tcl
Este archivo tiene algunas modificaciones con respecto al archivo ns-mobilenode.tcl que se encuentra dentro del directorio ../ns-2.28/lib/tcl. Es el encargado de asociar el nodo móvil con la interface de red, con la capa mac, etc. Además, desde este archivo se llama al archivo priority.tcl para obtener los valores de prioridad de las colas.

De los archivos descriptos, únicamente, d-tail y priq tienen acceso a la prioridad del paquete, para leerlo o modificarlo.

### 3.2 - Transformar un script 802.11 en 802.11e

Para transformar un script, un archivo tcl, que simula una red 802.11 en otro que sea capaz de soportar el estándar 802.11e se le deben realizar muy pocas modificaciones.

En primer lugar, se debe indicar al simulador que se va a cambiar el tipo de MAC, para que soporte el estándar 802.11e, y el tipo de encolado, para que soporte las colas con prioridad. Esto se hace modificando las variables ifq (tipo de cola) y mac (tipo de MAC):

```
set val(ifq)        Queue/DTail/PriQ;
set val(mac)        Mac/802_11e;
```

Además, se debe indicar la prioridad que va a tener el tráfico generado por el agente de cada nodo. Esto se indica con la variable prio_ después de la declaración del agente. Sólo se permiten valores de prio_ entre 0 y 3 (recordar que el estándar 802.11e declara solamente 4 colas de prioridades) siendo 0 el de mayor prioridad y 3 el de menor.

```
set udp_1 [new Agent/UDP]
udp_1 set prio_ 0
```

También es posible modificar los parámetros de cada una de las colas cambiando los valores declarados en el archivo priority.tcl. Es necesario recompilar el simulador si se realizan los cambios en este archivo, pero esto no es obligatorio si lo único que se modifica es el script que describe la simulación.

### 3.3 - Como funciona el 802.11e en NS-2

Una vez realizadas las modificaciones anteriores, el script es compatible con el estándar, específicamente con EDCA. De todas las modificaciones, el valor asignado a la variable prio_ es el que determina cual es la prioridad del tráfico.

Esta variable forma parte de la cabecera del paquete IP. En los archivos i.e. /Inc., que se encuentran en el directorio.../ns-2.28/common/, se define la clase que implementa la cabecera de IPv6 (también implementa la cabecera IP). En la definición de la clase se declara la variable prio_ que es donde se guarda la prioridad de cada paquete. También, se definen los procedimientos adecuados para leer o modificar el valor de dicha variable.

El procedimiento para asignar la prioridad a cada paquete es invocado desde una clase llamada Agente, que se encuentra definida parcialmente en C++ en los archivos agente/agente, ubicados en el directorio .../ns-2.28/common, y en OTcl, en el archivo ns-agent.tcl que se encuentra en el directorio ../ns-2.28/tcl/lib. Como se explicó más arriba, la prioridad del tráfico se asigna al agente. Los agentes son los puntos finales en donde los paquetes se generan o consumen y existen distintos tipos de agentes (TCP en distintas versiones, UDP, RTP).

Antes de comenzar con las modificaciones es importante mostrar como mejora el rendimiento de determinadas aplicaciones el estándar 802.11e. Es por esto, que a continuación se muestra un ejemplo. A partir de una topología de red determinada se realizan dos simulaciones, una que no tiene aplicada Calidad de Servicio y otra que sí la tiene. Luego, los resultados de ambas simulaciones son comparados. Todas las simulaciones realizadas a lo largo de esta tesis siguen este mismo patrón: se ejecutan dos veces, una con Calidad de Servicio y otra sin ésta, y se comparan los resultados.

A continuación se muestra cual sería el rendimiento de una red wireless sin el estándar 802.11e aplicado. El script utilizado en esta simulación se detalla en el Apéndice 2. El primer gráfico muestra la disposición de la red. Esto se hace utilizando la aplicación Nam.

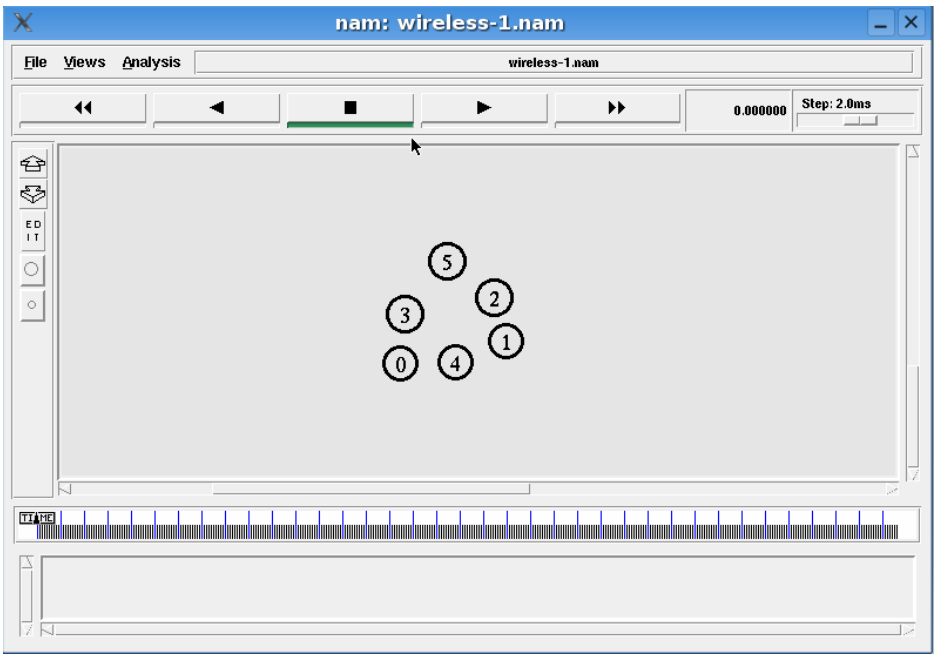



Figura 4.2 - Topología red wireless

Como se puede observar en el gráfico anterior, la red wireless utilizada en esta prueba es una red del tipo ad-hoc, compuesta por 6 nodos móviles. Se generan un total de cinco conexiones. Todas son iguales en cuanto a la cantidad de tráfico generado por segundo. En la siguiente tabla se muestra como se componen los tráficos:

| Tráfico | Origen | Destino | Inicio(seg.) | Prioridad |
|---|---|---|---|---|
| Tráfico_0 | 3 | 0 | 40 | 2 |
| Tráfico_1 | 4 | 1 | 0 | 3 |
| Tráfico_2 | 5 | 2 | 30 | 1 |
| Tráfico_3 | 1 | 3 | 20 | 2 |
| Tráfico_4 | 2 | 4 | 10 | 0 |

Tabla 4.2 - Definición de tráficos

En el gráfico a continuación se muestra el resultado de ejecutar el script RedWireless_2.1 del Apéndice 2. No se aplica ningún parámetro de Calidad de Servicio. El primer tráfico generado, Trafico_1, comienza en el instante 0 de la simulación, los restantes lo hacen cada 10 segundos. En los primeros 10 segundos, Trafico_1 consume un total de 56 KBytes/s. A los 10 segundos de iniciada la simulación se inicia el segundo tráfico, Trafico_4, y el ancho de banda total se empieza a compartir entre ambos tráficos en forma casi simétrica. Lo mismo sucede a los 20, 30 y 40 segundos de tiempo de simulación. Cada vez que se inicia un nuevo tráfico el ancho de banda que recibe cada tráfico es similar y, a medida que se agreguen nuevas conexiones, el rendimiento de las restantes se va degradando. Si alguna de las conexiones necesita cumplir con determinados requisitos en su rendimiento no tenemos forma alguna de cumplir con esos requerimientos.

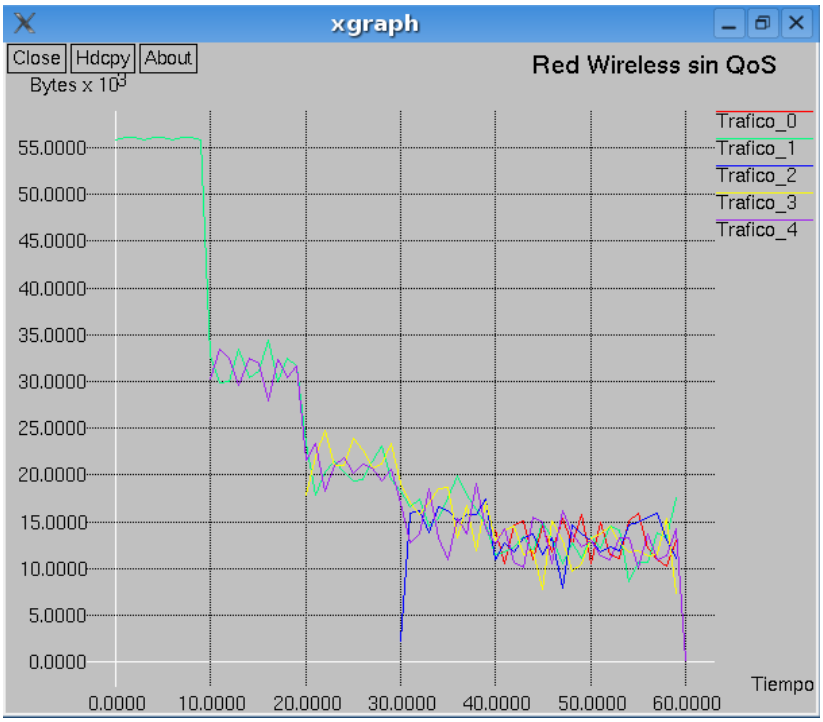



Figura 4.3 - Resultado simulación sin Calidad de Servicio

A continuación se muestra el gráfico del resultado de ejecutar el script RedWireless_2.2 que se encuentra en el Apéndice 2. Este script es igual al ejecutado en el ejemplo anterior con la diferencia que ahora se aplican técnicas de Calidad de Servicio, específicamente el estándar IEEE 802.11e.

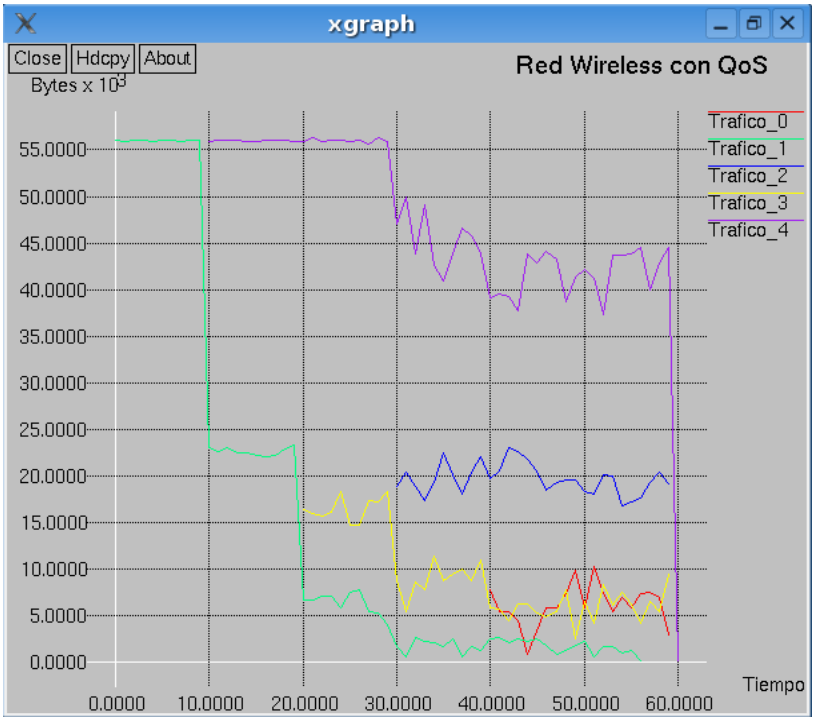



Figura 4.4 - Resultado simulación con Calidad de Servicio

Como se puede observar, el rendimiento de la red es completamente diferente al de la red que se examinó en el gráfico anterior. Para poder realizar la comparación entre ambas redes se mantiene la misma cantidad de tráficos y los mismos instantes de inicio de las transmisiones.

En el segundo 0 de la simulación se inicia Trafico_1 que tiene prioridad 3 (recordar que 0 es la mayor prioridad y 3 la menor). Como sucede en la anterior simulación, esta transferencia consume un total de 56 KBytes/s durante los primeros 10 segundos. A los 10 segundos se inicia la siguiente conexión, Trafico_4, con prioridad 0. Esta conexión es la que consigue el mayor ancho de banda durante toda la simulación, lo cual es correcto porque es la que tiene la mayor prioridad. En el instante 20, Trafico_3, con prioridad 2, empieza a enviar tráfico. Trafico_4 mantiene su tasa de transferencia pero no sucede lo mismo con Trafico_1. Éste ve decrementado su rendimiento a causa de Trafico_3. Al iniciarse Trafico_2, con prioridad 1, todos los tráficos ven deteriorado su rendimiento, inclusive el tráfico de mayor prioridad, Trafico_4. El último tráfico en iniciarse es Trafico_0, con prioridad 2, la misma prioridad que Trafico_3. Como se puede observar en el gráfico, estos dos tráficos tienen prácticamente el mismo rendimiento. Trafico_4 sigue siendo el tráfico que más ancho de banda consume. De sus 56 KBytes/s iniciales baja aproximadamente 10 KBytes/s entre los 30 y 40 segundos para terminar consumiendo, en promedio, entre 38 y 44 KBytes/s en el resto de la simulación.

### 3.4 - Modificaciones al código del NS-2

Para permitir que el simulador NS-2 pueda proveer Calidad de Servicio usando el campo Flow Label de la cabecera de IPv6 y mapear los valores de dicho campo en alguna de las 4 colas definidas en el estándar IEEE 802.11e es necesario realizar las siguientes modificaciones al código fuente del NS-2.

Como se explica en el Apéndice 1, el simulador se instaló en el directorio /ns-allinone-2.28. Dentro de este directorio existe un subdirectorio llamado ns-2.28 que es donde se encuentra todo el código del simulador y es ahí donde se realizaron la mayoría de las modificaciones al código. A partir de ahora, la expresión ../ será una referencia al path /ns-allinone-2.28/ns-2.28/.

El funcionamiento básico del simulador se encuentra en el directorio ../common. Acá se encuentran especificadas varias clases (en C++) que implementan funcionalidades comunes del simulador como, por ejemplo, la clase Node que se utiliza para representar un nodo en la red cableada (ya sea un nodo final o un nodo intermedio en la red), la clase MobileNode para las redes wireless, etc. También se define el protocolo IP, con la clase IP, y la clase Agent que permite definir los agentes.

En primer lugar hay que definir de qué manera se va a representar el campo Flow Label en la cabecera IP de acuerdo a la propuesta de modificación explicada anteriormente. En la implementación del simulador, la cabecera IP tiene definido varios campos entre los que se encuentra origen, destino, tiempo de vida, prioridad e identificación de flujo.

El valor del campo prioridad es utilizado por la implementación del estándar 802.11e para determinar a qué cola se asignan los paquetes, por lo tanto debe tomar un valor entre 0 y 3 (las 4 colas que define el estándar). Esto lo transforma en el campo ideal para ser utilizado en el mapeo del campo Flow Label en una de las colas de dicho estándar pero no puede ser utilizado directamente, hay que hacer ciertas modificaciones.

El valor del campo Flow Label se determina al momento de escribir el script de la simulación. Este campo, de acuerdo a la propuesta detallada arriba, está dividido en 4 partes por lo que existen dos posibilidades diferentes para asignarle un valor. La primera es mediante un único parámetro que represente a todo el campo y la segunda es usando un parámetro por cada una de las cuatro partes que componen el campo. Teniendo en cuenta la estructura de la cabecera IPv6 la primera forma es la correcta pero a los efectos de lograr más claridad en la especificación de los parámetros se decidió utilizar la segunda. Es decir, el campo Flow Label se representa por 4 parámetros que tienen el mismo nombre que las correspondientes partes del campo.

Los 4 parámetros se especifican en la clase IP que está definida por los archivos ../common/ip.h y ../common/ip.cc.

Además, en el archivo de cabecera ip.h, se define el tamaño de la cabecera IP. Como el simulador es para IPv4 el tamaño de la cabecera es 20

bytes. Pero como en IPv6 el tamaño de la cabecera es de 40 bytes ese valor se debe modificar. Esto se hace modificando la siguiente constante en dicho archivo:

```
#define IP_HDR_LEN          40
```

Si se deja este valor en 20, el simulador igual funciona pero no serían correctos los datos obtenidos, especialmente en cuanto al throughput se refiere.

Los campos de la cabecera IP están definidos por una estructura, o registro como se lo conoce en varios lenguajes de programación, en el archivo ../common/ip.h. Según lo explicado más arriba se le deben agregar 4 nuevas variables para representar conjuntamente el campo Flow Label. Esto se realiza de la siguiente manera:

```
struct hdr_ip {
        …
        int bandwidth_;
        int delay_;
        int jitter_;
        int packetloss_;
        …
        }
```

Además, en el mismo archivo de cabecera (ip.h) y, también dentro de la definición de la estructura, se definen las funciones miembros de acceso a los campos, una por cada nueva variable, lo que implica que se deben especificar 4 nuevas funciones.

```
int& bandwidth() { return (bandwidth_); }
int& delay() { return (delay_); }
int& jitter() { return (jitter_); }
int& packetloss() { return (packetloss_); }
```

En el archivo de implementación de la clase C++ que define la cabecera IP, ../common/ip.cc, se deben agregar las siguientes sentencias que retornan el offset, en bytes, de los variables:

```
field_offset (“bandwidth_”, OFFSET(hdr_ip, bandwidth_));
field_offset (“delay_”, OFFSET(hdr_ip, delay_));
field_offset (“jitter_”, OFFSET(hdr_ip, jitter_));
field_offset (“packetloss_”, OFFSET(hdr_ip, packetloss_));
```

Dentro del simulador, un *agente* es un “punto final” que produce o consume paquetes de red y es el nombre común para la mayoría de los protocolos de transporte. Estos agentes están conectados a su clase padre llamada Agent. Esta clase está implementada en C++, en los archivos ../common/agent.h y ../common/agent.cc, y en OTcl en el archivo ../tcl/lib/ns-agent.tcl.

Con el fin de poder asignarle valores a las 4 nuevas variables definidas en la clase IP, se definieron las mismas variables en el archivo de cabecera de la clase Agent, ../common/agent.cc (la clase Agent desciende de la clase Connector). Como se explica más arriba, en el script donde se define la simulación la prioridad del tráfico se asigna al agente que, a su vez, se la pasará a la cabecera IPv6:

```
class Agent : public Connector {
        …
        int bandwidth_;
        int delay_;
        int jitter_;
        int packetloss_;
        …
        }
```

El script donde se especifica la simulación se escribe en OTcl pero las clases que componen el funcionamiento del simulador están implementadas en C++, por lo cual debe existir una manera de pasar los valores indicados en el script a las clases. Para hacerlo se deben vincular las variables miembro de C++ con las variables objeto de OTcl. Existen dos maneras de realizar esta vinculación para permitir el intercambio de datos entre C++ y OTcl: binding de variables y llamadas a funciones. El método utilizado en este caso es el de binding de variables. Para cada una de las 4 variables se especificaron las correspondientes sentencias encargadas de realizar el binding. Todas estas modificaciones se realizaron en el archivo de implementación de la clase, ../common/agent.cc.

En primer lugar, dentro de la función delay_bind_init_all(), se agrega la siguiente sentencia:

```
delay_bind_init_one("bandwidth_");
```

Y, en la function delay_bind_dispath(), se agrega esta sentencia:

```
if (delay_bind(varNane, localName, "bandwidth _", /
                (int*)& bandwidth_, tracer)) return TCL_OK;
```

Estas dos sentencias se repiten para cada una de las 4 nuevas variables que se agregaron (además de bandwidth se agregan para jitter, delay y packetloss) y se usan para realizar el binding de las variables, con lo cual, cuando en el script OTcl se le asigna un valor a una de estas variables automáticamente es conocido por la respectiva variable declarada en C++.

La clase Agent es la encargada de crear los paquetes y contiene un método llamado initpkt() que, además de ser utilizado por las clases derivadas para generar los paquetes a enviar, se encarga de llenar los siguientes campos en la cabecera IPv6: origen, destino, prioridad, tiempo de vida e identificador de flujo. Como las 4 nuevas variables se incluyen dentro de la cabecera IPv6, los valores se le asignan de la misma forma. Es por esto, que dentro de ese método, se agregan las siguientes sentencias:

```
iph->bandwidth() = bandwidth_;
iph->delay() = delay_;
iph->jitter() = jitter_;
iph->packetloss() = packetloss_;
```

Por último, todas las variables vinculadas deben ser inicializadas al momento de crearse el objeto al que pertenecen, si esto no se hace, se produce un error cuando se intenta compilar el simulador (al ejecutar el comando make). Esto se realiza en el siguiente archivo ../tcl/lib/ns-default.tcl y fueron agregadas las siguientes sentencias:

```
Agent set bandwidth_ 0
Agent set delay_ 0
Agent set jitter_ 0
Agent set packetloss_ 0
```

Antes de continuar es necesario explicar como determina el simulador la prioridad de cada paquete para mapearlo en una de las 4 colas que define el estándar 802.11e. En el script se indica la prioridad usando el campo Prio de la cabecera IPv6. Este valor, también, debe ser conocido por las clases encargadas de implementar el estándar 802.11e. Es por esto que en el archivo ../mac/802.11e/priq.cc está la siguiente sentencia, que permite obtener la prioridad de cada paquete:

```
#define PKT_LEVEL (p)   HDR_IP(p)_>prio();  (1)
```

Pero, si la función prio() en la cabecera IP es la que retorna la prioridad de cada paquete, ¿cómo puede hacerse lo mismo con las 4 variables que representan el campo Flow Label? Esto implica que se debe lograr que las modificaciones que se realicen al código del simulador no interfieran con la forma normal de asignar las prioridades. Es decir, sin importar que método se utilice para indicar la prioridad (con el campo Flow Label o directamente con el campo Prio) la sentencia indicada en (1) debe servir indistintamente para ambos métodos. Lograr esto es fundamental para que se puedan aplicar estas modificaciones al simulador sin interferir con cualquier otra forma de usarlo.

Para conseguirlo es necesario copiar el valor que indica la prioridad de cada paquete, obtenido a partir de los 4 parámetros del campo Flow Label, en el campo Prio dentro de la cabecera IPv6, con lo cual, cuando se ejecuta la sentencia (1) se obtiene la prioridad indicada en el campo Flow Label. A continuación se muestra el código que realiza esto:

```
if (bandwidth_ != 0 || delay_ != 0 || jitter_ != 0 || packetloss_ != 0) {
        if (bandwidth_ > 0) {
                if (bandwidth_ >= qosvalues[0][0])
                        iph->prio() = qosvalues[0][4];
                   else if (bandwidth_ >= qosvalues[1][0])
                        iph->prio() = qosvalues[1][4];
                   else if (bandwidth_ >= qosvalues[2][0])
                        iph->prio() = qosvalues[2][4];
                   else
                        iph->prio() = qosvalues[3][4];
        }
```

```
        if (delay_ > 0) {
                if (delay_ <= qosvalues[0][1])
                        iph->prio() = qosvalues[0][4];
                else if (delay_ <= qosvalues[1][1])
                        iph->prio() = qosvalues[1][4];
                else if (delay_ <= qosvalues[2][1])
                        iph->prio() = qosvalues[2][4];
                else
                        iph->prio() = qosvalues[3][4];
        }

        if (jitter_ > 0) {
                if (jitter_ <= qosvalues[0][2])
                        iph->prio() = qosvalues[3][4];
                else if (jitter_ <= qosvalues[1][2])
                        iph->prio() = qosvalues[2][4];
                else if (jitter_ <= qosvalues[2][2])
                        iph->prio() = qosvalues[1][4];
                else
                        iph->prio() = qosvalues[0][4];
        }

        if (packetloss_ > 0) {
                if (packetloss_ <= qosvalues[0][3])
                        iph->prio() = qosvalues[3][4];
                else if (packetloss_ <= qosvalues[1][3])
                        iph->prio() = qosvalues[2][4];
                else if (packetloss_ <= qosvalues[2][3])
                        iph->prio() = qosvalues[1][4];
                else
                        iph->prio() = qosvalues[0][4];
        }
}
```

Si los 4 subcampos en los que se divide el campo Flow Label tienen valor 0 no se requiere Calidad de Servicio, por lo que no es necesario que se ejecute este código. La tabla 4.1 se representa en el código del simulador como una matriz, tiene el nombre "*qosvalues*", y contiene los mismos valores que los definidos en esa tabla. Las comparaciones entre los valores recibidos en el campo Flow Label y los definidos en la matriz permite determinar cuál es la prioridad del paquete, es decir, la cola, de las 4 definidas en el estándar 802.11e, a la que serán asignados. Estas comparaciones son contra un rango de valores, por ejemplo, contra todos los valores que se encuentran entre la fila 2 y 3 de una de las 4 columnas de la matriz, y no contra un único valor. También, se puede utilizar más de un subcampo simultáneamente para realizar las comparaciones o hacerlas contra un único valor y no contra un rango, como se realiza en este trabajo. La forma de hacerlo queda a criterio de los implementadores.

## 4 - Pruebas realizadas para medir la Calidad de Servicio

Una vez finalizadas todas las modificaciones al NS-2 se diseñó e implementó el script para realizar las pruebas necesarias y así comprobar si realmente las modificaciones funcionaban como se esperaba. En el Apéndice 2 se encuentran los scripts utilizados en las pruebas. La red diseñada consiste de 9 nodos y es una red inalámbrica que concuerda con el estándar IEEE 802.11b (11 Mbits/s). Aunque existen redes inalámbricas de mayor velocidad no fueron utilizadas porque lo que se desea es generar el tráfico necesario para saturar la red y poder ver que se priorizan determinados tráficos sobre otros. Si se hubiese utilizado una red con mayor velocidad, como es el caso de una red que cumpla con los estándares 802.11a u 802.11g, simplemente se tendría que haber modificado el script para generar una mayor cantidad de tráfico hasta lograr saturar la red.

De los 9 nodos definidos, 6 de ellos generan tráfico que se discriminan de la siguiente manera: dos son tráficos de voz (tráfico UDP a 64 Kbits/s) y son los de mayor prioridad, otros dos son de video (tráfico UDP con velocidades de 2 Mbits/s, 2.5 Mbits/s y 4 Mbits/s según la simulación) y de menor prioridad que los tráficos de voz, otra es una conexión Telnet (tráfico TCP) y la última es una conexión de FTP (tráfico TCP) que es la de menor prioridad.

El tráfico generado en las conexiones de voz y video es constante. En estas conexiones el generador de tráfico es del tipo CBR, Constant Bit Rate. Las otras dos conexiones son TCP, donde una es del tipo FTP en la cual el emisor está continuamente enviando nuevos datos. Por último, la conexión restante es del tipo Telnet en la cual el tiempo de envío entre dos paquetes consecutivos sigue una distribución exponencial con promedio de envío calculado en base al valor de la variable *interval_*, que debe ser distinta de 0. 100 ms es el valor escogido en estos ejemplos.

La simulación se ejecuta por un tiempo total de 60 segundos y cada uno de los tráficos definidos comienza en distintos momentos. La configuración de los tráficos definidos en la simulación queda de la siguiente manera:

| Tráfico | Origen | Destino | Tipo | Prioridad | Inicio(seg.) | Tasa |
|---|---|---|---|---|---|---|
| 1 | 8 | 1 | Voz | 0 | 1 | 64 Kb/s |
| 2 | 0 | 3 | Voz | 0 | 2 | 64 Kb/s |
| 3 | 1 | 5 | Video | 1 | 4 | 2 Mb/s |
| 4 | 7 | 2 | Video | 1 | 5 | 2 Mb/s |
| 5 | 2 | 0 | FTP | 3 | 3 | * |
| 6 | 4 | 7 | Telnet | 2 | 6 | ** |

Tabla 4.3 - Definición tipos de tráficos

* El simulador envía tráfico constantemente cuando la aplicación se declara como FTP

** El tiempo de llegada inter-paquetes sigue una distribución exponencial con un promedio de 100ms. Como el tamaño de los paquetes es de 400 bytes, el tráfico promedio de la conexión es de 32Kbits/seg

Con estas condiciones establecidas se ejecutó el script, el cual arrojó como resultado un archivo con extensión tr. A partir de este archivo se

realizaron todos los cálculos necesarios (bandwidth, jitter, delay y packet loss) para determinar el funcionamiento de cada uno de los tráficos en particular. Con el fin de poder obtener una mejor perspectiva, el mismo script se ejecutó sin las características de Calidad de Servicio configuradas. Todos los cálculos se realizaron en ambos archivos de salida y se compararon los resultados obtenidos.

Para ver los distintos comportamientos de la red, a medida que se incrementa el tráfico, se realizaron 3 pruebas diferentes. En cada una de las pruebas se aumentó el tráfico generado por cada una de las conexiones de video. Siguiendo con el patrón establecido, cada una de las 3 pruebas se realizaron dos veces, con y sin Calidad de Servicio. A continuación de cada gráfico se muestra una tabla con los resultados detallados de cada conexión.

En esta primera prueba, las conexiones de video envían sus paquetes a una tasa de 2 Mbits/s cada una. La conexión FTP empieza a los 3 segundos con una velocidad de transferencia mayor a los 3,5 Mbits/s. Pero, al comenzar las conexiones de video en los segundos 4 y 5, esa velocidad baja hasta los 1,2 Mbits/s de promedio y se mantiene así hasta el final de la simulación. Los tráficos de voz, que son los primeros en empezar a transmitir, comienzan en los segundos 1 y 2. Están representadas por las líneas roja y verde en el gráfico pero sólo se observa la línea roja entre los segundos 1 y 2. A partir de los 2 segundos, únicamente se observa la línea verde de la conexión Voz_2. Pero, aunque no esté visible, la línea roja, perteneciente a Voz_1, se encuentra debajo la línea verde ya que las dos conexiones consumen el mismo ancho de banda durante toda la simulación, 64 Kbits/s.

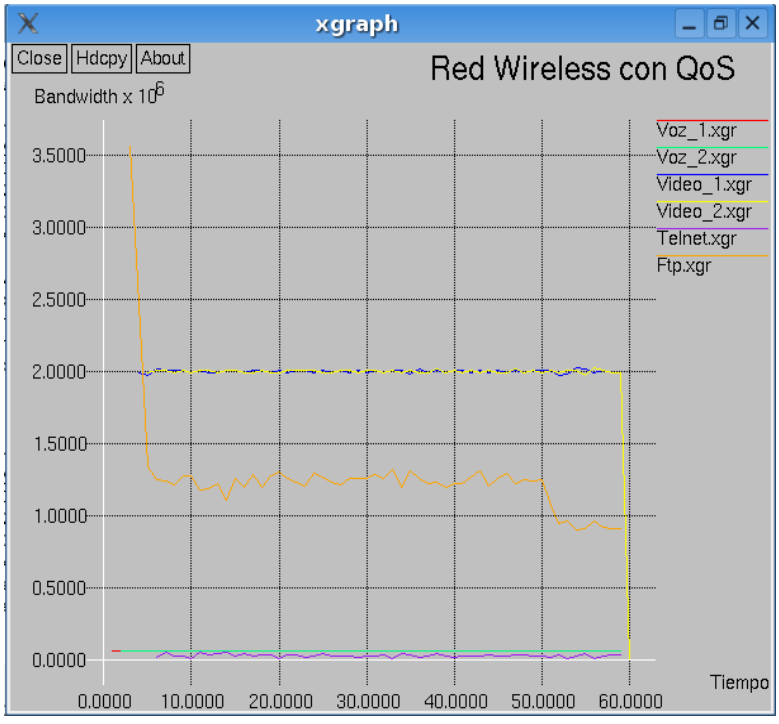



Figura 4.5 - Rendimiento con QoS y conexiones de video a 2 Mbits/s

En la siguiente tabla se muestran los datos estadísticos de todas las conexiones de la simulación. Como se puede observar, la conexión que pierde más paquetes es FTP pero, obviamente, TCP se encargará de recuperarlos reenviándolos. Esto provoca que el tiempo total de transferencia del archivo sea mayor pero la conexión, gracias a TCP, no se termina abruptamente (salvo que se produzca un timeout porque existe demasiada pérdida de paquetes, pero esto no debería pasar con la carga de tráfico existente en esta prueba). Video_1 pierde únicamente un paquete que no es reenviado ni por el protocolo de transporte, es UDP, ni por la aplicación. En las conexiones de voz y video los paquetes perdidos no se deberían reenviar. Tanto las conexiones de voz, como las de video, obtienen el ancho de banda, delay y jitter requerido para cumplir con sus requerimientos. Telnet consigue transferir a 32 Kbits/s, que es la velocidad de transferencia con la que fue configurado en el script.

| | **Voz_1** | **Voz_2** | **Video_1** | **Video_2** | **FTP** | **Telnet** |
|---|---|---|---|---|---|---|
| **Tipo Tráfico** | CBR/UDP | CBR/UDP | CBR/UDP | CBR/UDP | FTP/TCP | Telnet/TCP |
| **Origen** | 8 | 0 | 1 | 7 | 2 | 4 |
| **Destino** | 1 | 3 | 5 | 2 | 0 | 7 |
| **Tiempo Inicio (seg)** | 1 | 2 | 4 | 5 | 3 | 6 |
| **Paquetes Env.** | 2950 | 2900 | 10000 | 9822 | 9011 | 550 |
| **Paquetes Recib.** | 2950 | 2900 | 9999 | 9822 | 8991 | 550 |
| **Paquetes Perd.** | 0 | 0 | 1 | 0 | 20 | 0 |
| **Retardo (ms)** | 2.62798 | 3.21243 | 4.58352 | 3.5018 | 124.09 | 11.5454 |
| **Throughput (KB)** | 64.0187 | 64.0096 | 1999.7 | 2000.07 | 1262.13 | 32.7533 |
| **Jitter (ms)** | 2.03946 | 2.4624 | 2.06879 | 1.79068 | 4.14043 | 9.92655 |

Tabla 4.4 - Estadísticas simulación con QoS y conexiones de video a 2 Mbits/s

El siguiente gráfico corresponde a la misma simulación que la anterior pero sin las características de QoS aplicadas. Como se puede observar, las conexiones de video, especialmente Video_1, sufren una disminución en su rendimiento al incrementarse sustancialmente la cantidad de paquetes perdidos. Esto, junto con la falta de prioridades en los diferentes tráficos, provoca que tales conexiones no sean tan estables como cuando se aplica Calidad de Servicio, sino que presentan picos pronunciados en sus rendimientos. Las conexiones de voz tampoco tienen un rendimiento tan parejo como lo tienen en la simulación anterior, especialmente Voz_2. También pierde rendimiento la conexión FTP que no llega a 1 Mbits/s aunque la pérdida de paquetes disminuye. Solamente se pierden 3 paquetes contra los 20 de la prueba anterior. Pero, al aumentar la cantidad total de paquetes perdidos entre todos los tráficos a causa de las colisiones, el tiempo efectivo de transferencia disminuye y, por lo tanto, también lo hace la tasa de transferencia de los tráficos.

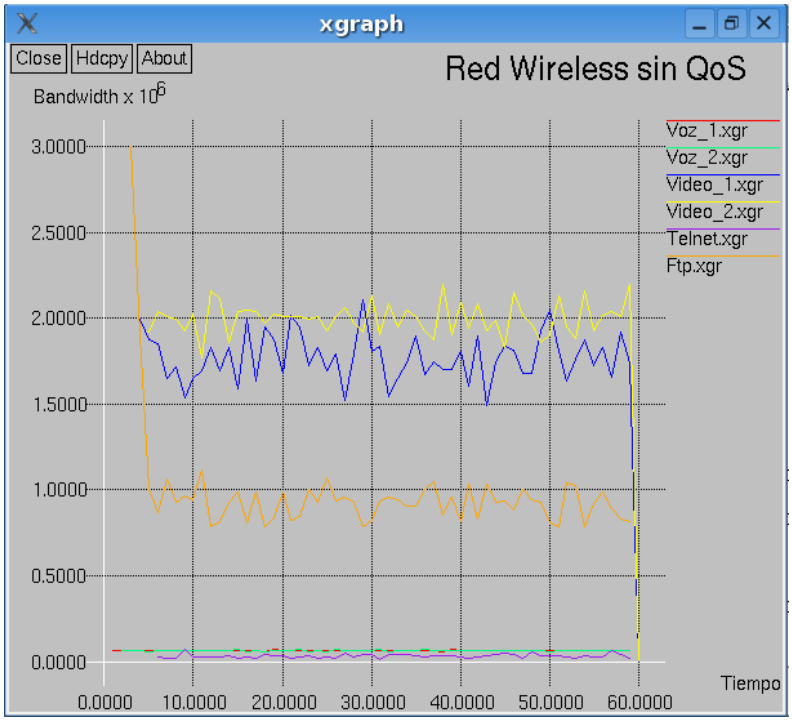



Figura 4.6 - Rendimiento sin QoS y conexiones de video a 2 Mbits/s

La tabla a continuación muestra los resultados del gráfico anterior. La conexión Voz_2 tiene una pérdida de 10 paquetes, que no llega al 0,35%, pero el problema mayor es el gran incremento en el Delay (Retardo). Sin embargo, es Video_1 el que sufre las peores consecuencias de no aplicar Calidad de Servicio al tener una pérdida de paquetes de más del 11%, mucho mayor al porcentaje permitido para una conexión de estas características. Sin dudas, esto decrementará de manera notable la calidad de imagen del video recibido.

| | **Voz_1** | **Voz_2** | **Video_1** | **Video_2** | **FTP** | **Telnet** |
|---|---|---|---|---|---|---|
| **Tipo Tráfico** | CBR/UDP | CBR/UDP | CBR/UDP | CBR/UDP | FTP/TCP | Telnet/TCP |
| **Origen** | 8 | 0 | 1 | 7 | 2 | 4 |
| **Destino** | 1 | 3 | 5 | 2 | 0 | 7 |
| **Tiempo Inicio (seg)** | 1 | 2 | 4 | 5 | 3 | 6 |
| **Paquetes Env.** | 2950 | 2900 | 10000 | 9822 | 6943 | 540 |
| **Paquetes Recib.** | 2950 | 2890 | 8854 | 9812 | 6940 | 540 |
| **Paquetes Perd.** | 0 | 10 | 1146 | 10 | 3 | 0 |
| **Retardo (ms)** | 9.24224 | 147.416 | 283.119 | 58.8353 | 22.804 | 8.72867 |
| **Throughput (KB)** | 64.0113 | 63.8018 | 1770.77 | 1997.98 | 974.192 | 32.2812 |
| **Jitter (ms)** | 6.16577 | 9.86254 | 3.90705 | 2.78449 | 5.39405 | 7.17217 |

Tabla 4.5 - Estadísticas simulación sin QoS y conexiones de video a 2 Mbits/s

A continuación, utilizando la misma topología de red, se ejecuta la simulación, pero ahora las conexiones de video generan tráfico a una velocidad de 2,5 Mbits/s con la intención de ir saturando la red y ver como se comportan los distintos tráficos cuando esto sucede.

En el siguiente gráfico podemos observar como se comporta la red ante esta situación cuando se aplica Calidad de Servicio. Los dos tráficos de video tienen un rendimiento muy cercano a los 2,5 Mbits/s pero es Video_2 el que tiene un mejor desempeño, especialmente porque su cadencia es más constante. Video_1 tiene un desempeño más inestable, especialmente entre los 25 y los 30 segundos, donde alcanza sus extremos, máximo y mínimo, en lo que a la velocidad de transferencia se refiere. El crecimiento del tráfico de video es a costa del tráfico de FTP que decae hasta los 755 Kbits/s de promedio contra los 1,262 Mbits/s que tenía en la simulación anterior.

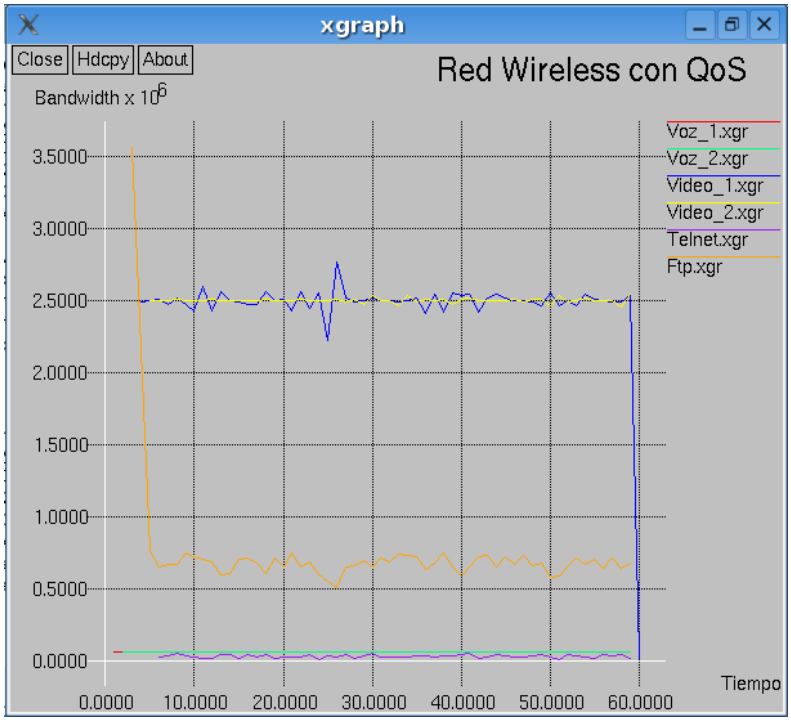



Figura 4.7 - Rendimiento con QoS y conexiones de video a 2,5 Mbits/s

La tabla correspondiente al gráfico anterior se muestra a continuación. Como se puede observar, las conexiones de voz obtienen los requerimientos necesarios para este tipo de conexiones. Además, ninguna de las dos sufren pérdida de paquetes. Las dos conexiones de video sólo pierden un paquete cada una. Sin embargo, Video_1 presenta una mayor inestabilidad que Video_2 aunque, el rendimiento promedio de ambas conexiones es similar (el rendimiento promedio de Video_2 es apenas superior al de Video_1). También la conexión Telnet se mantiene, en comparación a la simulación anterior mostrada en la figura 4.7. No sólo se mantiene sino que obtiene un pequeño incremento en su rendimiento. FTP es la conexión que más sufre en este nuevo esquema, aunque pierde un paquete menos que en la simulación anterior con Calida de Servicio aplicada, su rendimiento promedio es sensiblemente menor.

| | Voz_1 | Voz_2 | Video_1 | Video_2 | FTP | Telnet |
|---|---|---|---|---|---|---|
| **Tipo Tráfico** | CBR/UDP | CBR/UDP | CBR/UDP | CBR/UDP | FTP/TCP | Telnet/TCP |
| **Origen** | 8 | 0 | 1 | 7 | 2 | 4 |
| **Destino** | 1 | 3 | 5 | 2 | 0 | 7 |
| **Tiempo Inicio (seg.)** | 1 | 2 | 4 | 5 | 3 | 6 |
| **Paquetes Env.** | 2950 | 2900 | 12500 | 12277 | 5369 | 581 |
| **Paquetes Recib.** | 2950 | 2900 | 12499 | 12276 | 5350 | 581 |
| **Paquetes Perd.** | 0 | 0 | 1 | 1 | 19 | 0 |
| **Retardo (ms)** | 3.05361 | 3.90671 | 17.0129 | 6.16361 | 209.631 | 21.2554 |
| **Throughput (KB)** | 64.0188 | 64.0184 | 2499.68 | 2499.94 | 750.93 | 34.7745 |
| **Jitter (ms)** | 2.4797 | 3.19603 | 2.23818 | 1.90498 | 8.78763 | 17.7439 |

Tabla 4.6 - Estadísticas simulación con QoS y conexiones de video a 2.5 Mbits/s

La misma simulación se ejecutó sin las características de Calidad de Servicio aplicadas. El resultado se muestra en el siguiente gráfico. Lo primero que se puede observar es que ninguna de las dos conexiones de video alcanzan los 2,5 Mbits/s requeridos. Video_2 obtiene un mejor rendimiento que Video_1 pero los dos sufren una variación muy grande en el retardo debido, especialmente, a la gran pérdida de paquetes que sufren. FTP incrementa su rendimiento con respecto al que tenía en el gráfico anterior. Las demás conexiones se mantienen estables con respecto al gráfico anterior en cuanto al rendimiento pero las de voz sufren un incremento en el Delay, especialmente Voz_2. Por su parte, la conexión Telnet obtiene un rendimiento mayor con picos que superan los 64 Kbits/s.

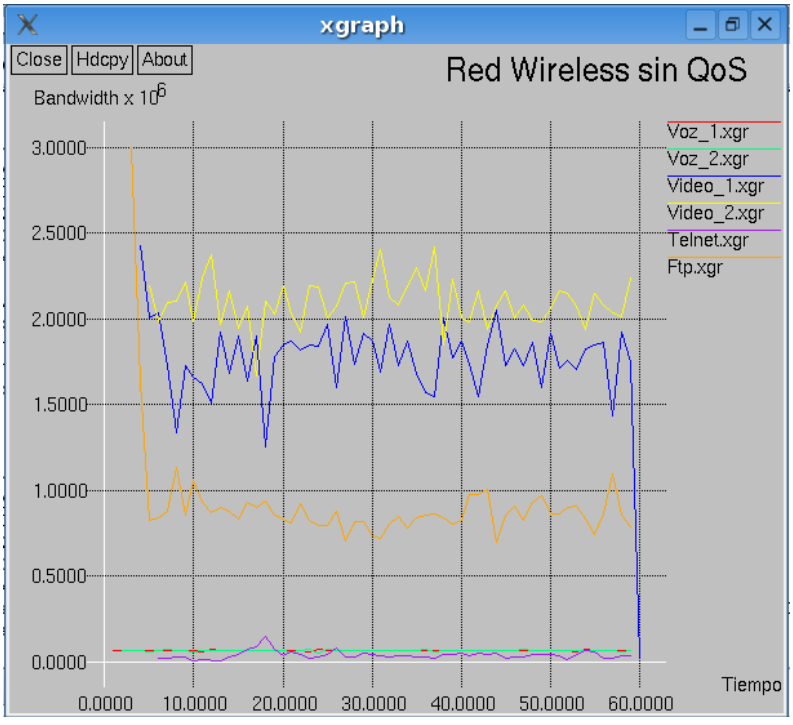



Figura 4.8 - Rendimiento sin QoS y conexiones de video a 2,5 Mbits/s

Ésta es la tabla correspondiente al gráfico anterior. En primer lugar, Voz_1, que alcanza el rendimiento deseado, sufre un aumento en su retardo y jitter pero no pierde ningún paquete. En cambio, Voz_2 sufre un retardo mucho mayor, pierde 9 paquetes y no logra alcanzar el rendimiento deseado. Lo que crece demasiado es la cantidad de paquetes que se pierden en las conexiones de video. Video_1 pierde el 28,69% de los paquetes y Video_2 el 16,02%, lo que provoca que las dos conexiones se encuentren muy lejos de cumplir con los requerimientos solicitados. Sin dudas, estas son pérdidas muy altas y especialmente para este tipo de tráfico. Por último, se puede observar como los tráficos de FTP y Telnet, este último en mucha menor medida, incrementan su rendimiento en detrimento de los otros tráficos que tienen mayores requerimientos.

| | **Voz_1** | **Voz_2** | **Video_1** | **Video_2** | **FTP** | **Telnet** |
|---|---|---|---|---|---|---|
| **Tipo Tráfico** | CBR/UDP | CBR/UDP | CBR/UDP | CBR/UDP | FTP/TCP | Telnet/TCP |
| **Origen** | 8 | 0 | 1 | 7 | 2 | 4 |
| **Destino** | 1 | 3 | 5 | 2 | 0 | 7 |
| **Tiempo Inicio (seg)** | 1 | 2 | 4 | 5 | 3 | 6 |
| **Paquetes Env.** | 2950 | 2900 | 12500 | 12277 | 6524 | 622 |
| **Paquetes Recib.** | 2950 | 2891 | 8913 | 10310 | 6522 | 622 |
| **Paquetes Perd.** | 0 | 9 | 3587 | 1967 | 2 | 0 |
| **Retardo (ms)** | 10.3086 | 161.65 | 300.698 | 242.683 | 20.9791 | 10.9267 |
| **Throughput (KB)** | 64.0142 | 63.8138 | 1782.53 | 2099.96 | 915.614 | 37.202 |
| **Jitter (ms)** | 6.62276 | 10.1681 | 4.36773 | 3.01469 | 5.67607 | 8.50771 |

Tabla 4.7 - Estadísticas simulación sin QoS y conexiones de video a 2.5 Mbits/s

Por último, se incrementó nuevamente el ancho de banda requerido por las conexiones de video, pasando de los 2,5 Mbits/s de la simulación anterior a los 4 Mbits/s. Esto satura la red completamente.

Como se puede observar en el siguiente gráfico ninguna de las dos conexiones de video puede obtener el rendimiento que desea, especialmente Video_1 que no puede superar los 2,5 Mbits/s. En cambio, Video_2 supera los 3 Mbits/s pero, al igual que Video_1, se encuentra muy lejos de los 4 Mbits/s requeridos. También podemos observar que las conexiones de voz siguen funcionando correctamente y, como debía suceder por tener una menor prioridad, la conexión FTP, transmitiendo en un promedio de 500 Kbits/s, es la que, nuevamente, tiene una baja en su rendimiento.

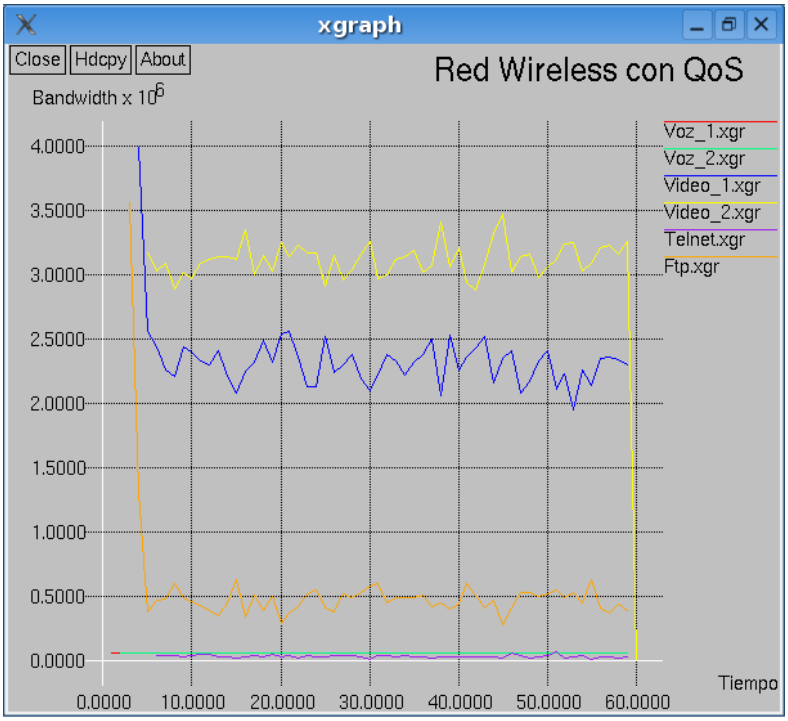



Figura 4.9 - Rendimiento con QoS y conexiones de video a 4 Mbits/s

Ésta es la tabla correspondiente al gráfico anterior. Como se puede observar en las conexiones de video, la pérdida de paquetes, especialmente en Video_1 que es del 41,58 % (en Video_2 es del 22,67 %), se incrementa notablemente deteriorando totalmente la calidad del video. Estas pérdidas provocan que el retardo promedio se incremente notablemente. Las demás conexiones no sufren en lo que a la pérdida de paquetes se refiere, pero la conexión FTP sufre una baja considerable en su rendimiento.

| | **Voz_1** | **Voz_2** | **Video_1** | **Video_2** | **FTP** | **Telnet** |
|---|---|---|---|---|---|---|
| **Tipo Tráfico** | CBR/UDP | CBR/UDP | CBR/UDP | CBR/UDP | FTP/TCP | Telnet/TCP |
| **Origen** | 8 | 0 | 1 | 7 | 2 | 4 |
| **Destino** | 1 | 3 | 5 | 2 | 0 | 7 |
| **Tiempo Inicio (seg.)** | 1 | 2 | 4 | 5 | 3 | 6 |
| **Paquetes Env.** | 2950 | 2900 | 20000 | 19643 | 3852 | 585 |
| **Paquetes Recib.** | 2950 | 2900 | 11684 | 15307 | 3832 | 584 |
| **Paquetes Perd.** | 0 | 0 | 8316 | 4336 | 20 | 1 |
| **Delay (ms)** | 3.39491 | 4.41414 | 231.808 | 175.096 | 292.77 | 38.7022 |
| **Throughput (KB)** | 64.0206 | 64.0153 | 2336.8 | 3116.98 | 538.102 | 34.8241 |
| **Jitter (ms)** | 2.71585 | 3.56992 | 3.59683 | 2.11357 | 14.0447 | 24.3424 |

Tabla 4.8 - Estadísticas simulación con QoS y conexiones de video a 4 Mbits/s

Obviamente, esta misma simulación se realizó sin QoS, y en este caso, los resultados obtenidos nos permiten observar como decrece el rendimiento de las distintas conexiones. El rendimiento de las conexiones de video es extremadamente malo. El de Video_1 es de 1,782 Mbits/s con una pérdida de paquetes del 55% y el de Video_2 es un poco mejor al estar en los 2,228

Mbits/s y con un 44% de pérdida de paquetes. La conexión que sí se ve favorecida, comparándola con lo que obtiene en la simulación anterior, es FTP que obtiene un rendimiento de 920 Kbits/s. Las dos conexiones de voz mantienen su rendimiento aunque Voz_2 tiene una pérdida mayor de paquetes. Se puede observar, que en torno a los 38 seg. de la simulación, hay un decrecimiento apreciable en su rendimiento. Pero, es la conexión telnet la que se lleva la peor parte al no poder intercambiar ningún paquete en toda la simulación. Al observar detenidamente el archivo de salida de la simulación, el archivo con extensión tr, se puede ver que en ningún momento se logra establecer la conexión TCP.

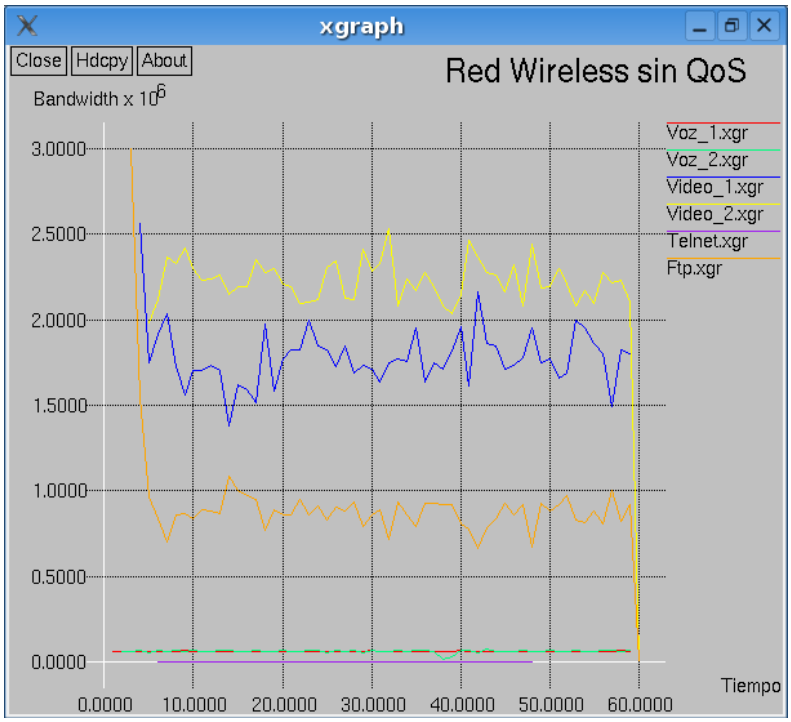



Figura 4.10 - Rendimiento sin QoS y conexiones de video a 2 Mbits/s

En la siguiente tabla se muestran detalladamente los resultados del gráfico anterior. Sin dudas, las conexiones de video sufren un notable deterioro con pérdida de paquetes superiores al 50%. Voz_2 sufre la mayor pérdida de paquetes de todas las simulaciones. Esto provocará que en el receptor se sufran algunas distorsiones y, por lo tanto, el audio se oirá entrecortado. En cambio, Voz_1 no pierde ningún paquete y obtiene el ancho de banda necesario, pero sufre un mayor retardo y jitter. Por su parte, FTP mantiene un buen rendimiento, no se ve afectado demasiado en esta simulación y solamente pierde dos paquetes. En cambio, Telnet no logra enviar ningún paquete. Esto último es un caso de inanición en donde una conexión no puede enviar ningún paquete porque existen otras conexiones que mantiene el medio constantemente ocupado.

Aunque al aplicar Calidad de Servicio a la red no se presentó el problema de la inanición esto podría sucederle a las colas de menor prioridad.

Una cola de mayor prioridad podría tener siempre paquetes para procesar lo que haría que las colas de menor prioridad nunca puedan enviar sus paquetes.

| | **Voz_1** | **Voz_2** | **Video_1** | **Video_2** | **FTP** | **Telnet** |
|---|---|---|---|---|---|---|
| **Tipo Tráfico** | CBR/UDP | CBR/UDP | CBR/UDP | CBR/UDP | FTP/TCP | Telnet/TCP |
| **Origen** | 8 | 0 | 1 | 7 | 2 | 4 |
| **Destino** | 1 | 3 | 5 | 2 | 0 | 7 |
| **Tiempo Inicio (seg.)** | 1 | 2 | 4 | 5 | 3 | 6 |
| **Paquetes Env.** | 2950 | 2900 | 20000 | 19643 | 6562 | 0 |
| **Paquetes Recib.** | 2950 | 2836 | 8913 | 10941 | 6560 | 0 |
| **Paquetes Perd.** | 0 | 64 | 11087 | 8702 | 2 | 0 |
| **Retardo (ms)** | 8.72373 | 164.796 | 310.149 | 247.089 | 19.8039 | 0 |
| **Throughput (KB)** | 64.0211 | 62.593 | 1782.65 | 2227.92 | 920.758 | 0 |
| **Jitter (ms)** | 6.30507 | 10.2613 | 4.87324 | 3.2758 | 5.40588 | 0 |

Tabla 4.9 - Estadísticas simulación sin QoS y conexiones de video a 4 Mbits/s

# Conclusiones

Con todas las pruebas finalizadas y analizadas es posible obtener ciertas conclusiones. En primer lugar, es importante destacar que las pruebas fueron realizadas utilizando un simulador y no en un ambiente real. Aunque una simulación no refleje exactamente la realidad, la aceptación del uso de los simuladores, y especialmente el del NS-2, se ha extendido dentro del ámbito académico.

No existen dudas con respecto a que los diferentes tipos de tráficos deben recibir tratamiento diferenciado por parte de la red. Sí, a esto le sumamos la terrible competencia por el acceso al medio y las consecuentes colisiones, no quedan dudas de la necesidad de brindar Calidad de Servicio. Los resultados de este trabajo sirven para corroborarlo. Al aumentar la carga de la red, los tráficos que tienen prioridades mayores pueden seguir obteniendo los requerimientos solicitados mientras que los de menor prioridad ven decrementado su rendimiento.

En primer lugar es conveniente destacar los motivos por los cuales se utiliza el campo Flow Label, por parte de los hosts, para solicitar una determinada Calidad de Servicio. Al estar ubicado en la cabecera de IPv6 es más simple y rápido para los routers determinar el tratamiento que debe recibir un paquete en particular. No hay necesidad de ir a buscar a la cabecera TCP de cada paquete para ver los puertos y poder determinar la Calidad de Servicio que hay que darle.

Tampoco importa cual es la tecnología de capa 2 que están atravesando los paquetes, los parámetros que se utilizan para indicar la Calidad de Servicio solicitada son constantes a lo largo de todo el camino entre el origen y el destino al encontrarse en la cabecera del protocolo de capa de red, por lo que, siempre se utilizan los mismos valores para determinar cuál es la Calidad de Servicio que se le debe brindar a cada paquete.

Aunque la utilización del protocolo IPv6 no ha tenido un crecimiento tan exponencial como si lo han tenido las redes inalámbricas, sino que ha sido más paulatino, no quedan dudas de que su uso será, con el tiempo, inevitable. Es por esto, que es conveniente que la integración entre ellas sea la mayor posible. Además, las herramientas de Calidad de Servicio, sin dudas, tendrán un papel cada vez más importante debido a la diversificación de las aplicaciones que hacen uso de las redes de datos. Y, este trabajo, sirve para confirmarlo.

A lo largo de esta tesis se hicieron varias pruebas con el fin de comprobar si, una red wireless, realmente mejora su funcionamiento al aplicarle técnicas de Calidad de Servicio usando el campo Flow Label, ubicado en la cabecera IPv6, y los resultados obtenidos hacen posible asegurar que el rendimiento de la red es muy superior cuando se las aplica a cuando no se lo hace.

En la primera prueba, tráfico de video a 2 Mbits/s, todas las conexiones de voz y video obtienen el rendimiento esperado al aplicar Calidad de Servicio. Esto no sucede cuando tales técnicas no son aplicadas. No sólo las conexiones de voz y video funcionan mejor, también lo hace la de FTP, transfiriendo, en promedio, 260 Kbits más por segundo.

Al elevar las conexiones de video a 2,5 Mbits/s, éstas aún pueden satisfacer sus necesidades si se aplican las herramientas de Calidad de Servicio. También sucede lo mismo con las conexiones de voz Por el contrario, la que pierde rendimiento es FTP. Esto es correcto que suceda porque es la conexión con menor prioridad. En cambio, si no se aplican tales herramientas, sólo Voz_1 alcanza el rendimiento solicitado y FTP, a diferencia del caso anterior, incrementa su consumo de ancho de banda perjudicando a las conexiones de mayor prioridad.

El último caso, con tráfico de video a 4 Mbits/s, la red se satura totalmente y, es lógico, que ninguna de las dos conexiones de video obtengan todo el ancho de banda necesario porque el tráfico que generan es mayor a la capacidad total de la red. Sin embargo, y a pesar de estar saturada la red, las conexiones de voz, que son las de mayor prioridad, sí consiguen cumplir con su requerimientos cuando se aplica Calidad de Servicio. Si esto no se hace, el funcionamiento de la red es totalmente pésimo. A tal punto, que la conexión TCP sobre la que debería correr Telnet no llega nunca a establecerse.

En síntesis, además de permitirle a las distintas conexiones cumplir con sus requerimientos y, de esta forma, obtener el rendimiento esperado por los usuarios finales, también mejora el funcionamiento general de la red. No quedan dudas de los beneficios que brinda la utilización de las herramientas de Calidad de Servicio.

Por último, queda detallar cuales son los trabajos futuros. En primer lugar, es importante implementar lo propuesto en este trabajo en un ambiente real para obtener datos más certeros de los que se han obtenido utilizando el simulador. Por ejemplo, se podría modificar la pila IPv6 y demás componente de un sistema operativo para realizar las pruebas.

Otro trabajo futuro es una modificación al protocolo IPv6 para permitirle a las aplicaciones receptoras indicarles a las aplicaciones emisoras que necesitan un tratamiento diferente al que están recibiendo hasta el momento, por ejemplo, un menor delay o jitter. Con lo cual, los emisores pueden modificar el o los correspondientes subcampos, de los 4 en que se dividió en campo Flow Label, y modificar a que cola, de las 4 que define el estándar 802.11e, se asignan los paquetes. Los parámetros se irían ajustando dinámicamente a medida que se realiza la transmisión.

Otro posible trabajo es mejorar el funcionamiento del estándar 802.11 para evitar problemas de inanición. Ante una red muy saturada, este problema también se podría presentar en el estándar 802.11e. Las colas de mayor prioridad podrían desplazar constantemente a las de menor prioridad, al no permitirles transmitir ningún paquete.

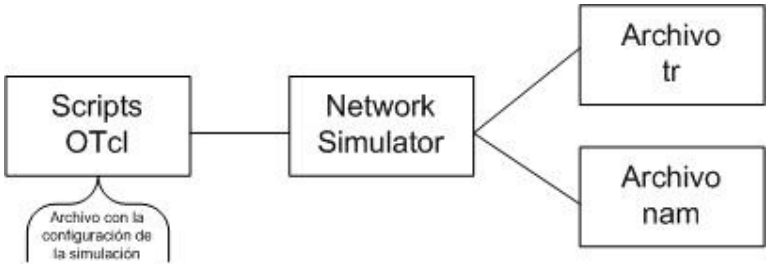
# Apéndice 1

# El simulador NS-2

Network Simulator 2, conocido como NS-2, es un simulador de redes orientado a eventos. Fue desarrollado como parte del proyecto VINT (Virtual Internet Testbed), que consistía en una colaboración de varios institutos y organizaciones entre las que se incluían la Universidad de Berkeley, AT&T, Xeros PARC y ETH. La primera versión del simulador estuvo disponible en el año 1995 y, para 1996, ya estaba disponible la segunda versión.

El NS-2 está disponible para varias plataformas y ofrece soporte para todo tipo de redes cableadas e inalámbricas. Permite la configuración de gran cantidad de parámetros tales como la topología de la red, la pila de protocolos y parámetros específicos de cada protocolo. También permite evaluar el impacto de diferentes tipos de tráfico de red. Al ser un producto de código abierto está en constante evolución.

El simulador NS consta de un núcleo principal, escrito en C++, al que se invoca simplemente tecleando ns en la línea de comandos (una vez este ha sido correctamente instalado). A partir de este punto el usuario puede interactuar directamente con el simulador, a través de un lenguaje de interface. El lenguaje de interface es OTcl, una versión de Tcl 2 orientada a objetos. En realidad, el método anterior no es el más recomendable, NS está pensado para ser ejecutado en modo *batch*. El usuario define mediante un *script* la pila de protocolos y otros aspectos básicos del tipo de red a simular y proporciona al ns diversos datos acerca del escenario a simular y del tipo y características de los tráficos que se va a generar. Conforme avanza la simulación, se generan un conjunto de datos de salida que se almacenan en un fichero de traza. A partir de las trazas de simulación, se pueden utilizar lenguajes como **Perl** y **AWK** para filtrar la traza y obtener los índices de prestaciones que se deseen evaluar. Finalmente, herramientas tales como Network Animator (**nam**) permiten realizar un análisis visual del envío y recepción de paquetes de datos y control a medida que avanza la simulación. La siguiente figura muestra un esquema del proceso de simulación




El simulador se puede ejecutar tanto en entornos Linux como Windows. Para realizar esta tesis instalé el simulador, la versión 2.28, en un Linux Debian

3.1 ejecutándose en una maquina virtual VMWare. El NS-2 se puede obtener desde el sitio web: http://www.isi.edu/nsnam/ns/.

Para hacer funcionar el simulador en forma completa se necesita que estén instalados varios componentes obligatorios, como el oTcl, Tcl/Tk, y otros que son opcionales, como Nam, Perl, etc. Todos estos componentes se pueden descargar desde la página web del simulador, o desde sus correspondientes sitios, y se compilan e instalan independientemente. Existe otra posibilidad y es la de bajar un único archivo que contiene todos los componentes juntos y se instalan en un solo paso. Esta última opción es la que utilicé para instalar el producto.

Desde la página web bajé un archivo que se llama ns-allinone-2.28.tar.gz. Su tamaño es de aproximadamente 56 megabytes. Mediante el siguiente comando se descomprime el archivo:

```
tar –xzvf  ns-allinone-2.28.tar.gz
```

Cuando se termina de descomprimir el archivo se crea un directorio ns-2.28. Para terminar de instalar el simulador se debe ingresar a este directorio y ejecutar el comando:

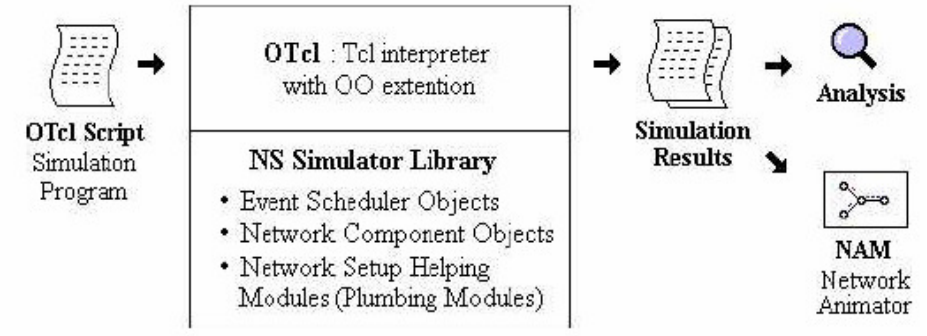
```
./install
```

Al finalizar la instalación el simulador está listo para ser utilizado. Si más adelante se hace alguna modificación al código fuente del NS se deben ejecutar los siguientes comandos:

```
./configure
./make
```

Para realizar las pruebas se utilizó el simulador desarrollado en la Universidad de Berkeley (NS, Network Simulator). NS es un simulador discreto orientado a eventos para modelar distintos tipos de redes de código abierto, el cual está desarrollado en C++ y Otcl (lenguaje de scripts orientado a objetos desarrollado por el MIT). El usuario escribe un script en Otcl que describe lo que se está intentado simular. Este script es procesado por el simulador. Los resultados de la simulación quedan en dos archivos que contienen prácticamente la misma información pero en distinto formato. Un archivo de texto con los resultados de la simulación bastante complicado de entender pero que mediante el uso de distintos scripts, escritos en Awk, Perl, etc., es posible obtener los resultados deseados que pueden ser visualizados gráficamente utilizando herramientas como el gnuplot o xgraph. La extensión del otro archivo es .nam y puede ser visualizado con la herramienta NAM (Network Animador) que permite ver en forma gráfica todo lo que sucede en la simulación.

A continuación se muestra una vista muy simplificada de lo que es el funcionamiento del simulador:



# Apéndice 2

# Scripts

En este apartado se muestran todos los scripts utilizados en las diferentes simulaciones.

A continuación se muestra el script RedWireless_2.1 utilizado para mostrar el funcionamiento de una red wireless 802.11.

```
#
# globals and flags
#
set ns [new Simulator]
set AKAROA 0

#
#number of nodes
#
set num_mobile_nodes 6

# Parameter for wireless nodes
#
set opt(chan)           Channel/WirelessChannel    ;# channel type
set opt(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set opt(netif)          Phy/WirelessPhy            ;# network interface type
set opt(mac)            Mac/802_11                 ;# MAC type
set opt(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set opt(ll)             LL                         ;# link layer type
set opt(ant)            Antenna/OmniAntenna        ;# antenna model
set opt(ifqlen)         50                         ;# max packet in ifq
set opt(adhocRouting)   DumbAgent                  ;# routing protocol

set opt(x)        670   ;# X dimension of the topography
set opt(y)        670   ;# Y dimension of the topography

Phy/WirelessPhy set RXThresh_ 3.65263e-10
Phy/WirelessPhy set CSThresh_ 3.65263e-10

$ns color 1 Red
$ns color 2 Blue
$ns color 3 Green
$ns color 4 Black
$ns color 4 Violet

#Open the nam trace file
#
set nf [open out.nam w]
$ns namtrace-all-wireless $nf $opt(x) $opt(y)
set ntr [open out.tr w]
$ns trace-all $ntr

set chan    [new $opt(chan)]
set topo    [new Topography]
$topo load_flatgrid $opt(x) $opt(y)

# Create God
```

```
create-god $num_mobile_nodes

# creating base station

$ns node-config -adhocRouting $opt(adhocRouting) \
                -llType $opt(ll) \
                -macType $opt(mac) \
                -ifqType $opt(ifq) \
                -ifqLen $opt(ifqlen) \
                -antType $opt(ant) \
                -propType $opt(prop)    \
                -phyType $opt(netif) \
                -channel $chan      \
                -topoInstance $topo \
                -wiredRouting ON \
                -agentTrace ON \
                -routerTrace OFF \
                -macTrace OFF    \
                -movementTrace OFF

# creating mobile nodes
#
for {set i 0} {$i < $num_mobile_nodes} {incr i} {
 set wl_node_($i) [$ns node]
 $wl_node_($i) random-motion 0           ;# disable random motion
 puts "wireless node $i created ..."
}

$wl_node_(0) set X_ 5.0
$wl_node_(0) set Y_ 5.0
$wl_node_(0) set Z_ 0.0

$wl_node_(1) set X_ 100.0
$wl_node_(1) set Y_ 26.0
$wl_node_(1) set Z_ 0.0

$wl_node_(2) set X_ 90.0
$wl_node_(2) set Y_ 64.0
$wl_node_(2) set Z_ 0.0

$wl_node_(3) set X_ 10.0
$wl_node_(3) set Y_ 50.0
$wl_node_(3) set Z_ 0.0

$wl_node_(4) set X_ 55.0
$wl_node_(4) set Y_ 7.0
$wl_node_(4) set Z_ 0.0

$wl_node_(5) set X_ 48.0
$wl_node_(5) set Y_ 97.0
$wl_node_(5) set Z_ 0.0

######## Tráficos generados ##########

set src_udp0 [new Agent/UDP]
$src_udp0 set class_ 0
$src_udp0 set fid_ 0
set dst_udp0 [new Agent/Null]
$ns attach-agent $wl_node_(4) $src_udp0
$ns attach-agent $wl_node_(1) $dst_udp0
set app [new Application/Traffic/CBR]
```

```
$app attach-agent $src_udp0
$ns connect $src_udp0 $dst_udp0
$ns at 0.0 "$app start"

set src_udp1 [new Agent/UDP]
$src_udp1 set class_ 1
$src_udp1 set fid_ 1
set dst_udp1 [new Agent/Null]
$ns attach-agent $wl_node_(1) $src_udp1
$ns attach-agent $wl_node_(3) $dst_udp1
set app1 [new Application/Traffic/CBR]
$app1 attach-agent $src_udp1
$ns connect $src_udp1 $dst_udp1
$ns at 20.0 "$app1 start"

set src_udp2 [new Agent/UDP]
$src_udp2 set class_ 2
$src_udp2 set fid_ 2
set dst_udp2 [new Agent/Null]
$ns attach-agent $wl_node_(2) $src_udp2
$ns attach-agent $wl_node_(4) $dst_udp2
set app2 [new Application/Traffic/CBR]
$app2 attach-agent $src_udp2
$ns connect $src_udp2 $dst_udp2
$ns at 10.0 "$app2 start"

set src_udp3 [new Agent/UDP]
$src_udp3 set class_ 3
$src_udp3 set fid_ 3
set dst_udp3 [new Agent/Null]
$ns attach-agent $wl_node_(3) $src_udp3
$ns attach-agent $wl_node_(0) $dst_udp3
set app3 [new Application/Traffic/CBR]
$app3 attach-agent $src_udp3
$ns connect $src_udp3 $dst_udp3
$ns at 40.0 "$app3 start"

set src_udp4 [new Agent/UDP]
$src_udp4 set class_ 4
$src_udp4 set fid_ 4
set dst_udp4 [new Agent/Null]
$ns attach-agent $wl_node_(5) $src_udp4
$ns attach-agent $wl_node_(2) $dst_udp4
set app4 [new Application/Traffic/CBR]
$app4 attach-agent $src_udp4
$ns connect $src_udp4 $dst_udp4
$ns at 30.0 "$app4 start"

# Define node initial position in nam
for {set i 0} {$i < $num_mobile_nodes} {incr i} {
 $ns initial_node_pos $wl_node_($i) 30
}

# Tell nodes when the simulation ends
#
for {set i 0} {$i < $num_mobile_nodes } {incr i} {
 $ns at 60.0 "$wl_node_($i) reset";
}

if {$AKAROA == 0} {
    $ns at 60.001 "$app stop"
```

```
    $ns at 60.001 "$app1 stop"
    $ns at 60.001 "$app2 stop"
    $ns at 60.001 "$app3 stop"
    $ns at 60.002 "puts \"NS EXITING...\" ; $ns halt"
}

proc stop {} {
    global ns ntr nf
    close $ntr
    close $nf
}

# run the simulation
$ns run
```

Este script, llamado RedWireless_2.2, es el mismo que el anterior pero ahora con las propiedades de Calidad de Servicio aplicadas.

```
#
# globals and flags
#
set ns [new Simulator]
set AKAROA 0

#number of nodes
#
set num_mobile_nodes 6

#
# Parameter for wireless nodes
#
set opt(chan)           Channel/WirelessChannel    ;# channel type
set opt(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set opt(netif)          Phy/WirelessPhy            ;# network interface type
set opt(mac)            Mac/802_11e                ;# MAC type
set opt(ifq)            Queue/DTail/PriQ           ;# interface queue type
set opt(ll)             LL                         ;# link layer type
set opt(ant)            Antenna/OmniAntenna        ;# antenna model
set opt(ifqlen)         50                         ;# max packet in ifq
set opt(adhocRouting)   DumbAgent                  ;# routing protocol

set opt(x)        670   ;# X dimension of the topography
set opt(y)        670   ;# Y dimension of the topography

Phy/WirelessPhy set RXThresh_ 3.65263e-10
Phy/WirelessPhy set CSThresh_ 3.65263e-10

$ns color 1 Red
$ns color 2 Blue
$ns color 3 Green
$ns color 4 Black
$ns color 4 Violet

#Open the nam trace file
#
set nf [open bwout.nam w]
$ns namtrace-all-wireless $nf $opt(x) $opt(y)
set ntr [open bwout.tr w]
$ns trace-all $ntr

set chan    [new $opt(chan)]
```

```
set topo    [new Topography]
$topo load_flatgrid $opt(x) $opt(y)

# Create God
create-god $num_mobile_nodes

# creating base station
#

$ns node-config -adhocRouting $opt(adhocRouting) \
                -llType $opt(ll) \
                -macType $opt(mac) \
                -ifqType $opt(ifq) \
                -ifqLen $opt(ifqlen) \
                -antType $opt(ant) \
                -propType $opt(prop)    \
                -phyType $opt(netif) \
                -channel $chan      \
                -topoInstance $topo \
                -wiredRouting ON \
                -agentTrace ON \
                -routerTrace OFF \
                -macTrace OFF    \
                -movementTrace OFF

# creating mobile nodes
for {set i 0} {$i < $num_mobile_nodes} {incr i} {
 set wl_node_($i) [$ns node]
 $wl_node_($i) random-motion 0           ;# disable random motion
 puts "wireless node $i created ..."
}

$wl_node_(0) set X_ 5.0
$wl_node_(0) set Y_ 5.0
$wl_node_(0) set Z_ 0.0

$wl_node_(1) set X_ 100.0
$wl_node_(1) set Y_ 26.0
$wl_node_(1) set Z_ 0.0

$wl_node_(2) set X_ 90.0
$wl_node_(2) set Y_ 64.0
$wl_node_(2) set Z_ 0.0

$wl_node_(3) set X_ 10.0
$wl_node_(3) set Y_ 50.0
$wl_node_(3) set Z_ 0.0

$wl_node_(4) set X_ 55.0
$wl_node_(4) set Y_ 7.0
$wl_node_(4) set Z_ 0.0

$wl_node_(5) set X_ 48.0
$wl_node_(5) set Y_ 97.0
$wl_node_(5) set Z_ 0.0

set src_udp0 [new Agent/UDP]
$src_udp0 set class_ 0
$src_udp0 set prio_ 3
$src_udp0 set fid_ 0
set dst_udp0 [new Agent/Null]
```

```
$ns attach-agent $wl_node_(4) $src_udp0
$ns attach-agent $wl_node_(1) $dst_udp0
set app [new Application/Traffic/CBR]
$app attach-agent $src_udp0
$ns connect $src_udp0 $dst_udp0
$ns at 0.0 "$app start"

set src_udp1 [new Agent/UDP]
$src_udp1 set class_ 1
$src_udp1 set prio_ 2
$src_udp1 set fid_ 1
set dst_udp1 [new Agent/Null]
$ns attach-agent $wl_node_(1) $src_udp1
$ns attach-agent $wl_node_(3) $dst_udp1
set app1 [new Application/Traffic/CBR]
$app1 attach-agent $src_udp1
$ns connect $src_udp1 $dst_udp1
$ns at 20.0 "$app1 start"

set src_udp2 [new Agent/UDP]
$src_udp2 set class_ 2
$src_udp2 set prio_ 0
$src_udp2 set fid_ 2
set dst_udp2 [new Agent/Null]
$ns attach-agent $wl_node_(2) $src_udp2
$ns attach-agent $wl_node_(4) $dst_udp2
set app2 [new Application/Traffic/CBR]
$app2 attach-agent $src_udp2
$ns connect $src_udp2 $dst_udp2
$ns at 10.0 "$app2 start"

set src_udp3 [new Agent/UDP]
$src_udp3 set class_ 3
$src_udp3 set prio_ 2
$src_udp3 set fid_ 3
set dst_udp3 [new Agent/Null]
$ns attach-agent $wl_node_(3) $src_udp3
$ns attach-agent $wl_node_(0) $dst_udp3
set app3 [new Application/Traffic/CBR]
$app3 attach-agent $src_udp3
$ns connect $src_udp3 $dst_udp3
$ns at 40.0 "$app3 start"

set src_udp4 [new Agent/UDP]
$src_udp4 set class_ 4
$src_udp4 set prio_ 1
$src_udp4 set fid_ 4
set dst_udp4 [new Agent/Null]
$ns attach-agent $wl_node_(5) $src_udp4
$ns attach-agent $wl_node_(2) $dst_udp4
set app4 [new Application/Traffic/CBR]
$app4 attach-agent $src_udp4
$ns connect $src_udp4 $dst_udp4
$ns at 30.0 "$app4 start"

# Define node initial position in nam
for {set i 0} {$i < $num_mobile_nodes} {incr i} {
 $ns initial_node_pos $wl_node_($i) 30
}

# Tell nodes when the simulation ends
```

```
#
for {set i 0} {$i < $num_mobile_nodes } {incr i} {
 $ns at 60.0 "$wl_node_($i) reset";
}

if {$AKAROA == 0} {
    $ns at 60.001 "$app stop"
    $ns at 60.001 "$app1 stop"
    $ns at 60.001 "$app2 stop"
    $ns at 60.001 "$app3 stop"
    $ns at 60.002 "puts \"NS EXITING...\" ; $ns halt"
}

proc stop {} {
    global ns ntr nf
    close $ntr
    close $nf
}

# run the simulation
$ns run
```

A continuación se muestra el script utilizado, en todas las demás simulaciones, en lo que resta de la tesis. Éste se utiliza para generar tráfico sin Calidad de Servicio con conexiones de video a 2 Mbits/seg.

```
# ======================================================================
# Define options
#
# ======================================================================
set opt(chan)         Channel/WirelessChannel   ;# channel type
set opt(prop)         Propagation/TwoRayGround  ;# radio-propagation model
set opt(netif)        Phy/WirelessPhy           ;# network interface type
set opt(mac)          Mac/802_11                ;# MAC type
set opt(ifq)          Queue/DropTail/PriQueue   ;# interface queue type
set opt(ll)           LL                        ;# link layer type
set opt(ant)          Antenna/OmniAntenna       ;# antenna model
set opt(ifqlen)       50                        ;# max packet in ifq
set opt(nn)           9                         ;# number of mobilenodes
set opt(adhocRouting) DSDV                      ;# routing protocol
set opt(x)            15                        ;# x coordinate of topology
set opt(y)            15                        ;# y coordinate of topology

Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 2.0e-14
Phy/WirelessPhy set RXThresh_ 1.77827941e-13
Phy/WirelessPhy set bandwidth_ 11Mb
Phy/WirelessPhy set Pt_ 0.031622777
Phy/WirelessPhy set freq_ 2.472e9
Phy/WirelessPhy set L_ 1

###############Parámetros de 802.11#########################
Mac/802_11 set CWMin_ 31
Mac/802_11 set CWMax_ 1023
Mac/802_11 set SlotTime_ 0.000020
Mac/802_11 set SIFS_ 0.000010
Mac/802_11 set PreambleLength_ 144
Mac/802_11 set ShortPreambleLength_ 72
Mac/802_11 set PreambleDataRate_ 1.0e6
Mac/802_11 set PLCPHeaderLength_ 48
```

```
Mac/802_11 set PLCPDataRate_ 1.0e6
Mac/802_11 set ShortPLCPDataRate_ 2.0e6
Mac/802_11 set RTSThreshold_ 3000
Mac/802_11 set ShortRetryLimit_ 7
Mac/802_11 set LongRetryLimit_ 4
Mac/802_11 set dataRate_ 11.0e6
Mac/802_11 set basicRate_ 1.0e6
############################################################

Agent/TCP set packetSize_ 1400
Agent/UDP set packetSize_ 1400

# create simulator instance
set ns_   [new Simulator]

set tracefd  [open wirelessinqos.tr w]
set namtrace [open wirelessinqos.nam w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

# Create topography object
set topo   [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

# create God
create-god $opt(nn)

# configure for base-station node
$ns_ node-config  -adhocRouting $opt(adhocRouting) \
                  -llType $opt(ll) \
                  -macType $opt(mac) \
                  -ifqType $opt(ifq) \
                  -ifqLen $opt(ifqlen) \
                  -antType $opt(ant) \
                  -propType $opt(prop) \
                  -phyType $opt(netif) \
                  -channelType $opt(chan) \
                  -topoInstance $topo \
                  -wiredRouting ON \
                  -agentTrace ON \
                  -routerTrace OFF \
                  -macTrace OFF


for {set i 0} {$i < $opt(nn)} {incr i} {
 set wl_node_($i) [$ns_ node]
 $wl_node_($i) random-motion 0
}

$wl_node_(0) set X_ 0.0
$wl_node_(0) set Y_ 1.0
$wl_node_(0) set Z_ 0.0

$wl_node_(1) set X_ 0.0
$wl_node_(1) set Y_ 6.0
$wl_node_(1) set Z_ 0.0

$wl_node_(2) set X_ 9.0
$wl_node_(2) set Y_ 4.0
```

```
$wl_node_(2) set Z_ 0.0

$wl_node_(3) set X_ 10.0
$wl_node_(3) set Y_ 5.0
$wl_node_(3) set Z_ 0.0

$wl_node_(4) set X_ 5.0
$wl_node_(4) set Y_ 7.0
$wl_node_(4) set Z_ 0.0

$wl_node_(5) set X_ 5.0
$wl_node_(5) set Y_ 12.0
$wl_node_(5) set Z_ 0.0

$wl_node_(6) set X_ 10.0
$wl_node_(6) set Y_ 2.0
$wl_node_(6) set Z_ 0.0

$wl_node_(7) set X_ 9.0
$wl_node_(7) set Y_ 6.0
$wl_node_(7) set Z_ 0.0

$wl_node_(8) set X_ 2.0
$wl_node_(8) set Y_ 5.0
$wl_node_(8) set Z_ 0.0

######################### Conexiones de Voz ##################

set src_udp0 [new Agent/UDP]
$src_udp0 set class_ 2
$src_udp0 set fid_ 0
set dst_udp0 [new Agent/Null]
$ns_ attach-agent $wl_node_(8) $src_udp0
$ns_ attach-agent $wl_node_(1) $dst_udp0
$ns_ connect $src_udp0 $dst_udp0
set app0 [new Application/Traffic/CBR]
$app0 set packetSize_ 160
$app0 set interval_ 0.020
$app0 attach-agent $src_udp0
$ns_ at 1.0 "$app0 start"

set src_udp1 [new Agent/UDP]
$src_udp1 set class_ 2
$src_udp1 set fid_ 1
set dst_udp1 [new Agent/Null]
$ns_ attach-agent $wl_node_(0) $src_udp1
$ns_ attach-agent $wl_node_(3) $dst_udp1
$ns_ connect $src_udp1 $dst_udp1
set app1 [new Application/Traffic/CBR]
$app1 set packetSize_ 160
$app1 set interval_ 0.020
$app1 attach-agent $src_udp1
$ns_ at 2.0 "$app1 start"

###############Conexiones de Video ############################

set src_udp2 [new Agent/UDP]
$src_udp2 set class_ 2
$src_udp2 set fid_ 2
set dst_udp2 [new Agent/Null]
$ns_ attach-agent $wl_node_(1) $src_udp2
```

```
$ns_ attach-agent $wl_node_(5) $dst_udp2
$ns_ connect $src_udp2 $dst_udp2
set app2 [new Application/Traffic/CBR]
$app2 set packetSize_ 1400
$app2 set rate_ 2000Kb
$app2 attach-agent $src_udp2
$ns_ at 4.0 "$app2 start"

set src_udp3 [new Agent/UDP]
$src_udp3 set class_ 2
$src_udp3 set fid_ 3
set dst_udp3 [new Agent/Null]
$ns_ attach-agent $wl_node_(7) $src_udp3
$ns_ attach-agent $wl_node_(2) $dst_udp3
$ns_ connect $src_udp3 $dst_udp3
set app3 [new Application/Traffic/CBR]
$app3 set packetSize_ 1400
$app3 set rate_ 2000Kb
$app3 attach-agent $src_udp3
$ns_ at 5.0 "$app3 start"

###################Conexiones TCP-FTP ############################

set src_tcp4 [new Agent/TCP]
$src_tcp4 set class_ 2
$src_tcp4 set fid_ 4
$src_tcp4 set packetSize_ 1000
set dst_tcp4 [new Agent/TCPSink]
$ns_ attach-agent $wl_node_(2) $src_tcp4
$ns_ attach-agent $wl_node_(0) $dst_tcp4
$ns_ connect $src_tcp4 $dst_tcp4
set app4 [new Application/FTP]
$app4 set packetSize_ 1000
$app4 attach-agent $src_tcp4
$ns_ at 3.0 "$app4 start"

############Conexión de Telnet#################

set src_tcp5 [new Agent/TCP]
$src_tcp5 set class_ 2
$src_tcp5 set fid_ 5
$src_tcp5 set packetSize_ 400
set dst_tcp5 [new Agent/TCPSink]
$ns_ attach-agent $wl_node_(4) $src_tcp5
$ns_ attach-agent $wl_node_(7) $dst_tcp5
$ns_ connect $src_tcp5 $dst_tcp5
set app5 [new Application/Telnet]
$app5 set packetSize_ 400
$app5 set interval_ 0.100
$app5 attach-agent $src_tcp5
$ns_ at 6.0 "$app5 start"

# Define initial node position in nam
for {set i 0} {$i < $opt(nn)} {incr i} {
    $ns_ initial_node_pos $wl_node_($i) 20
}

# Tell all nodes when the simulation ends
for {set i } {$i < $opt(nn) } {incr i} {
    $ns_ at 60.0000 "$wl_node_($i) reset";
}
```

```
$ns_ at 60.000 "$app0 stop"
$ns_ at 60.000 "$app1 stop"
$ns_ at 60.000 "$app2 stop"
$ns_ at 60.000 "$app3 stop"
$ns_ at 60.000 "$app4 stop"
$ns_ at 60.000 "$app5 stop"
$ns_ at 60.003 "puts \"NS EXITING...\" ; $ns_ halt"

proc stop {} {
    global ns_ tracefd namtrace
    close $tracefd
    close $namtrace
}

puts "Starting Simulation..."
$ns_ run
```

Este mismo script se utilizó para generar tráfico de video a 2,5 Mbits/seg. y 4,0 Mbits/seg., sólo se tuvo que cambiar la siguiente sentencia en la parte del scritp que defines las conexiones de video a 2500Kb y 4000Kb.

```
$appX set rate_ 2000Kb
```

X debe ser reemplazado por el número de conexión correspondiente.

El script siguiente es idéntico al anterior en lo que a la generación de tráfico se referiere pero se aplica Calidad de Servicio.

```
# ======================================================================
# Define options
#
# ======================================================================
set opt(chan)         Channel/WirelessChannel    ;# channel type
set opt(prop)         Propagation/TwoRayGround   ;# radio-propagation model
set opt(netif)        Phy/WirelessPhy            ;# network interface type
set opt(mac)          Mac/802_11e                ;# MAC type
set opt(ifq)          Queue/DTail/PriQ           ;# interface queue type
set opt(ll)           LL                         ;# link layer type
set opt(ant)          Antenna/OmniAntenna        ;# antenna model
set opt(ifqlen)       50                         ;# max packet in ifq
set opt(nn)           9                          ;# number of mobilenodes
set opt(adhocRouting) DSDV                       ;# routing protocol
set opt(x)            15                         ;# x coordinate of topology
set opt(y)            15                         ;# y coordinate of topology

Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 2.0e-14
Phy/WirelessPhy set RXThresh_ 1.77827941e-13
Phy/WirelessPhy set bandwidth_ 11Mb
Phy/WirelessPhy set Pt_ 0.031622777
Phy/WirelessPhy set freq_ 2.472e9
Phy/WirelessPhy set L_ 1

###############Parámetros de 802.11#########################
Mac/802_11 set CWMin_ 31
Mac/802_11 set CWMax_ 1023
Mac/802_11 set SlotTime_ 0.000020
Mac/802_11 set SIFS_ 0.000010
```

```
Mac/802_11 set PreambleLength_ 144
Mac/802_11 set ShortPreambleLength_ 72
Mac/802_11 set PreambleDataRate_ 1.0e6
Mac/802_11 set PLCPHeaderLength_ 48
Mac/802_11 set PLCPDataRate_ 1.0e6
Mac/802_11 set ShortPLCPDataRate_ 2.0e6
Mac/802_11 set RTSThreshold_ 3000
Mac/802_11 set ShortRetryLimit_ 7
Mac/802_11 set LongRetryLimit_ 4
Mac/802_11 set dataRate_ 11.0e6
Mac/802_11 set basicRate_ 1.0e6
############################################################

###############Parámetros de 802.11e#######################
Mac/802_11e set dataRate_ 11.0e6
Mac/802_11e set basicRate_ 1.0e6
Mac/802_11e set backoff_mode_ 1
Mac/802_11e set RTSThreshold_ 3000
Mac/802_11e set PreambleLength_ 72
############################################################

Agent/TCP set packetSize_ 1400
Agent/UDP set packetSize_ 1400

# create simulator instance
set ns_   [new Simulator]

set tracefd  [open wirelessqos.tr w]
set namtrace [open wirelessqos.nam w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

# Create topography object
set topo   [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

# create God
create-god $opt(nn)

# configure for base-station node
$ns_ node-config  -adhocRouting $opt(adhocRouting) \
                  -llType $opt(ll) \
                  -macType $opt(mac) \
                  -ifqType $opt(ifq) \
                  -ifqLen $opt(ifqlen) \
                  -antType $opt(ant) \
                  -propType $opt(prop) \
                  -phyType $opt(netif) \
                  -channelType $opt(chan) \
                  -topoInstance $topo \
                  -wiredRouting ON \
                  -agentTrace ON \
                  -routerTrace OFF \
                  -macTrace OFF


for {set i 0} {$i < $opt(nn)} {incr i} {
 set wl_node_($i) [$ns_ node]
 $wl_node_($i) random-motion 0
```

```
}

$wl_node_(0) set X_ 0.0
$wl_node_(0) set Y_ 1.0
$wl_node_(0) set Z_ 0.0

$wl_node_(1) set X_ 0.0
$wl_node_(1) set Y_ 6.0
$wl_node_(1) set Z_ 0.0

$wl_node_(2) set X_ 9.0
$wl_node_(2) set Y_ 4.0
$wl_node_(2) set Z_ 0.0

$wl_node_(3) set X_ 10.0
$wl_node_(3) set Y_ 5.0
$wl_node_(3) set Z_ 0.0

$wl_node_(4) set X_ 5.0
$wl_node_(4) set Y_ 7.0
$wl_node_(4) set Z_ 0.0

$wl_node_(5) set X_ 5.0
$wl_node_(5) set Y_ 12.0
$wl_node_(5) set Z_ 0.0

$wl_node_(6) set X_ 10.0
$wl_node_(6) set Y_ 2.0
$wl_node_(6) set Z_ 0.0

$wl_node_(7) set X_ 9.0
$wl_node_(7) set Y_ 6.0
$wl_node_(7) set Z_ 0.0

$wl_node_(8) set X_ 2.0
$wl_node_(8) set Y_ 5.0
$wl_node_(8) set Z_ 0.0

################ Conexiones de Voz #############

set src_udp0 [new Agent/UDP]
$src_udp0 set delay_ 150
$src_udp0 set class_ 2
$src_udp0 set fid_ 0
set dst_udp0 [new Agent/Null]
$ns_ attach-agent $wl_node_(8) $src_udp0
$ns_ attach-agent $wl_node_(1) $dst_udp0
$ns_ connect $src_udp0 $dst_udp0
set app0 [new Application/Traffic/CBR]
$app0 set packetSize_ 160
$app0 set interval_ 0.020
$app0 attach-agent $src_udp0
$ns_ at 1.0 "$app0 start"

set src_udp1 [new Agent/UDP]
$src_udp1 set delay_ 150
$src_udp1 set class_ 2
$src_udp1 set fid_ 1
set dst_udp1 [new Agent/Null]
$ns_ attach-agent $wl_node_(0) $src_udp1
$ns_ attach-agent $wl_node_(3) $dst_udp1
```

```
$ns_ connect $src_udp1 $dst_udp1
set app1 [new Application/Traffic/CBR]
$app1 set packetSize_ 160
$app1 set interval_ 0.020
$app1 attach-agent $src_udp1
$ns_ at 2.0 "$app1 start"

###############Conexiones de Video ############

set src_udp2 [new Agent/UDP]
$src_udp2 set delay_ 200
$src_udp2 set class_ 2
$src_udp2 set fid_ 2
set dst_udp2 [new Agent/Null]
$ns_ attach-agent $wl_node_(1) $src_udp2
$ns_ attach-agent $wl_node_(5) $dst_udp2
$ns_ connect $src_udp2 $dst_udp2
set app2 [new Application/Traffic/CBR]
$app2 set packetSize_ 1400
$app2 set rate_ 4000Kb
$app2 attach-agent $src_udp2
$ns_ at 4.0 "$app2 start"

set src_udp3 [new Agent/UDP]
$src_udp3 set delay_ 200
$src_udp3 set class_ 2
$src_udp3 set fid_ 3
set dst_udp3 [new Agent/Null]
$ns_ attach-agent $wl_node_(7) $src_udp3
$ns_ attach-agent $wl_node_(2) $dst_udp3
$ns_ connect $src_udp3 $dst_udp3
set app3 [new Application/Traffic/CBR]
$app3 set packetSize_ 1400
$app3 set rate_ 4000Kb
$app3 attach-agent $src_udp3
$ns_ at 5.0 "$app3 start"

#############Conexiones TCP-FTP ##############

set src_tcp4 [new Agent/TCP]
$src_tcp4 set bandwidth_ 1000
$src_tcp4 set class_ 2
$src_tcp4 set fid_ 4
$src_tcp4 set packetSize_ 1000
set dst_tcp4 [new Agent/TCPSink]
$ns_ attach-agent $wl_node_(2) $src_tcp4
$ns_ attach-agent $wl_node_(0) $dst_tcp4
$ns_ connect $src_tcp4 $dst_tcp4
set app4 [new Application/FTP]
$app4 set packetSize_ 1000
$app4 attach-agent $src_tcp4
$ns_ at 3.0 "$app4 start"

############Conexión de Telnet################

set src_tcp5 [new Agent/TCP]
$src_tcp5 set delay_ 1000
$src_tcp5 set class_ 2
$src_tcp5 set fid_ 5
$src_tcp5 set packetSize_ 400
set dst_tcp5 [new Agent/TCPSink]
```

```
$ns_ attach-agent $wl_node_(4) $src_tcp5
$ns_ attach-agent $wl_node_(7) $dst_tcp5
$ns_ connect $src_tcp5 $dst_tcp5
set app5 [new Application/Telnet]
$app5 set packetSize_ 400
$app5 set interval_ 0.100
$app5 attach-agent $src_tcp5
$ns_ at 6.0 "$app5 start"

# Define initial node position in nam
for {set i 0} {$i < $opt(nn)} {incr i} {
    $ns_ initial_node_pos $wl_node_($i) 20
}

# Tell all nodes when the simulation ends
for {set i } {$i < $opt(nn) } {incr i} {
    $ns_ at 60.0000 "$wl_node_($i) reset";
}

$ns_ at 60.000 "$app0 stop"
$ns_ at 60.000 "$app1 stop"
$ns_ at 60.000 "$app2 stop"
$ns_ at 60.000 "$app3 stop"
$ns_ at 60.000 "$app4 stop"
$ns_ at 60.000 "$app5 stop"
$ns_ at 60.003 "puts \"NS EXITING...\" ; $ns_ halt"

proc stop {} {
    global ns_ tracefd namtrace
    close $tracefd
    close $namtrace
}

puts "Starting Simulation..."
$ns_ run
```

# Bibliografía

Tutorial de IPv6 - ConsultIntel - Palet, Jordi
Excelente tutorial para comenzar a entender por qué se necesita IPv6 y sus características.
http://bjcu.uca.edu.ni/web_biblioteca/LibrosIsti/Tutorial_de_IPV6.pdf

Understanding IPv6 by David Morton
Basado en la primeras RFCs que definieron el protocolo pero no está desactualizado. Es simple y no demasiado técnico
www.itp-journals.com/nasample/c0655.pdf

IPv6 for Cisco IOS Software File 1 Of 3: Overview
Excelente resumen sobre la funcionalidad de IPv6.
http://www.cisco.com/ipv6

Cisco - The ABCs of IPv6
Un muy buen documento explicando IPv6 de manera general.
http://www.cisco.com/ipv6

Neighbor Discovery and Stateless Autoconfiguration in IPv6 By Thomas Narten
Un paper excelente para aprender como trabajan estos dos protocolos. Recomiendo su lectura.
www.cs.ucsb.edu/~almeroth/classes/ F99.595N/autoconfig.pdf

RFC 2185 - Routing Aspects of IPv6 Transition - Septiembre 1997
Este documento da una introducción a los aspectos a tener en cuenta de ruteo en la transición de IPv4 a IPv6.

RFC 2373 - IP Version 6 Addressing Architecture - Julio 1998
Describe la arquitectura de las direcciones IPv6. Incluye: el modelo de direcciones, su representación textual, definición de direcciones unicast, anycast, etc.

RFC 2374 - An IPv6 Aggregatable Global Unicast Address Format - Julio 1998
Describe un formato de dirección IPv6, para usar en Internet, diseñado para facilitar un ruteo escalable.

RFC 2460 - Internet Protocol, Version 6 Specification - Diciembre 1998
Este documento especifica la versión 6 del protocolo Internet.

RFC 2461 - Neighbor Discovery for IP versión 6 - Diciembre 1998.
Esta RFC explica cómo interactúan los nodos vecinos en un link. Cubre varios protocolos de IPv4, y es una de los aspectos más importantes de IPv6. No se puede conocer el funcionamiento de IPv6 si no se la lee.

RFC 2462 - IPv6 Stateless Address Autoconfiguration - Diciembre 1998
Explica los pasos que un nodo debe seguir para autoconfigurar sus interfaces. Idem a la RFC anterior.

RFC 2463 - Internet Control Message Protocol (ICMPv6) for IP Version 6 Specification Diciembre 1998.
Describe un conjunto de mensajes ICMP para usar con la versión 6 del protocolo IP.

RFC 2464 - Transmission of IPv6 Packets over Ethernet Networks - Diciembre 1998
Describe el formato de los frames para transmitir paquetes IPv6, y el método para formar las direcciones de link-local.

RFC 2473 - Generic Packet Tunneling in IPv6 Specification - Diciembre 1998
Este documento define el modelo y los mecanismos genéricos para la encapsulación de paquetes IPv6 o IPv4 en túneles IPv6.

RFC 2529 - Transmission of IPv6 over IPv4 Domains without Explicit Tunnels - Marzo 1999
Especifica un método que permite que nodos IPv6 aislados puedan funcionar utilizando un dominio IPv4, sin necesidad de establecer un túnel manual.

RFC 2711 - IPv6 Router Alert Option - Octubre 1999
Idem a la RFC 2113, pero para IPv6.

RFC 2772 - 6Bone Backbone Routing Guidelines - Febrero 2000
Este documento provee una guía a los operadores de los equipos de ruteo del 6bone para usarlo como referencia en el desarrollo de un sistema de ruteo estable y escalable.

RFC 2893 - Transition Mechanisms for IPv6 Hosts and Routers - Agosto 2002
Este documento describe los mecanismos de compatibilidad que puede utilizarse para que trabajen conjuntamente los IPv4 e IPv6.

RFC 3056 - Connection of IPv6 Domains via IPv4 Clouds - Abril 2003
Especifica un mecanismo interno para que sitios IPv6 se puedan comunicar entre ellos a través de una infraestructura de ruteo IPv4 sin establecer túneles explícitos.

RFC 3177 - IAB/IESG Recommendations on IPv6 Address Allocations to Sites - Septiembre 2001
Este documento es una recomendación, a las organizaciones correspondientes, de cómo se debe realizar la asignación de bloques de direcciones IPv6 a los sitios.

RFC 3513 - Internet Protocol Version 6 (IPv6) Addressing Architecture - Abril 2003
Esta RFC es una actualización de la RFC 2373.

Cisco - Chapter 2 - IPv6 Addressing
Explica en forma detallada como están compuestas las direcciones en IPv6. Es un archivo pdf que se puede conseguir en la siguiente dirección.

IEEE 802.11 - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications
Estándar de la IEEE que especifica el funcionamiento de las redes wireless 802.11

802.11 wireless networks - The definitive guide - O´Reilly - Matthew Gast - 2005
Excelente libro sobre las redes wireless 802.11. Muy claro y fácil de leer. Además expllica 802.11 sobre distintos Sistemas Operativos

802.11 Wireless LAN Fundamentals – Cisco Press – Rpshan, Leary – 2003
Muy buen libro, en el cual, además de las características de las redes wireless, se explican movilidad y se dan consejos de cómo desplegar una red wireless

Wireless Network Operation - SOI Asia IT workshop – 2005
Una detalla presentación del funcionamiento de una red wireless. Muestra como configurar un AP
3Com - IEEE 802.11b Wireless LANs – 2000
Un buen documento describiendo el funcionamiento de las redes wireless 802.11b

RFC 1633 - Integrated Services in the Internet Architecture: an Overview – Braden, Clark y Shenker – 1994
Define la arquitectura de servicios integrados para proveer Calidad de Servicio

RFC 2474 - Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers - Nichols, Black, Baker y Blake - 1998
Este documento explica el campo Differentiated Services (DS). Además define un conjunto procedimientos en el reenvio de paquetes

RFC 2475 - An Architecture for Differentiated Services
Esta RFC define la implementación de una arquitectura escalable de servicios diferenciados. Debería ser leída con la RFC 2474.

Quality of Service: Delivering QoS on the Internet and in Corporate Networks
Paul Ferguson and Geoff Huston – John Wiley & Sons - 1998
Excelente libro sobre Calidad de Servicio. Además de Calidad de Servicio sobre IP también la explica sobre Frame Relay y ATM.

QoS in IP networks – Kun I. Park – Springer - 2005
Este libro explica Calidad de Servicio empezando con conceptos matemáticos. Incluye Calidad de Servicio sobre IP, ATM y MPLS.

IEEE 802.11e - Amendment 8: Medium Access Control (MAC) Quality of Service enhancements
Estándar de la IEEE. Es la enmienda a las redes IEEE 802.11 para que puedan proveer Calidad de Servicio. No se puede leer sin tener el estándar original de la IEEE 802.11 a mano.

IEEE 802.11e Wireless LAN for Quality of Service - Stefan Mangold, Peter May, Ole Klein
Documento que presenta una breve introducción al estándar 802.11e y realiza simulaciones para comprobar el funcionamiento del mismo.
www.ing.unipi.it/ew2002/proceedings/H2006.pdf

Understanding MAC protocol architectural implications of 802.11 QoS amendments: A guide to IEEE 802.11e technology - Simon Cheng, Kamila Pinchota
Introducción al estándar 802.11e. También presenta una introducción al estándar 802.11

QoS issues and enhancements for IEEE 802.11 Wireless LAN - Qiang Li, Thierry Turletti
Este documento describe el funcionamiento del estandar 802.11e. También presenta distintas alternativas para brindar Calidad de Servicios.
http://www-sop.inria.fr/planete/qni/RR-4612.pdf

Supporting Multimedia Traffic in 802.11e WLANs - Casetti, Chiasserini, Merello, Olmo
Documento donde se estudia como trabaja una red 802.11e cuando se utilizan distintos tipos de tráficos, como VoIP y video.

Simulation Study of IEEE 802.11e EDCF - Dajiang He, Charles Q. Shen
Documento en el que se realizan simulaciones para comprobar el funcionamiento de método EDCF del estándar 802.11e.
http://www.cs.columbia.edu/~charles/publication/vtc2003.pdf

The ns manual – UC Berkeley
Manual del simulador NS-2.

NS2 - Network Simulator
Este documento presenta una breve introducción al simulador y describe un pequeño script inicial.

A brief guide to NS-2 - Sandeep Gupta
Documento donde se muestra como crear una red wireless usando el simulador
http://profile.iiita.ac.in/sandeep_wc02/ns/nstutorial.pdf