ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA
ESCUELA DE INGENIERÍA ELECTRÓNICA

"IMPLEMENTACIÓN DE UN MECANISMO PARA CALIDAD DE SERVICIO BASADO EN EL ALGORITMO DRAP EN REDES INALÁMBRICAS DE ÁREA LOCAL DE TIPO INFRAESTRUCTURA"

TESIS DE GRADO

Previa la obtención del título de

Ingeniero en Electrónica y Computación

Presentado por:

VERONICA MARIBEL ARCOS AZQUI

JORGE FABRICIO ALVAREZ HARO

Riobamba-Ecuador

2010

A mis padres Ángel y Mariana por su amor, comprensión, me han dado todo lo que soy como persona, mis valores, principios, perseverancia, empeño y me han enseñado a enfrentar las adversidades sin perder nunca la dignidad ni desfallecer en el intento.

A mi hermana y segunda madre Nubia Cristina, que no alcanzo a ver los resultados pues partió tempranamente de esta vida, de carácter fuerte quien me demostró un amor inigualable, convirtiéndose en mi ejemplo de valentía, capacidad, generosidad, superación e inspiración a seguir.

A mi sobrino Joel Adrián, alegría de mi vida quiero dedicarte el haber finalizado esta meta trazada la cual no hubiese sido posible sin el apoyo incondicional tanto sentimental como económico por parte de tu Mami, que hoy desde el cielo velara por nosotros.

JORGE FABRICIO ALVAREZ HARO

A mis padres Carmen y Oswaldo por todos sus sacrificios y renuncias realizadas para permitirme forjar mi carrera, a mis hermanos Fabián, Patricia, Nelson, Miriam y Humberto de quienes siempre tuve su apoyo incondicional, a mis tíos y abuelitos que confiaron en mí.

A mis sobrinos Adriana, William, Jessica, Jair y Karen, aquellas personitas muy importantes en mi vida quienes dieron fuerzas para luchar y conseguir esta meta tan anhelada.

A la familia Peña Barreno quienes me recibieron en su hogar y me dieron calor familiar.

VERONICA MARIBEL ARCOS AZQUI.

A Dios, por haberme guiado por el camino del bien y haberme bendecido al ponerme en un hogar maravilloso al nacer y contar con unos padres excepcionales.

A mis Padres Angel y Mariana, mi segunda madre y hermana, Nubia, que ahora estácerca a Dios y que ha estado y permanecerá conmigo todo el tiempo, mi hermano Fredy, así como a todos mis tíos; por siempre haberme dado su fuerza y apoyo, sin dudar ni un solo momento en mi inteligencia y capacidad.

A nuestro Director de Tesis Ing. Alberto Arellano, quién nos ayudó en todo momento con sus ideas y orientaciones al desarrollo de este proyecto.

A la Escuela Superior Politécnica de Chimborazo por abrirnos sus puertas a jóvenes como nosotros, preparándonos para un futuro competitivo.

JORGE FABRICIO ALVAREZ HARO

A Dios nuestro padre celestial por brindarme salud y vida, a la Santísima Virgen del Rosario de Agua Santa, a San Vicente Ferrer; por ayudarme a concluir con éxito esta etapa de mi vida.

Al Ingeniero Alberto Arellano, Director de Tesis por su apoyo incondicional para la realización de esta tesis. Al Ingeniero Renato Barragán por sus comentarios y sugerencias.

A Dr. Nubia Álvarez que fue ejemplo de vida, lucha y perseverancia para alcanzar todos los objetivos, y hoy sabemos que desde el cielo se encuentra orgullosa por nuestro triunfo ya que fue pilar importante en la realización de nuestra vida estudiantil, su recuerdo se mantendrá vivo en nuestra mente y corazón.

A la Escuela Superior Politécnica de Chimborazo por haberme dado la oportunidad de forjarme en sus aulas.

VERONICA MARIBEL ARCOS AZQUI.

FIRMAS DE RESPONSABILIDAD

NOMBRE	**FIRMAS**	**FECHA**
Dr. Romeo Rodríguez DECANO FACULTAD DE INFORMATICA Y ELECTRONICA	……	……
Ing. Paul Romero DIRECTOR DE ESCUELA DE ELECTRONICA	……	……
Ing. Alberto Arellano A DIRECTOR DE TESIS	……	……
Ing. Renato Barragán MIEMBRO DEL TRIBUNAL	……	……
Ing. Wiliam Calvopiña MIEMBRO DEL TRIBUNAL	……	……
Lcdo. Carlos RodrÍguez DIRECTOR DEL CENTRO DE DECUMENTACION	……	……
NOTA DE TESIS	……	

TEXTO DE RESPONSABILIDAD

"Nosotros, Verónica Maribel Arcos Azqui y Jorge Fabricio Álvarez Haro somos responsables de las ideas, doctrinas y resultados expuestos en esta tesis; y, el patrimonio intelectual de la Tesis de Grado pertenece a la ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO."

Verónica Maribel Arcos Azqui

Jorge Fabricio Álvarez Haro

INDICE DE ABREVIATURAS

AES.- Señal de confirmación de recepción de paquetes

AIMD.- El aumento de auditivos / multiplicativo descenso, algoritmo de Control de Retroalimentación utilizado en TCP

ANSI.- Instituto Nacional Estadounidense de Estándares

AP.- Punto de acceso.

API.- Interfaz de Programas de Aplicación

BSD.- Distribución de Software Berkeley

CA.- Evasión de Colisiones

CSMA/CA.- Acceso múltiple por división de tiempo con anulación de colisión.

CSMA/CD.- Acceso múltiple por división de tiempo con detención de colisión

CTS.- Libre para enviar

DEC.- Corporación de Equipamiento Digital

DHCP.- Protocolo de Configuración Dinámica de Host

DS.- Servicio de Distribución

DSCP.- Servicios Diferenciados de Código Puntual

EAP.- Protocolo de Autenticación Extendido

EAPOL.- EAP sobre LAN

FIFO.- Primero en entrar primero en salir

FTP.- Protocolo de Transferencia de Archivos

GPL.- Licencia Pública General

HAL.- Capa de Acceso de Hardware

IEEE.- Instituto de Ingenieros Eléctricos y Electrónicos

IETF.- Grupo de Trabajo en Ingeniería de Internet

ISM.-Bandas de radiofrecuencias para usos médicos, industriales y científicos.

ISO.- Organización Internacional de Estándares

ITU.- Unión Internacional de Telecomunicaciones

MAC.- Control de Acceso al Medio

MADWIFI.- Driver Atheros Multibanda para Aplicaciones Inalámbricas.

NAS.- Servidor de Acceso a la Red

NIC.- Chip de Tarjeta de Red

OSI.- Interconexión de Sistemas Abiertos

PBX.- Central Privada Automática

PCMIA.- Asociación Internacional de Desarrollo de Tarjetas de Memoria

RSA.- Sistema Criptográfico con clave pública

RTS.-Demanda de envío

RTP.-Protocolo de Transporte de Tiempo Real

SSID.-Identificador de Servicio Básico, nombre de Red Inalámbrica

TCP/IP.- Protocolo de control de transmisión/Protocolo de Internet

TKIP.- Protocolo de Integridad de Claves Temporales

TLS.- Seguridad de Capa de Transporte

TOS.-Tipo de Servicio

UDP.-Protocolo de Datagrama de Usuario

VAP.-Access Point Virtual

WEP.- Seguridad Equivalente a la cableada

WPA.-Acceso Protegido para Redes Inalámbricas

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

CAPÍTULO I

MARCO METODOLOGICO

2.2. ANTECEDENTES

Las computadoras han llegado a un punto tan cotidiano y necesario para la difusión de la información que la optimización del flujo de ésta es de vital importancia para las actividades diarias, como por ejemplo; mandar correos electrónicos o comunicarnos en tiempo real.

Definitivamente, los medios tangibles e intangibles que se utilizan para establecer la interconexión de las computadoras es la pieza más importante de la estructura para comunicación por computadoras. A la conjunción de las computadoras y los medios que las comunican e interconectan se denomina a grandes rasgos como una red de comunicación.

La unidad principal de información dentro de estas redes es llamada paquete. Una red de paquetes divide la información enviada a través de ella en

unidades más cortas con la finalidad de que se vayan asegurando pequeñas cantidades de información en el nodo receptor.

Es imprescindible también, no hablar de las redes en casi todos los ámbitos empresariales, escolares y domésticos; es difícil imaginarse no utilizarlas con fines de compartir datos, comunicación, diversión, entre muchos más. En sus inicios, este tipo de redes de comunicación se enlazaban mediante cables, lo cual requiere de una organización estática de las computadoras, ya que la estructura física de la red requiere que los cables estén fijos y conectados a dispositivos centrales, tales como switches o routers, dichas redes se les conoce como redes cableadas y están presentes en la mayoría de los ambientes de redes de computadoras. Debido al avance de la tecnología, han surgido las computadoras portátiles o laptops, usadas con mayor frecuencia día a día; ya que ofrecen nuevos servicios, de los cuales destaca la posibilidad de conectarse a las redes inalámbricas.

Las redes inalámbricas han tomado gran importancia en el mundo de la comunicación debido a que los dispositivos pueden tener movilidad hasta cierto punto, es decir, que deben estar dentro del área de cobertura de la red. Este tipo de redes no se conectan por el medio tradicional de una red cableada; su nombre lo dice, sin cables. Por lo tanto, el medio por el cual una red inalámbrica se comunica es a través de ondas electromagnéticas, las mismas

que no se bloquean ni se distorsionan por objetos sólidos, por lo que pasan fácilmente a través de puertas, tabiques, suelos y techos.

El medio inalámbrico es un recurso muy endeble para realizar comunicaciones, debido a que puede sufrir decadencia en su capacidad de comunicación por fenómenos físicos, lo cual puede provocar pérdidas de información.

Los estándares para WLAN (Wireless Local Area Network) que se definen bajo el estándar 802.11 de la IEEE, siglas de Institute of Electrical and Electronics Engineers, proveen las especificaciones bajo las cuales la comunicación en redes inalámbricas de área local es posible, tomando en cuenta los niveles más bajos de la arquitectura OSI (Open System Interconnection); es decir, la capa física y la de enlace de datos. El 802.11, además, es una familia de protocolos en donde han surgido diferentes técnicas de transmisión por modulación. Las más usadas actualmente se comunican en la banda de frecuencia de los 2.4 Ghz y 5 Ghz, mediante radiofrecuencia, tales como 802.11 b y g, a través de un medio broadcast[1], los cuales requieren de un mecanismo de control de acceso al medio

[1] Modo de transmisión de información donde un nodo emisor envía información a una multitud de nodos receptores de manera simultánea.

que permita controlar cuál y cuándo una estación puede transmitir o recibir paquetes.

La calidad de servicio o Quality of Service, también abreviada como QoS por sus siglas en inglés, juega un papel importante en las redes y para efectos de este trabajo de tesis, se puede dar una definición general. Así, QoS se define como la capacidad que tiene una red para dar gestión preferencial a ciertos flujos de datos, de tal manera que pueda brindar garantía para su transmisión.

Los principales intentos que hay para proporcionar garantías de calidad de servicio en las redes inalámbricas son los que están propuestos en el estándar IEEE 802.11. El primero de ellos es la implementación del DCF (Función de Coordinación Distribuida), el cual está implementado en la mayoría de las tarjetas de hoy en día; sin embargo, este mecanismo fue definido para tráfico best-effort, lo cual no da ninguna garantía de QoS (Calidad de Servicio) para el tráfico de tiempo real. El segundo, es la implementación de un PCF (Función de Coordinación Puntual), el cual para su funcionamiento necesita de la ayuda del DCF, por lo cual tampoco garantiza calidad de servicio en este tipo de redes. Otras propuestas que intenta garantizar QoS son SWAN (Stateless Wireless Ad Hoc Network - Estado de red inalámbrica Ad-Hoc), para redes de tipo Ad-Hoc[2], y DRAP (Dual-Queue Rate

[2] Tipo de red inalámbrica en la cual no se necesita de un access point o dispositivo central para que los nodos se comuniquen entre ellos.

Controlled Access Point – Punto de Acceso de Velocidad Controlada por Doble Cola), para redes de tipo infraestructura.

En el presente trabajo de investigación se realiza la implementación de un mecanismo para garantizar calidad de servicio basado en el algoritmo DRAP mediante el uso de herramientas de software libre, dicho algoritmo fue desarrollado por Máster Rubén González Benítez.

2.3. JUSTIFICACIÓN

Las aplicaciones en tiempo real (comunicación multimedia, transacciones bancarias) como es sabido, requieren de muchos recursos en una red para poder llevarse a cabo de manera adecuada y entendible. Al hablar de multimedia, estamos haciendo referencia al conjunto de vídeo, audio, animaciones, lo cual es equivalente en primer lugar a consumo de recursos hablando estrictamente de una computadora. Segundo, que es el que nos incumbe, necesita que la red no tenga demasiado tráfico para poder entablar una buena comunicación y además que haya gestión sobre los paquetes, lo cual no siempre es posible. En redes inalámbricas, la comunicación de este tipo presenta retardo en los paquetes, pérdida de los mismos, debido a que no hay control sobre qué tipo de paquetes se debe dar prioridad o porque el enlace entre los nodos que se comunican se saturan de información.

Mediante el uso del algoritmo DRAP en la implementación de un mecanismo que brinde calidad de servicio (QoS) para las aplicaciones en tiempo real se dará una gestión preferencial a los paquetes de audio y controlará las fuentes de video del tráfico best-effort, de tal manera de garantizar al flujo de audio una mejor calidad en el servicio de multimedia.

Para ello, en esta tesis se lleva a cabo el desarrollo de una capa intermedia basándose en el algoritmo DRAP, mediante herramientas del sistema operativo GNU/Linux. Con la implementación de esta capa se pretende proporcionar calidad de servicio en las redes inalámbricas de área local, utilizando la distribución GNU/Linux (Debian) como plataforma de trabajo.

Después de realizar un middleware para calidad de servicio (capa intermedia para la clasificación del flujo de tráfico y dar prioridad a las aplicaciones en tiempo real), siguiendo el modelo DRAP, con todos los elementos que requiere, el paso siguiente es realizar las pruebas pertinentes para corroborar o refutar la hipótesis de este anteproyecto.

2.4. OBJETIVOS

2.4.2. OBJETIVO GENERAL

- Implementar y evaluar un mecanismo para calidad de servicio basado en el algoritmo DRAP para redes inalámbricas de área local de tipo infraestructura.

2.4.3. OBJETIVOS ESPECÍFICOS

- Estudiar el algoritmo DRAP y su aplicación a un mecanismo que brinde Calidad de Servicio (QoS) en redes inalámbricas de área local (WLAN), con la integración del sistema operativo GNU/Linux.

- Desarrollar e implementar una capa intermedia (Middleware) basándose en el algoritmo DRAP usando herramientas de software libre, para clasificar el tráfico multimedia sobre otro tipo de tráfico, priorizando el audio, y controlando las fuentes de vídeo del tráfico best-effort.

- Evaluar los resultados mediante la implementación de un escenario de tráfico en tiempo real, donde podamos comprobar que los resultados

obtenidos con el mecanismo DRAP son superiores que los que ofrece el estándar PCF en redes inalámbricas referente a los servicios de aplicaciones multimedia.

2.5. HIPÓTESIS

A través de la implementación de un mecanismo de calidad de servicio basado en el algoritmo DRAP usando software libre, que se garantizará el flujo de tráfico de aplicaciones en tiempo real (reservación ancho de banda, priorizar paquetes) en las redes inalámbricas de tipo infraestructura en comparación con el estándar PCF

CAPÍTULO II

MARCO TEORICO-CONCEPTUAL

2.1 Soporte del Sistema Operativo GNU/LINUX para Redes Inalámbricas

Como ya se ha hecho mención, el mercado wireless ha tenido un enorme crecimiento en los últimos años; el cual se atribuye a la gran oferta de productos con esta tecnología, que satisfacen las necesidades de los usuarios. Por tal motivo es necesario introducirse en las capacidades que posee el sistema operativo GNU/Linux para trabajar con las WLAN, las herramientas para proporcionar gestión al tráfico de red, a modo de establecer preferencias, para configurar una tarjeta de red inalámbrica; asimismo como la característica que tiene para clasificar paquetes de acuerdo a ciertas necesidades, la cual es importante para el tema central de este trabajo. Todas estas herramientas se ocuparán para este proyecto de tesis, debido a las facilidades que ofrece GNU/Linux con respecto a otros sistemas operativos.

2.1.1. Sistema operativo GNU/Linux

GNU/Linux es un sistema operativo de núcleo monolítico[3] el cual fue inicialmente creado como un pasatiempo por un joven estudiante llamado Linus Torvalds, en la Universidad de Helsinki en Finlandia. Linus tuvo un interés en Minix, un pequeño sistema Unix y decidió desarrollar un sistema que excediera los estándares de Minix. Él comenzó su trabajo en 1991 cuando liberó la versión 0.02 y trabajó constantemente hasta 1994 cuando la versión 1.0 del kernel de Linux fue liberada. El kernel, es el corazón de todos los sistemas GNU/Linux, es desarrollado y liberado bajo la licencia GNU General Public Licence, la cual es una licencia que está orientada principalmente a proteger la libre distribución, modificación y uso del software. Asimismo, su código está libremente disponible para quien lo requiera. El kernel es el que forma la base alrededor del cual un sistema operativo GNU/Linux se desarrolla. Actualmente hay cientos de compañías y organizaciones y un igual número de individuos que han liberado sus propias versiones de sistemas operativos basados en el kernel de Linux.

Aparte del hecho de que es libremente distribuido, GNU/Linux es funcional, adaptable y robusto y lo ha hecho la principal alternativa para sistemas propietarios

[3] Núcleo grande y complejo, que engloba todos los servicios del sistema, obviamente en un solo núcleo.

como Unix y Microsoft, IBM, Hewlett-Packard y otros gigantes del mundo de la computación han abarcado y entrado al mundo GNU/Linux.

Durante su segunda década de existencia, ha sido adoptado mundialmente como una plataforma de servidor, esto quiere decir que su mayor aplicación está en convertirse en plataforma para instalar y administrar servicios diversos como páginas Web, correo, File Transfer Protocolo o FTP, entre muchísimos otros. Su uso como escritorio también está disponible. También está disponible para ser incorporados directamente en microchips y cada vez más está siendo usado en aparatos electrodomésticos y dispositivos. GNU/Linux es usado como sistema operativo en una amplia variedad de plataformas de hardware y computadoras, incluyendo los computadores de escritorio, tales como PCs x86[4] y x86-64, y Macintosh, servidores, supercomputadoras, mainframes[5], así como también en teléfonos celulares.

La expresión de Linux es utilizada para nombrar a las distribuciones que existen de este sistema, las cuales son diferentes versiones de GNU/Linux que corren variadas aplicaciones pero que contienen el mismo núcleo, también se les conoce como distros o distribuciones.

[4] Denominación genérica dada a ciertos microprocesadores de la familia Intel, sus compatibles y a la arquitectura básica de estos procesadores

[5] Computadora grande, potente y costosa usada principalmente por una gran compañía para el procesamiento de una gran cantidad de datos.

Dentro de cada distribución, siempre hay pequeñas o grandes diferencias.

Primero hay diferencias en cuanto al entorno de escritorio, ya que muchas varían su interfaz como la colocación y forma de íconos, las barras donde se ubican los programas, etc.

También tienen diferencias en cuanto a los comandos que ocupan (puede ser el mismo comando pero se escribe de diferente manera) y su intérprete como puede ser bash, ksh, csh, etc. De igual forma hay variación en cuanto a las rutas donde se guardan los archivos de los programas, y de muchas configuraciones. Por ejemplo, un archivo de configuración de un servidor ssh[6] se puede encontrar en la ruta /etc/ssh/sshd.conf en una distribución y en /etc/sshd.conf en alguna otra como Debian.

La distribución que se ocupará para desarrollar este trabajo de investigación será una de las más famosas y utilizadas a nivel mundial Debian.

Debian, es conocido como el sistema operativo universal, viene con 18000 programas pre-compilados que hacen más fácil su instalación. Posee un gestor de paquetes muy intuitivo y fácil debido a que instala dependencias automáticamente.

[6] Nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red

En este caso, se utilizará la versión Debian Etch (4.0), uno de los sistemas operativos más usados junto con RedHat, es desarrollado por personas voluntarias y se mantiene a base de donaciones de cualquier tipo de personas, aún así es una de las distribuciones más estables y seguras del medio. Sus desarrolladores están comprometidos con la filosofía del software libre descartando llevar paquetes aunque sea con un mínimo aspecto de software privativo[7] e incluyendo sólo software cien por ciento libre, además soporta un total de once arquitecturas de procesador e incluye los entornos KDE[8], GNOME[9], igualmente programas criptográficos, es compatible con la versión 2.3 y con aquellos programas desarrollados para la versión 3.1, contiene un proceso de instalación totalmente integrado, con soporte de particiones cifradas, introduce una nueva interfaz gráfica del sistema de instalación que soporta tanto grafías que utilizan caracteres compuestos como lenguas complejas. El sistema de instalación de Debian GNU/Linux ahora está traducido a 58 idiomas.

También se puede decir de Debian que es una de las distribuciones GNU/Linux que se utiliza como base para otras múltiples distribuciones como Knoppix, Linspire, MEPIS, Xandros y la familia Ubuntu, es el más utilizado para implementar servicios de red tales como servidores ssh, servidores web, de correo, entre muchos otros.

---

7 Software en el cual los usuarios no están permitidos o limitados para hacer modificaciones, redistribuirlo o incluso usarlo libremente y no pueden obtener el código fuente del mismo.

8 Se basa en el principio de la personalización. Todos los componentes de KDE pueden ser configurados en mayor o menor medida por el usuario.

9 Entorno de escritorio de Linux para sistemas operativos de tipo Unix bajo tecnología X Windows está más basado a lo funcional que un diseño bonito, muy intuitivo y rápido para acceder a las aplicaciones.

Debian es considerado como una de las distribuciones más estables y completas en cuanto a versiones de kernel, repositorios[10] y aplicaciones, fue desarrollado con vista en administración de servicios y desarrollo y no para usuarios finales. Es por ello que se que se ha elegido esta distribución para el desarrollo de este proyecto.

2.1.2. GNU/Linux y el soporte para redes inalámbricas

Actualmente el sistema operativo GNU/Linux es uno de los más potentes y útiles para el manejo y administración de redes inalámbricas, así como para la administración y configuración de las tarjetas de dichas redes. Desde la configuración de una dirección, hasta controlar el ancho de banda mediante ciertos programas. En fin, es un mundo de posibilidades para las WLAN cuando se utiliza GNU/Linux como plataforma de administración. En esta sección se explicará detalladamente el uso de cada herramienta wireless para mayor información.

Dentro de este contexto se encuentra a una persona que ha sido muy importante en el tema de las redes inalámbricas en su soporte bajo el sistema operativo GNU/Linux, sin importar su distribución: Jean Tourrilhes, quien es el autor del modulo de wireless para Linux que se utilizan para poner a trabajar una tarjeta de red inalámbrica después de que ya haya sido instalado.

[10] Espacio de almacenamiento desde el cual los paquetes de software pueden ser recuperados e instalados en una computadora GNU/Linux.

2.1.2.1. La extensión wireless

Linux wireless LAN es un proyecto de código abierto (Open Source Project) patrocinado por Hewlett Packard, a través de la contribución de Jean Tourrilhes desde 1996 y es construido con el aporte de muchos usuarios de GNU/Linux alrededor del mundo, con el cual el proyecto va mejorando cada día.

Este proyecto inició cuando Jean Tourrilhes intentó de instalar una tarjeta Wavelan[11] en una computadora con GNU/Linux, él tenía dos diferentes versiones del driver para esta tarjeta, uno para la Personal Computer Memory Card International Association o por sus siglas, PCMCIA y otro para la ranura Industry Standard Architecture (ISA) totalmente diferentes con métodos diferentes para instalarlos y debido a que el sistema operativo Microsoft Windows no lo dejó cambiar dichos parámetros optó por modificar el código del driver.

Es por ello que decidió programar una interfaz o API (Application Programming Interface) para las tarjetas inalámbricas, la cual permita a los usuarios manipular cualquier dispositivo de red inalámbrico en una forma estándar y uniforme. Por supuesto, estos dispositivos son fundamentalmente diferentes, así que la estandarización sólo seria en los métodos pero no en los valores.

[11] Controlador de dispositivo de bajo nivel para la tarjeta inalámbrico WaveLAN ISA de NCR / AT&T / Lucent y RoamAbout DS de Digital (DEC).

Esta interfaz debería ser flexible y extensible. La necesidad fundamental fue la configuración de los dispositivos, pero las estadísticas también es deseable. Se necesitó también algo simple de implementar y conforme al estándar de GNU/Linux para tener algo mucho más fácil de compartir y mantener.

La interfaz necesitaría evolucionar con la aparición de nuevos productos y con necesidades específicas. Se intentó ser lo más genérico posible pero Tourrilhes estuvo obligado a restringirse en un conjunto de dispositivos específicos, enfocándose a dispositivos basados en tecnología Wireless LAN. Como sólo fue una extensión al actual interfaz de red de Linux, él decidió llamarla Wireless Extensions o Extensiones Inalámbricas. Las palabras interfaz y API son también ambiciosas para este simple conjunto de herramientas.

Las Wireless Extensions han sido implementadas en tres partes complementarias. La primera es la interfaz de usuario, un conjunto de herramientas para manipular esas extensiones. La segunda parte es una modificación del kernel de Linux para soportar y definir las extensiones. Y la tercera, es la interfaz del hardware, la cual es implementada en cada driver del nodo para verificar las extensiones a las actuales manipulaciones del hardware. Las modificaciones del kernel han sido incluidas en las versiones 2.0.30 a la 2.1.17. Por defecto, las Wireless Extensions están deshabilitadas cuando una distribución GNU/Linux es instalada. Por lo tanto, se requiere realizar un proceso de recompilación del kernel y habilitar la opción CONFIG_NET_RADIO para el soporte wireless.

Dependiendo de la versión del kernel, el nombre de la opción puede variar.

Las herramientas deben ser capaces de trabajar en cualquier sistema GNU/Linux que haya sido compilado con la opción anterior. Las modificaciones de los drivers es probablemente el reto más importante de los creadores de cada driver. Cada uno necesita soportar las wireless extensions y permitir el correspondiente diálogo con el hardware específico.

La interfaz de usuario está compuesta de tres programas y una entrada /proc en Linux. La ruta /proc/net/wireless es un archivo que da información y estadísticas sobre el actual sistema inalámbrico. Se despliega en pantalla los siguientes puntos, los cuales son proporcionados por cada dispositivo:

- Status: Su actual estado. Este es una información independiente por cada dispositivo.
- Quality - link: calidad general de recepción.
- Quality - level: fuerza de la señal en el receptor.
- Quality - noise: nivel de silencio (sin paquetes) en el receptor.
- Discarded - nwid: número de paquetes descartados debido al id de la red inválido.
- Discarded - crypt: número de paquetes incapaces de descifrar.
- Discarded - misc: aún no está en uso

Esta información permite tener a los usuarios un mejor panorama del sistema. Así por ejemplo, un alto valor de Discarted – nwid puede indicar que hay un problema de configuración del nwid o que hay una red adyacente. La diferencia entre Quality – link y Quality - level es que la primera indica qué tan buena es la recepción y la segunda qué tan fuerte es la señal. Cuando los valores de Quality han sido actualizados desde la última leída de la entrada, un punto seguirá al valor. Los otros tres programas, forman parte de las herramientas inalámbricas o wireless tools: iwconfig, iwlist e iwspy, detalladas a continuación.

2.1.2.2. Las herramientas wireless

Las herramientas wireless o wireless tools y la extensión wireless (wireless extension) son también proyecto de código abierto patrocinado por Hewlett Packard y desarrollados y mantenidos por la misma persona creadora de las wireless extensions y por la comunidad GNU/Linux. En esta sección se explicarán las herramientas wireless de Linux o mejor conocidas pos su nombre el inglés: Wireless tools. Las cuales son un conjunto de herramientas que permiten manipular las Wireless Extensions.

Las wireless tools usan una interfaz textual y son bastante ordinarias para cualquier persona que haya utilizado la línea de comandos de algún sistema GNU/Linux, pero son importantes porque soportan las extensiones completas. A continuación se

listan los comandos disponibles y sus posibles parámetros de las wireless tools. Los cuales, como se logra apreciar son amplios y para diversos objetivos.

2.1.2.2.1. iwconfig

Manipula los parámetros básicos de la interfaz inalámbrica. Es un clon de ifconfig, usado para la configuración de dispositivos estándares, como interfaces para redes alambradas. Un ejemplo básico y sencillo del uso de este comando podría ser: iwconfig eth0 essid “una red”, en donde eth0 es el nombre de la interfaz inalámbrica y “una red” es el nombre de la red inalámbrica.

- **ap**: Este parámetro registra el host a un Access Point mediante la dirección MAC.
- **commit**: Aplica los cambios realizados a una interfaz.
- **essid**: Sirve para indicar el nombre de la red a la que se quiere conectar.
- **frag**: Indica el tamaño en que se fragmentarán los paquetes.
- **freq**: Indica la frecuencia de conexión a la que operara el host (k/M/G).
- **channel**: Utilizado para indicar el canal en que actúa o en el que funcionará el dispositivo.
- **key**: Utilizado para indicar la llave en caso de que la red cuente con algún tipo de cifrado.

- **mode**: Utilizado para indicar el modo en que trabajará la tarjeta. Los modos en los que puede operar son: Ad-hoc, Managed, Master, Repeater, Monitor y Secondary.
- **nick**: Sirve para poner un nombre a la estación de trabajo.
- **nwid**: Utilizada para poner un identificador de red. Este parámetro es utilizado sólo por el hardware que funciona con versiones anteriores al 802.11.
- **power**: Para manipular el ahorro de energía.
- **rate**: Define el bitrate o tasa de transferencia a la que funcionará la interfaz.
- **rts**: Agrega una confirmación a un paquete antes de ser enviado, para asegurarse de que el canal en que se está trabajando se encuentre limpio.
- **sens**: Indica la sensibilidad de umbral mínima.
- **txpower**: Define la potencia de envío dBm.

2.1.2.2.2 **Iwlist**

Es usado para desplegar alguna información adicional de una interfaz de red inalámbrica, la cual no es desplegada por el comando iwconfig. Permite iniciar el escaneo y listar las redes inalámbricas al alcance, su tasa de transferencia, las llaves, entre muchas otras características.

Cuenta con los siguientes parámetros:

- **ap / peers / access point:** despliega una lista de los Access Point que se encuentran dentro del rango o cobertura. En ocasiones también muestra la calidad del enlace.
- **bit / bitrate / rate:** muestra las tasas de transferencia soportadas por el dispositivo.
- **channel / freq / frequency:** muestra una lista de los canales y la frecuencia a la que puede trabajar cada uno de estos.
- **enc / key / encryption:** muestra los tipos de cifrados soportados y el tamaño de las claves de cifrado.
- **event:** lista de eventos soportados por el dispositivo.
- **power:** muestra los atributos de ahorro de energía que pueden ser utilizados por la interfaz.
- **retry:** muestra el límite de retransmisiones y la duración de las retransmisiones en el dispositivo.
- **scan / scanning:** muestra una listas de la los Access Point o nodos Ad-Hoc a los que se puede conectar el host.
- **txpower:** muestra la potencia de transmisión de la interfaz.
- **-versión** muestra la versión de las herramientas, así como la recomendada y actual Wireless Extensions versión de la herramienta y de las distintas interfaces inalámbricas.

2.1.2.2.3. Iwspy

Es usado para establecer una lista de direcciones para monitorear en una interfaz inalámbrica y para obtener por nodo la calidad del enlace y el ruido. Esta información es la misma que la que está disponible en /proc/net/wireless: Calidad del Enlace, Potencia de la Señal y Nivel de Ruido. Esta información es actualizada cada vez que un nuevo paquete es recibido por lo que cada dirección de la lista, añade algo de sobrecarga en el controlador. Esta función sólo opera para los nodos de parte de la célula inalámbrica actual, no puede controlar los puntos de acceso que no están asociadas (se puede utilizar la exploración para eso) ni en los nodos en otras células.

- **Igetthr:** recupera los valores del umbral, definidos por el parámetro setthr.
- **off:** remueve la lista de direcciones definidas por el parámetro setthr y deshabilita esta opción.
- **setthr:** define el límite inferior y superior del umbral.

2.1.2.2.4. Iwpriv

Permite manipular una extensión inalámbrica o wireless específica de un driver. Podría decirse que es una herramienta un tanto experimental. Algunos drivers, como el Wavelan, pueden definir algunos parámetros o funcionalidades extras, iwpriv sd usado para manipularlas. Sin argumentos, iwpriv los comandos privados

disponibles en cada interfaz y los parámetros que requiera cada uno. En teoría, la documentación de cada dispositivo debe indicar cómo usar esos comandos específicos para cada dispositivo y sus efectos.

- **--all:** muestra una lista de los comandos privados que no requieren parámetros.
- **roam:** habilita o deshabilita el roaming, si este es soportado por la interfaz.
- **port:** lee y reconfigura el tipo de puerto.

2.1.2.2.5. Ifrename

Permite nombrar interfaces basadas en varios criterios estáticos. Es una herramienta que le permite asignar un nombre coherente a cada una de sus interfaces de red. De forma predeterminada, los nombres de interfaz son dinámicos, y cada interfaz de red se le asigna el nombre del primero que esté disponible (eth0, eth1). El orden de las interfaces de red al momento de crearse puede variar, para interfaces incorporadas, el orden de arranque del kernel puede variar, para la interfaz extraíbles, el usuario puede enchufar en cualquier orden. De forma predeterminada, cambia el nombre de todos los sistemas actuales definidas en / etc iftab /.

2.1.2.2.6. iwevent

Muestra los eventos generados durante la conexión. Cada línea despliega el evento específico, el cual describe qué ha sucedido en la interfaz inalámbrica específica. Este comando no toma argumentos. Hay dos clases de eventos inalámbricos. El primero de ellos es relacionado a un cambio de configuración en la interfaz (típicamente hecho a través de iwconfig). Sólo son reportadas configuraciones que pueden resultar de una interrupción de conectividad. Todos esos eventos serán generados en todas las interfaces inalámbricas por el subsistema del kernel que soporta esto. La segunda clase de eventos son generados por el hardware cuando algo sucede o una tarea ha sido finalizada.

2.1.2.2.7. iwgetid

Proporciona información sobre la interfaz cuando esta se encuentra conectada. Es usada para encontrar el NWID, ESSID o AP de la red que actualmente está en uso. La información reportada es la misma que la de iwconfig, sólo que iwgetid es más fácil para integrarse en scripts. Por defecto, este comando imprimirá en pantalla el ESSID del dispositivo, y si el dispositivo no tiene algún ESSID, entonces imprimirá su NWID.

- **-a,--ap:** despliega la dirección MAC del Access Point al que se encuentra conectada la interfaz.

- **-c,--channel:** proporciona información sobre el canal actual.
- **-f,--freq:** muestra la frecuencia a la que se encuentra operando la interfaz.
- **-m,--mode:** despliega el modo en que está trabajando la interfaz.
- **-p,--protocol:** muestra el nombre del protocolo que está utilizando la interfaz. Esto permite identificar todas las tarjetas que son compatibles también puede ser utilizado para comprobar Wireless apoyo a la extensión de la interfaz, ya que este es el único atributo que todos los conductores de apoyo Wireless Extensión tienen el mandato de apoyar.

2.2. Calidad de Servicio en Redes Inalámbricas

Cada día se expanden más las redes inalámbricas y con ello se van creando estándares que trabajan a ciertas velocidades, por ejemplo, el estándar 802.11b que trabaja a una velocidad de 11 Mbps. Sin embargo, como tales estándares trabajan a velocidades inferiores, a comparación de las redes cableadas, hay menos garantía de que se cumplan los requerimientos básicos de dichas aplicaciones, como las videoconferencias. En este caso, el requerimiento básico de una videoconferencia es que la comunicación sea lo más entendible que se pueda. De esta forma, mientras haya menor ancho de banda, menor será la garantía de bajo retardo en los paquetes de audio primordialmente.

Esto se debe a que se requiere que los paquetes no tengan un retardo que haga que la videoconferencia no sea entendible. Actualmente no existen garantías para calidad de servicio en WLAN, debido a que los estándares mismos no la implementan. Es por ello que han surgido propuestas, mencionadas en posteriores secciones, para tratar de garantizar este aspecto importante de las redes como lo es la QoS.

2.2.1. Limitaciones en redes inalámbricas para QoS

El medio inalámbrico como se ha mencionado es un medio muy endeble debido a diferentes factores que pueden degradar la calidad de servicio. Por tal motivo en este apartado se mencionarán los principales problemas que presentan este tipo de redes. El primer parámetro a analizar es la tasa de perdida. La tasa de pérdidas se puede considerar como un punto muy importante que afecta a la calidad de servicio en las redes. La tasa de pérdida se refiere al número de paquetes que se pierden al establecer una conexión entre dos puntos. Si la tasa de perdida aumenta la fluidez de la comunicación será seriamente afectada, lo cual implica que se pierda constante la comunicación entre los nodos.

El retardo extremo a extremo es otro punto a considerar en lo que calidad de servicio se refiere. Este término se refiere al tiempo que un paquete tarda en llegar desde el dispositivo fuente al destino. Este retardo en las redes inalámbricas se puede presentar por el número de saltos que un paquete debe de dar para llegar hasta su destino, esto en las redes de tipo ad-hoc, o por los cuellos de botella que se

pueden generar en el access point en las redes de tipo infraestructura. Los retardos en el trafico de tiempo real afectan considerablemente lo que es la calidad de servicio, ya que retardo es igual a pérdida de paquetes de audio, lo que es lo más importante que se debe de garantizar en este tipo de tráfico.

El jitter se puede definir como el cambio o variación en cuanto a la cantidad de latencia entre paquetes de datos que se reciben, en otras palabras de puede decir que es la diferencia entre el tiempo en que llega un paquete y el tiempo que se cree que llegara el paquete. La variación del jitter en las redes inalámbricas se puede ver afectada por problemas en el medio como ruido, viento, entre otros, al igual que el ancho de banda disponible con que cuenten los nodos entre los que se ha establecido la conexión.

Por otro lado, la tasa de transferencia (throughput) es un punto clave en lo que a calidad de servicio se refiere. Cuando se desea garantizar calidad de servicio, el contar con una tasa de transferencia constante es fundamental para garantizar que la información viajara de una manera rápida, ya que una disminución en esta generara que se tenga que dar más prioridad a un cierto tipo de trafico para garantizar calidad de servicio al tráfico que tenga mayor importación en un tiempo determinado.

Por todo lo mencionado anteriormente la calidad de servicio implica considerar muchos factores, y más aun si se refiere a calidad de servicio en redes inalámbricas. Es

por ello que el conocimiento de estos factores permite tener una mayor visión sobre los retos a los que se enfrentan este tipo de redes, para ofrecerles a las personas una buena calidad de servicio.

2.2.2. Estándar DCF y PCF

El acceso al medio inalámbrico se controla con la coordinación de funciones. El acceso CSMA/CD es suministrado por la Función de Coordinación Distribuida DCF. Si no requiere un servicio de no contienda, este es suministrado por la Función de Coordinación de Puntos (PCF), la cual está por encima del DCF. Los servicios sin contienda son suministrados únicamente en redes de infraestructura.

2.2.2.1 DCF

Es la base del mecanismo de acceso CSMA/CA. Al igual que Ethernet, es decir, si un nodo o estación desea transmitir antes debe escuchar el canal. Para evitar colisiones, las estaciones usan un tiempo de espera aleatorio después de cada paquete. En ciertas circunstancias DCF usa CTR/RTS para evitar la posibilidad de colisiones.

Si la estación detecta que el canal se encuentra libre durante un periodo llamado Espacio Inter-Tramas Distribuido (DCF, Distributed Inter-FrameSpace: DIFS), inicia su transmisión. El hecho que el canal esté libre en el DIFS no evita que dos o más estaciones intenten transmitir simultáneamente (a continuación del DIFS), por lo que existe la posibilidad de que se produzcan colisiones. Por otro lado, si al censar el canal,

éste se detecta como ocupado, la estación deberá retardar su transmisión hasta que el canal se encuentre libre por al menos un periodo DIFS o un Espacio Inter-Tramas Extendido (Extended Inter-Frame Space: EIFS). Existe la posibilidad que más de una estación detecte el canal ocupado y retarde su acceso, por lo que terminada la transmisión todas estas estaciones detectarán libre el canal por un periodo DIFS o EIFS y transmitirán en el mismo instante produciéndose una colisión. Para reducir esta probabilidad, el IEEE 802.11b utiliza el mecanismo de Evasión de Colisiones (*Collision Avoidance*: CA).

Como parte del mecanismo CA, antes de comenzar a transmitir se realiza un procedimiento de contienda (*backoff).* (Ver figura II.1)

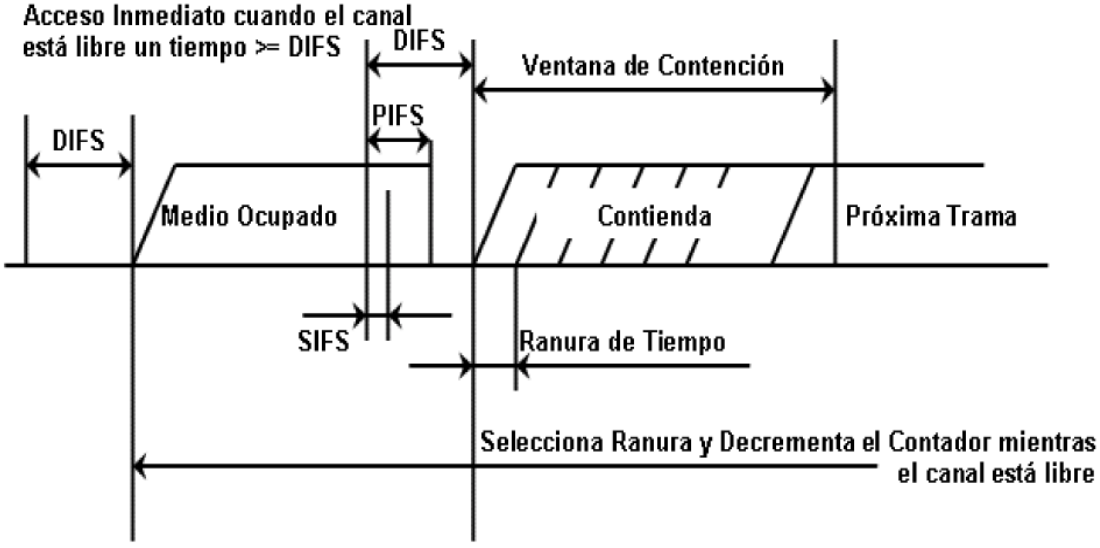


Figura II.1.- Mecanismo de Acceso Básico del Estándar IEEE 802.11

2.2.2.2 PCF

Proporciona servicios libres de contención, ciertas estaciones llamadas coordinadoras de puntos se usan para asegurar que el medio se está proporcionando sin contención. Los PCF residen en los Access points por lo que se usan solo en redes infraestructura. Los PCF permiten a las estaciones transmitir paquetes cada cierto intervalo y suelen tener mayor prioridad que los servicios basados en contienda.

Las estaciones acceden el medio inalámbrico coordinadas por un Punto de Coordinación (Point Coordinator: PC) que encuesta a los nodos si desean o no transmitir. El modo PCF tiene prioridad más alta que DCF, ya que puede comenzar a transmitir después de un tiempo más corto que DIFS; este tiempo es llamado Espacio Inter-Tramas de la Función de Coordinación Puntual (PCF Inter-Frame Space: PIFS). El tiempo se divide en periodos repetidos llamados supertramas (superframes), donde se alternan Periodos Libres de Contención (Contention Free Period: CFP) y Periodos de Contención (Contention Period: CP). Un CFP seguido de un CP es una super-trama. Durante el CFP, se usa la PCF para acezar el medio mientras que durante el CP se usa la DCF. Un super-frame comienza con un frame de alerta: beacon), independientemente si la PCF está o no activa.

El “beacon” es un paquete de administración que sirve para sincronizar los relojes locales en las estaciones. Para que las estaciones acecen el medio compartido, el PC genera mensajes de “beacon” a intervalos regulares, las cuales informan a las

estaciones bases que un nuevo CFP ha comenzado. El PC pregunta a una estación si desea transmitir. En caso de no obtener respuesta después de un PIFS, pregunta a otra estación o da por terminado el CFP.

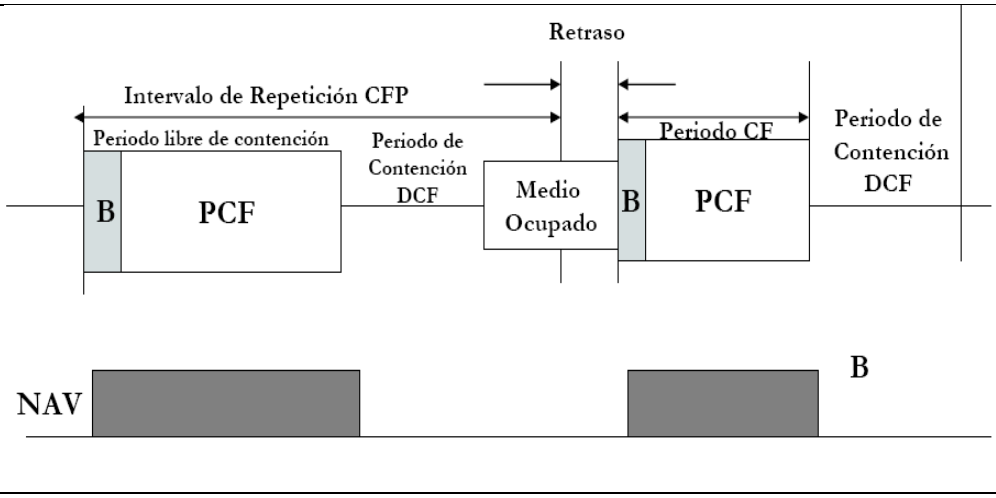


Figura II.2.- Función de Coordinación Puntual

A partir de lo anterior es claro que durante un CFP el canal no permanece desocupado por un periodo mayor a PIFS. El PC continúa preguntando a las otras estaciones hasta que el CFP expire. En esta forma de operación no se puede evitar totalmente que se produzcan colisiones, ya que, por ejemplo, si una estación escondida pierde la señal de alerta, y supone que aún se está transmitiendo en corresponde. Otro problema es que una estación se puede apropiar del canal injustamente, lo que va en detrimento de las otras estaciones.

El funcionamiento de PCF es totalmente compatible con el modo DCF, observándose que el funcionamiento es transparente para las estaciones.

2.3 Modelos de calidad de servicio

Con el motivo de garantizar que todos los servicios que se ofrecen en las redes inalámbricas cubran el requisito de calidad de servicio, han surgido una serie de modelos que tratan de cumplir con el objetivo mencionado. Unos modelos se enfocan a tratan de satisfacer esto basándose en garantizar que los datos que viajan en el medio lleguen correctamente entre el origen y el destino sin darle tanta importancia al tiempo que se necesita para esto, mientras que hay otros modelos que se preocupan por garantizar que los datos lleguen en un tiempo determinado, es decir, no se deben de exceder de cierto tiempo, lo anterior es de vital importancia en las videoconferencias, en donde el audio juega un papel primordial en donde el tiempo es realmente importante. A continuación se describirán los diferentes modelos para calidad de servicio, entre los que se abordaran son: SWAN, DRAP, INSIGNIA, el primer modelo a analizar será el SWAN.

2.3.1 Modelo SWAN

SWAN (Stateless Wireless Ad Hoc Networks) es considerado como un modelo red sin estado, capaz de diferenciar servicios en redes de tipo Ad-Hoc. Además de que es distribuido y basado en la capa MAC. Este modelo también es capaz de distinguir el tráfico best-effort del tiempo real. En la figura II.3 se observa entre que capas se coloca este modelo.

Para el funcionamiento de este modelo cada nodo debe de contar con una herramienta que sea capaz de clasificar los paquetes, para así poder hacer funcionar lo que se conoce como control de admisión, el cual está colocado entre la capa MAC e IP.

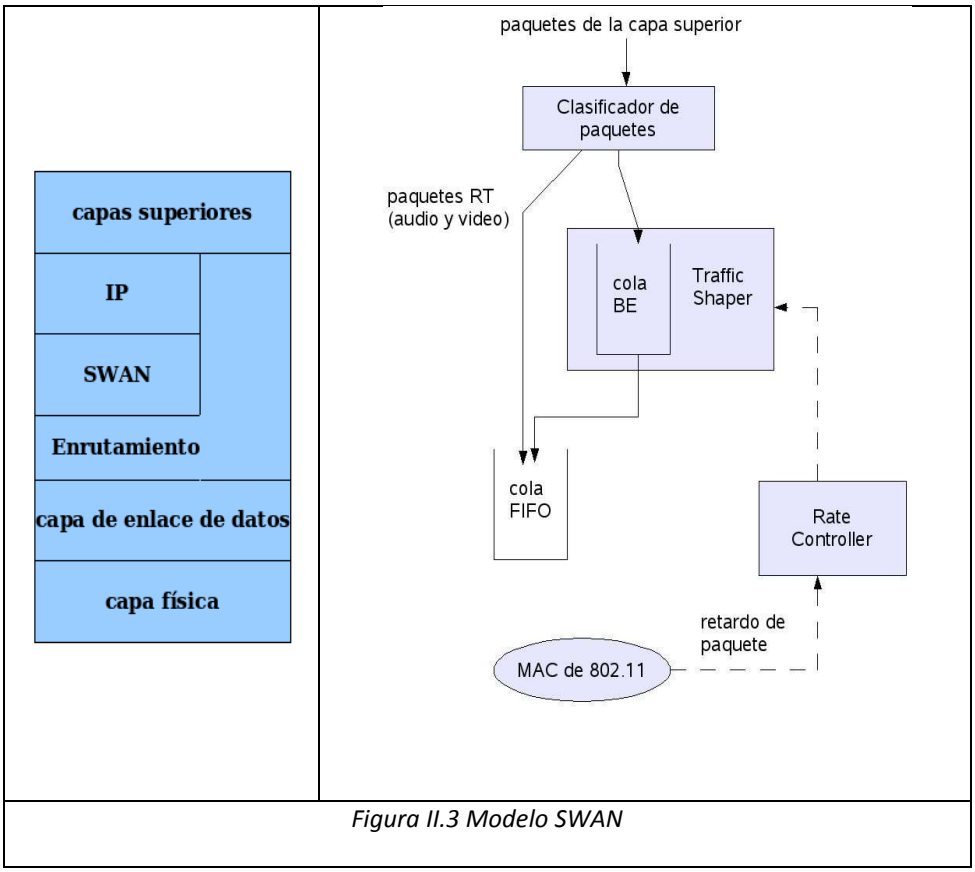


Figura II.3 Modelo SWAN

Para el funcionamiento de este modelo se modifica la cabecera del paquete IP, se modifica el campo DS (DiffServ), lo anterior se hace si el paquete es de tipo de tiempo real. Con la modificación anterior el tráfico que sea marcado como best-effort será colocado en el leaky-bucket en donde lo que se hace es disminuir el número de paquetes transmitidos, esto basándose en la aplicación del algoritmo AIMD (Additive Increase Multiplicative Decrease).

El algoritmo AIMD se basa en los retardos de los paquetes en la capa MAC, en donde al detectar retardos lo que hace este algoritmo es reducir la tasa de transmisión, en caso de no haber retardos realiza los cálculos necesarios para incrementar la tasa de transmisión.

Con la reducción se logra dar una mayor tasa de transferencia al tráfico de tiempo real. Como el tráfico de tiempo real y el best-effort comparten el medio hay que lograr que el tráfico total no sobrepase la tasa del umbral (throughput) para evitar que los paquetes sufran de un retardo excesivo, con lo que se evita saturación en el medio. El control de admisión de SWAN consiste en enviar paquetes a lo largo de toda la ruta, es decir, extremo a extremo, con el fin de determinar si el ancho de banda es el necesario para poder establecer otra sesión de tiempo real.

Por lo anterior se dice que SWAN no garantiza que todo el flujo de tráfico de tiempo real pueda satisfacer la calidad de servicio, ya que en ocasiones será necesario rechazar o readmitir sesiones ya establecidas.

2.3.2. Modelo INSIGNIA

Este modelo fue desarrollado por Antonio García-Macías, Franck Rousseau, Gilles Berger-Sabbatel y Leyla Toumi, Andrzej Duda. Insignia está basado en señalizaciones inband y un manejador de recursos de estados blandos (soft-state) necesario para soportar entornos altamente dinámicos con variaciones de tiempo en la

topología de red, conexiones de nodos y calidad de servicio punto a punto. Este modelo consiste en un mecanismo basado en DiffServ que pueda implementar calidad de servicio en entornos WLAN, teniendo como fundamentos la marcación y clasificación, un planificador de paquetes y un traffic shaper.

El funcionamiento de este modelo se basa en la creación de una arquitectura para calidad de servicio fundamentado en el modelo DiffServ y el uso de un traffic shaper para manejar los recursos de los nodos móviles. Los creadores de este modelo suponen que los host cuentan con una celda de comunicación que usa la capa MAC semejante a la del 802.11 de la IEEE. Además para el funcionamiento se considera que el ancho de banda disponible depende del número de hosts activos y la cantidad de tráfico que genera cada uno de estos.

Para controlar el ancho de banda útil en este modelo, se debe de tomar en cuenta la carga de tráfico y el número de host en la red, este control es necesario para los soft-state. La gestión global de Qos se realiza a través de las siguientes reglas:

- Reservar un determinado ancho de banda en todas las celdas.
- Reservar un determinado ancho de banda en las celdas de una determinada ruta.
- Reservar un determinado ancho de banda en las celdas en rutas con movilidad frecuente.
- Reservar un determinado ancho de banda en las celdas vecinas.

- Reservar un determinado ancho de banda en una celda.

Para el manejo de la señalización se necesita intercambiar información del protocolo Ipv6 entre todos los elementos de su arquitectura. La pila de Ipv6 cuenta con dos módulos: QoS y administrador de movilidad, y un mecanismo DiffServ. El protocolo de señalización de Insignia se basa en: in-band y out-of-band. El enfoque in-band permitirá tener un contador de cambios en las celdas. La señalización in-band consiste en insertar comandos dentro de los paquetes de datos transmitidos entre un host móvil y un router de acceso. Lo anterior puede solucionarse de dos formas: para comandos cortos como QOS_REQUEST, se puede codificar la información en una parte del campo de la etiqueta de flujo del Ipv6.

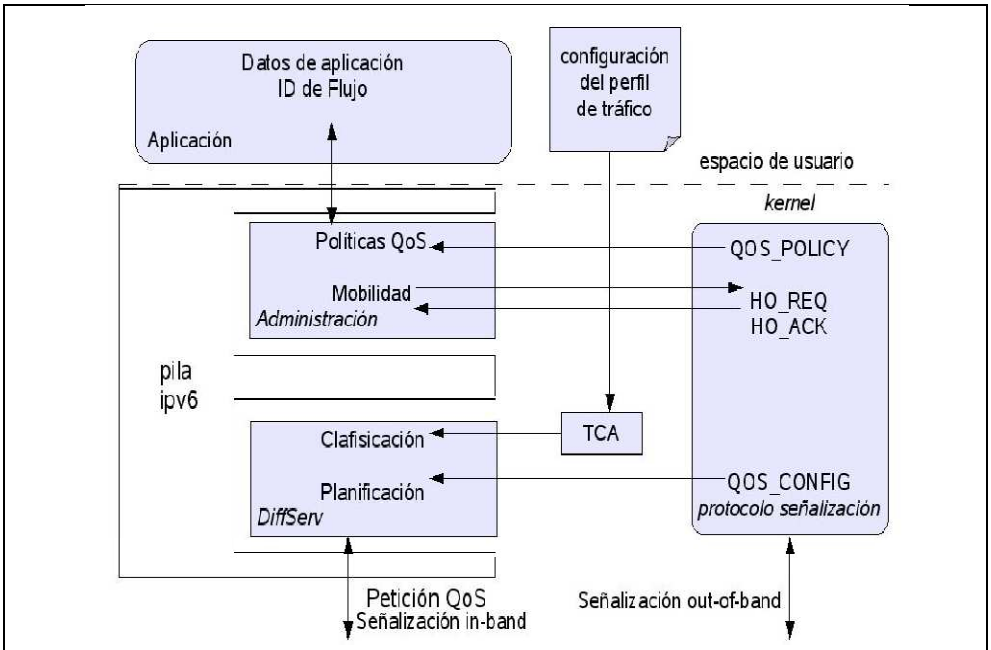


Figura II.4. Modelo Insignia

La otra solución es emplear extensiones de cabeceras. Los creadores de este modelo proponen la definición de un nuevo tipo de ICMPv6 que ya contenga los comandos de señalización.

2.4. Modelo DRAP

El mecanismo Dual-Queue Rate-Controlled Access Point, o DRAP por sus siglas en inglés, es otro modelo más que propone garantizar la calidad de servicio sin hacer alguna modificación a las especificaciones del estándar 802.11; por el contrario, propone implementar una capa intermedia independiente de la capa MAC y la IP, y así crear una especie de middleware a través del kernel del sistema operativo Linux, aprovechando las ventajas que este sistema de software libre ofrece. El término middleware hace referencia precisamente a esta capa intermedia, ya que tiene como significado "software de en medio".

DRAP fue desarrollado en previos trabajos por Rubén Álvaro González Benítez y Lloren Cerdà . Este algoritmo está desarrollado para redes de tipo infraestructura, de ahí las dos últimas palabras (Access Point) hacen pensar que el algoritmo se implantará en una computadora personal que fungirá como punto de acceso. Al igual que SWAN distingue el tráfico de tiempo real (audio y vídeo) del tráfico best-effort.

DRAP es un esquema que garantiza bajo retardo y baja pérdida de paquetes de audio, mientras se limita la tasa del tráfico de vídeo y TCP. Está enfocado a funcionar en ambientes de videoconferencias (figura II.5).

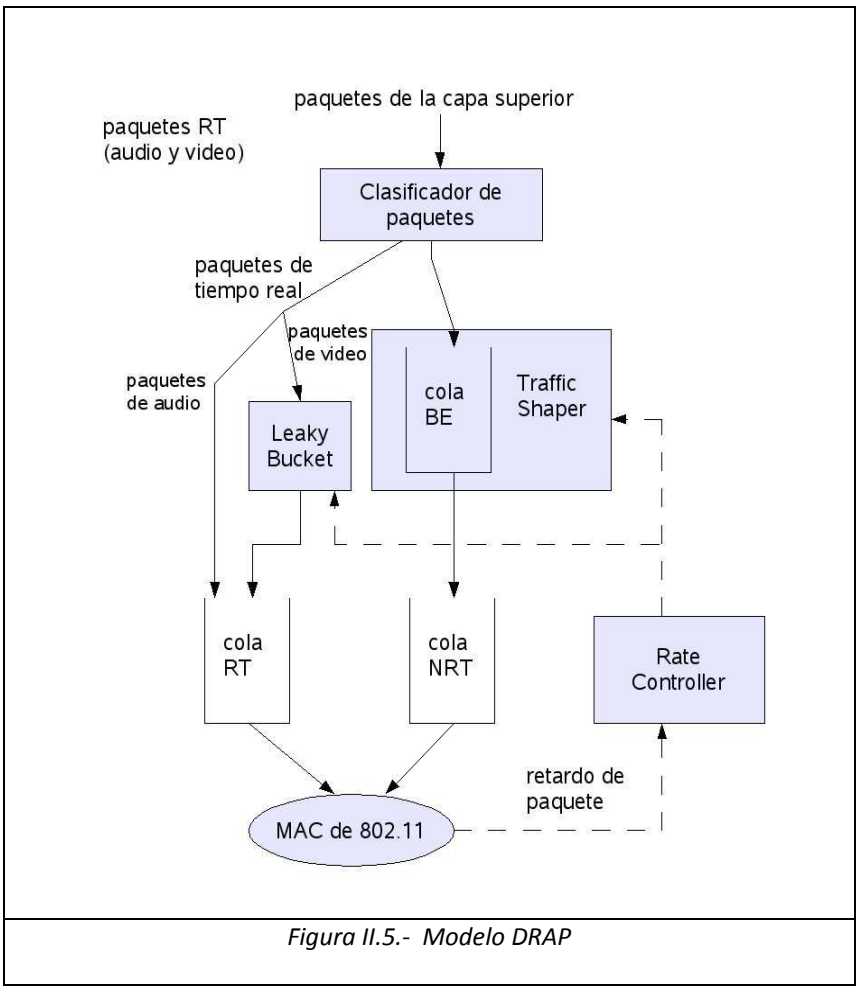


Figura II.5.- Modelo DRAP

El funcionamiento de este algoritmo se explica a continuación. Se posee de un clasificador de paquetes, el cual identifica si el paquete es best-effort (BE) ó de tiempo real (RT); el paquete de tráfico best-effort es enviado al traffic shaper, en caso contrario, es decir, que el paquete sea de tiempo real, el de audio es enviado directamente a la cola del tráfico de tiempo real, mientras que el de vídeo es trasladado al leaky bucket. Este último es el encargado de descartar los paquetes en

caso de encontrase tráfico excesivo, para esto se aplica el algoritmo AIMD (Additive Increase Multiplicative Decrease). Mientras tanto el traffic shaper es el encargado de controlar la tasa de transferencia de los paquetes besteffort.

Por lo tanto, la prioridad más alta de tráfico para ser atendido sería para el de tiempo real, donde el audio al pasar directo a la cola tendría una prioridad mayor que el vídeo, y posteriormente el best-effort.

Todo se realiza con la finalidad de que los paquetes de audio tengan un seguimiento en la red sin retardo, ya que si lo hubiera en ellos, la comunicación sería muy poco entendible.

Para efectos de este trabajo citaremos los aspectos más importantes para el desempeño de este algoritmo.

2.4.1. Traffic shaping

El traffic shaping intenta controlar el tráfico en redes de ordenadores para así lograr optimizar o garantizar el rendimiento, baja latencia, y/o un ancho de banda determinado retrasando paquetes. La catalogación de tráfico propone conceptos de clasificación, disciplinas de colas, imposición de políticas, administración de congestión, calidad de servicio (QoS) y regulación. Esto incluye decisiones en las cuales los paquetes se aceptan y en qué tasa de transferencia de entrada de una

interfaz además de determinar cuáles paquetes transmitir y en qué orden y a qué velocidad. En la mayoría de los casos, el traffic control consiste en una cola simple, en la cual recolecta los paquetes entrantes y los va transmitiendo mientras el dispositivo tenga la capacidad de hacerlo.

Por otra parte, esto consiste en una práctica utilizada por diversos ISPs para no sobrepasar sus capacidades de servicio, desde hace mucho tiempo el control de tráfico está soportado por distintas tecnologías de transporte en forma nativa lo que permite la clasificación y priorización de tráficos como VoIP en nuestro caso el tráfico multimedia por sobre otros de mejor tolerancia al retraso y a jitter.

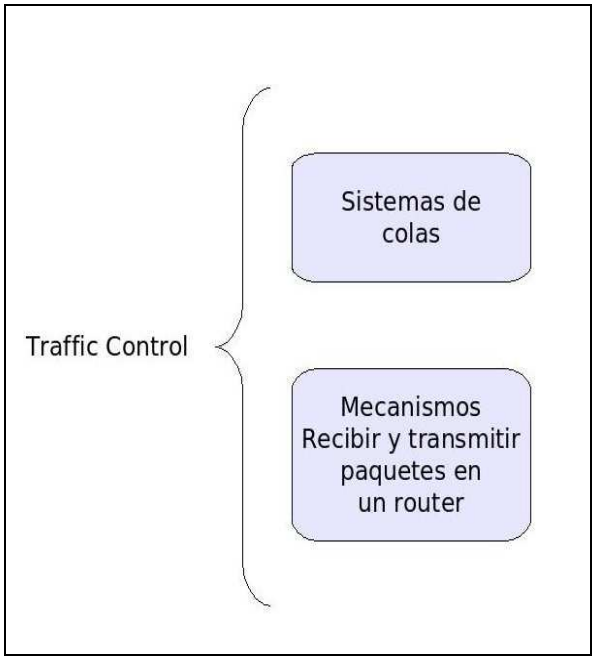


Figura II.6.- Elementos de un traffic control

A continuación se describirán cada uno de los elementos y conceptos que involucra un sistema de control de tráfico (figura II.6).

Durante el proceso de control de tráfico, los paquetes atraviesan diferentes elementos cada uno de estos elementos tiene sus funciones y mecanismos específicos (clasificar, rechazar, ordenar, etc.) que combinados dar lugar al control de tráfico.

2.4.2. Elementos de Traffic shaping

2.4.2.1. Shaping

En pocas palabras, shaper o shaping significa retardar paquetes para satisfacer una tasa de transferencia deseada. El shaping es el mecanismo por el cual los paquetes son retrasados antes de la transmisión en una cola de salida para satisfacer una tasa de salida deseada. Esto es uno de los deseos más comunes de los usuarios, que buscan soluciones de control de ancho de banda. El shaper intenta limitar o racionar el tráfico para no exceder una tasa (medida frecuentemente en paquetes por segundo o bits/bytes por segundo). Una de las ventajas de limitar el ancho de banda es la habilidad para controlar la latencia de los paquetes. El mecanismo relacionado con el shaping es típicamente un mecanismo de token y bucket.

2.4.2.2. Classifying

Classifying significa clasificar, lo cual es una traducción exacta a su concepto en inglés. Un clasificador es el mecanismo por el cual, los paquetes son separados

para diferentes tratamientos, como podría ser separarlos en diferentes colas de salida. Durante el proceso de aceptación, enrutamiento y transmisión de un paquete, un dispositivo de red puede clasificar un paquete de diferentes formas. La clasificación puede incluir el marcado de un paquete, el cual usualmente sucede en los límites de una red, esto es, en la salida o en la entrada; bajo un sencillo control o clasificación que puede ocurrir en cada salto individualmente. El concepto de un clasificador puede verse en la figura II.7.

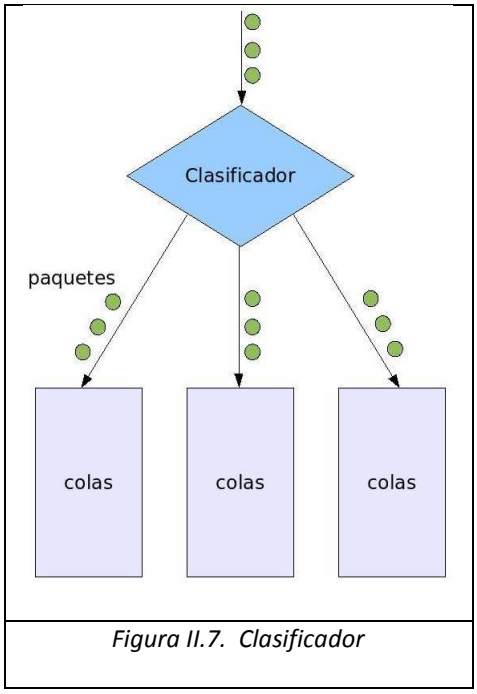


Figura II.7. Clasificador

2.4.2.3. Scheduling

Un scheduler, que significa programador, tiene la propiedad de ordenar o reordenar paquetes que vayan a salir. Es un mecanismo por el cual los paquetes son ordenados entre la entrada y la salida de una cola particular. El scheduler más común es el denominado FIFO. Desde una perspectiva general, cualquier conjunto de

mecanismo de control de tráfico sobre una cola de salida puede ser considerado como un scheduler, debido a que los paquetes son ordenados para su salida. Un algoritmo de cola "justo", por así llamarlo, intenta prevenir que algún cliente o flujo simple dominen la red.

2.4.2.4. Dropping

Esta acción lo que hace es descartar un paquete completo, flujo o clasificación. Forma parte fundamental del traffic shaper cuando lo que se desea es descartar paquetes que no son tan importantes.

2.4.2.5. Policing

Los policiers miden y limitan el tráfico en una cola particular. La palabra policing hace referencia a la acción de aplicar políticas; el policing, como elemento de un traffic control, es simplemente un mecanismo por el cual el tráfico puede ser limitado. Es más frecuentemente usado en el borde de una red para asegurar que un punto no está consumiendo más que el ancho de banda asignado. Un policier aceptará tráfico a una cierta tasa de transferencia y entonces realizará una acción al tráfico que exceda esa tasa. Una posible solución es borrar ese tráfico que se excede, aunque en el enfoque más conservador, el tráfico debe volver a clasificarse en lugar de borrarlo. Un policier es una respuesta de sí o no respecto de la tasa de transferencia a la cual el tráfico entra en la cola. Si el paquete ingresa a la cola

con una tasa igual o por debajo de lo permitido, se toma una acción (se permite meterlo a la cola); si la tasa está por encima, se toma alguna otra acción. Aunque internamente un mecanismo de tokens no tiene la capacidad de retardar un paquete como lo hace el mecanismo del shaping. En la figura II.8 se observa el trabajo que realiza un policier.

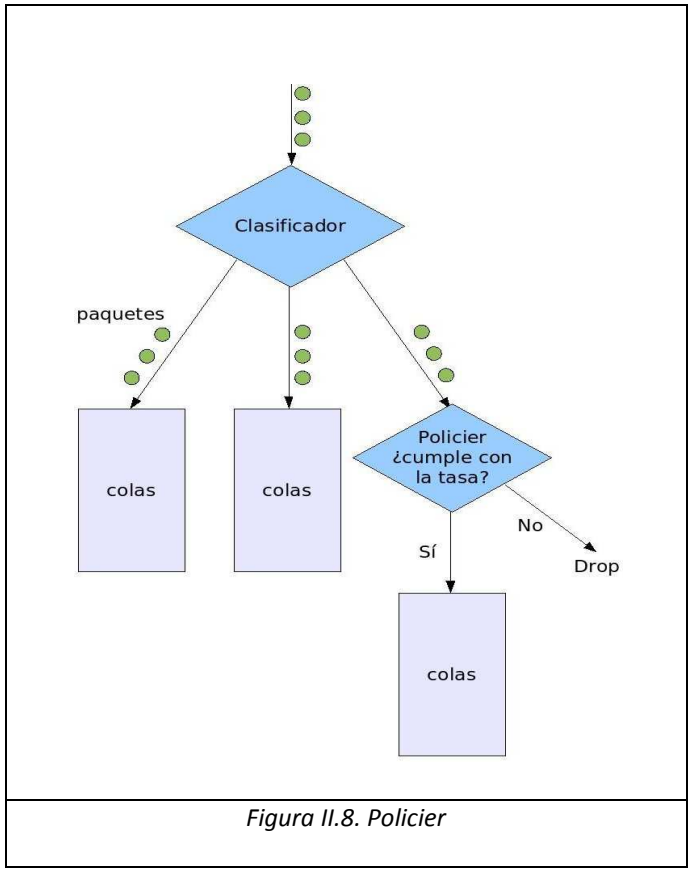


Figura II.8. Policier

2.4.2.6. Marking

Es un mecanismo por el cual un paquete es alterado. El marking o marcado entra en la modificación de un paquete de red. Existe un campo denominado Differentiated Services Code Point (DSCP por sus siglas en inglés) dentro de la estructura del paquete que ayuda a diferenciar el tipo de servicio o flujo de dato que se está enviando.

Lamentablemente, la mayoría de estos campos no vienen marcados con el tipo al que pertenecen; es por ello que se necesita de un mecanismo como este, para diferenciar los servicios.

2.4.3. Herramientas de Traffic Shaping

A partir de las versiones 2.2 del kernel de Linux, se incorpora un nuevo subsistema de red. Debido a la necesidad de nuevas facilidades en el ámbito de la configuración de redes, se fueron añadiendo parches a la implementación de la pila de protocolos TCP/IP que hicieron que con el paso del tiempo fuera inmanejable. Este nuevo diseño, se facilitaron tareas como el routing, firewall y código de clasificación de tráfico.

La herramienta iproute2 es la encargada de este nuevo paquete. Este fue creado por Alexey Kuznestov y consiste es una colección de utilidades para controlar el tráfico en Linux. Dentro de estas utilidades se pueden encontrar ejecutables como ip y tc. El ejecutable ip es el principal, y tiene diferentes funciones de gestión de rutas e interfaces. El ejecutable tc (Traffic Control) es el que permite controlar el tráfico con implementaciones de QoS.

2.4.3.1. Iptables

Iptables se considera el firewall de Linux, este firewall está asociado directamente con el kernel. Para el funcionamiento de iptables se necesita que el kernel cuente con

el soporte necesario, la integración de iptables se ha realizado a partir del kernel 2.4. En el kernel, los cortafuegos que se utilizaban eran ipchains (kernel 2.2) e ipfwadm (kernel 2.0) el cual estaba basado en ipfw de BSD (por sus siglas en ingles, Berkeley Software Distribution).

Iptables está diseñado para realizar las siguientes operaciones: filtrado de paquetes (packet filtering), seguimiento de conexiones (connection tracking) y traducciones de direcciones de red (Network Address Translation). Gracias a las operaciones anteriores, iptables permite definir reglas para decidir qué hacer con los paquetes que viajan por la red. Cada regla es agrupada en una cadena, estas a su vez son agrupadas en tablas. Las cadenas permiten tomar decisiones sobre los paquetes ip, las cadenas permitidas en iptables son 3: entrada (INTPUT), salida (OUTPUT) y reenvío (FORWARD). Mientras que las tablas también está formada por 3 elementos (Ver figura II.9).

2.4.3.1.1. Tabla FILTER: esta tabla toma decide si deja o no pasar el paquete. Todo paquete pasa por esta tabla. Las cadenas que conforman esta tabla son: INTPUT, OUTPUT y FORWARD.

2.4.3.1.2. Tabla NAT: es la encargada de configurar las reglas de reescritura de puertos o direcciones de paquetes. Las cadenas que conforman esta tabla son: PREROUTING, POSTROUTING y OUTPUT.

2.4.3.1.3. Tabla MANGLE: permite cambiar parámetros del paquete ip, por ejemplo, permite cambiar el valor del campo TOS. Esta contienes las seis cadenas de las tablas anteriores: INTPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING y OUTPUT.

Esta tabla es muy importante para la realización del presente trabajo ya que en ella se basa para re direccionar los paquetes según ciertos criterios que explicaremos más adelante.

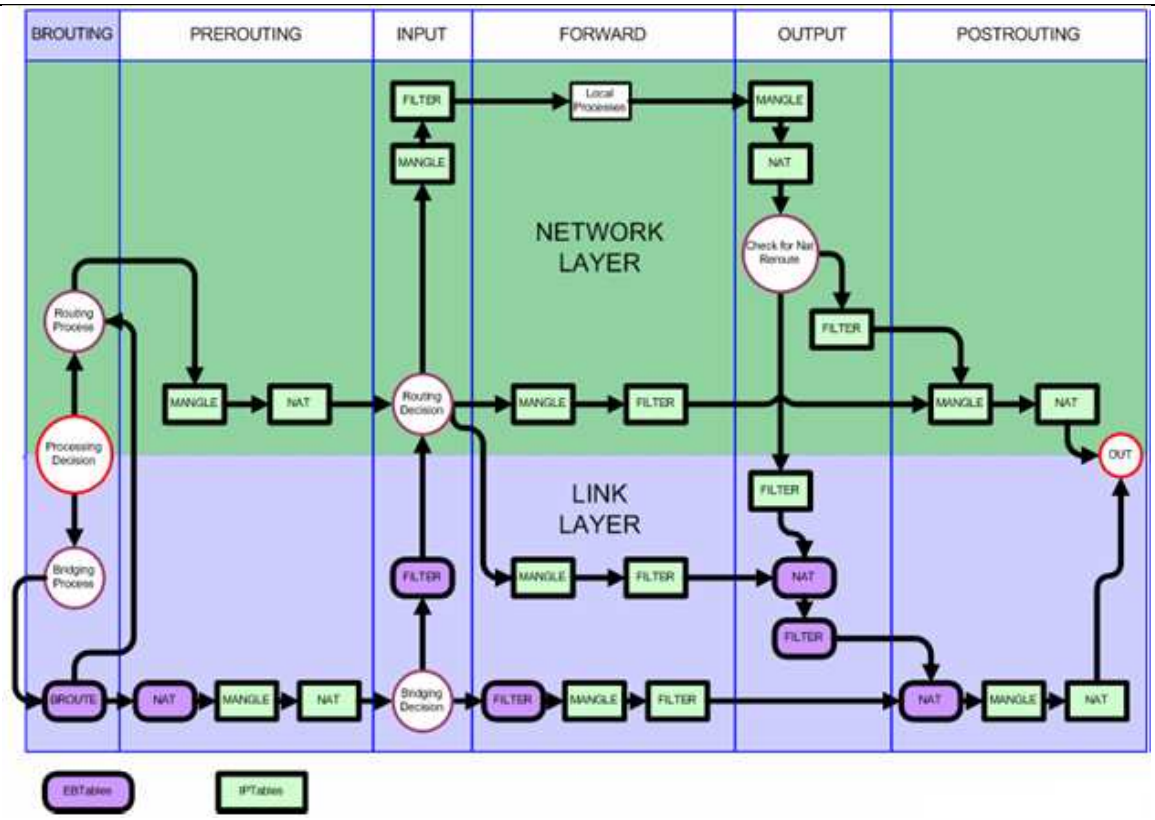


Figura II.9.- Flujo iptables

2.4.3.2. Iproute2 y tc

La utilidad tc es una de las aplicaciones clave en este proyecto de tesis. Por ello, es muy importante hablar primero de qué se trata y de dónde proviene. Iproute2

es una colección de utilidades para el control de las redes TCP/IP, administrar sus interfaces y para el control de tráfico en Linux. Fue desarrollada originalmente por Alexey Kuznetsov y es bien conocida por su implementación de QoS en el kernel de Linux. Este paquete reemplaza completamente las funcionalidades presentes en ifconfig, route, y arp y las extiende llegando a tener características similares a las provistas por dispositivos exclusivamente dedicados al enrutamiento y control de tráfico. Esta suite se instala por defecto en muchos sistemas GNU/Linux, como Debian.

Dentro de los beneficios y funciones que iproute2 proporciona son:

- Calidad de Servicio (QoS): esto se logra, con disciplinas de cola, priorizando los flujos del tráfico, colocando cada uno de clases, filtros y políticas diferentes.
- Mejor manejo de tablas de enrutamiento: se pueden tener varias puertas de enlace.
- Balanceo de carga: dando un equilibrio a las interfaces de red existentes en una computadora.
- Posibilidad de túneles: es una opción muy potente a la hora de establecer los datos de forma segura.

Iproute2 está usualmente liberado en un paquete llamado iproute o iproute2 y consta de varias herramientas, de las cuales las más importantes son *ip* y *tc.* La herramienta ip controla la configuración de los protocolos IPv4 e IPv6 y tc es utilizado para el control de tráfico.

El programa, tc está a nivel de usuario y puede ser usado para crear y asociar disciplinas de cola con dispositivos de red. Es usado para establecer varias clases de colas y asociar clases a cada una de esas colas; términos que se explicarán a continuación. Sus parámetros son muchos y variados, por lo que se pueden realizar ajustes muy específicos para cada acción que se quiera tomar.

2.4.3.3. Queues

Las colas son quizá el elemento clave para el traffic control. Una cola es un "lugar" que contiene un número finito de elementos, esperando para alguna acción o servicio.

Imagine la posición de una cola de la oficina de correo contra una cola en una sala de emergencia. Ambos tienen elementos en la cola que necesitan ser atendidos de alguna manera, pero tienen diferentes estrategias. La oficina de correo típicamente usa una estrategia del primero que entra, primero que sale. Los clientes son servidos en el orden que llegan a la cola. Del otro lado, esto es una estrategia no deseable para la administración de una sala de emergencia. Alguien en una condición crítica requiere atención urgente a pesar del orden de llegada a la cola. Si hay diez personas en la cola y sólo hay recursos para atender a dos, lo primero que hay que hacer es ordenar la cola en clases de acuerdo a la urgencia. Entonces se vacían las colas de acuerdo a la prioridad de las diferentes clases.

En redes, una cola es el lugar donde los paquetes esperan para ser transmitidos por el hardware, en este caso la interfaz de red. En el modelo más simple, los paquetes

son transmitidos en el concepto: el primero que llega es el primero que sale; la conocida cola First Input First Output (FIFO por sus siglas en inglés). Sin otros mecanismos, una cola no ofrecerá ninguna promesa de control de tráfico. Hay dos acciones primordiales en una cola; cualquier paquete que entre en una cola se dice que es puesto en la cola, o lo que es lo mismo "enqueued", en inglés. Para eliminar un paquete de la cola, se utiliza la palabra "dequeue". Una cola se vuelve más interesante cuando se complementa con otros mecanismos, los cuales pueden retardar paquetes, reordenarlos, borrarlos y darles prioridad en múltiples colas. Una cola puede también usar subcolas.

Desde la perspectiva de la capa de aplicación, un paquete simplemente es metido a la cola para su transmisión y la manera en la cual son sacados y transmitidos es irrelevante para la capa más alta. Por tanto, para esta capa el sistema de control de tráfico puede verse como una cola simple.

2.4.3.4. Flows

Un flujo es una conexión o conversación distinta entre dos host. Cualquier conjunto único de paquetes entre dos hosts puede ser considerado como un flujo.

Bajo el protocolo Transmission Control Protocol (TCP, por sus siglas en inglés), el concepto de una conexión con una dirección IP y puerto fuente y destino representa un flujo. En traffic control frecuentemente separa el tráfico en clases o

flujos los cuales pueden ser sumados y transmitidos como un flujo totalizado. Como una alternativa, los mecanismos pueden intentar dividir el ancho de banda equitativamente basado en flujos individuales. Los flujos se vuelven importantes cuando se intenta dividir el ancho de banda entre un conjunto de flujos, especialmente cuando ciertas aplicaciones deliberadamente generan un número grande de flujos.

2.4.3.5. Tokens y Buckets

Para controlar la tasa de transferencia de los paquetes que salen, una implementación puede contar el número de paquetes o bytes sacados, aunque esto requiera un complejo uso de temporizadores y medidas para limitar de forma exacta. En lugar del uso de estos mecanismos complejos, un método es generar tokens en una tasa deseada y sólo quitar paquetes o bytes si un token está disponible. Para entender lo que significa token, se pondrá el siguiente ejemplo. Considerando la analogía de un parque de diversiones con una cola de personas esperando para entrar y subirse a los carros de carreras que atraviesan una pista de tamaño fijo.

Los carros llegan a la cabeza de la cola en una tasa de tiempo fija, cada persona debe esperar a que un carro esté disponible; el carro es una analogía a un token y la persona es la analogía a un paquete. Sólo un cierto número de personas pueden experimentar el carro en un periodo de tiempo particular. Ahora, una cola vacía y una larga fila de

carro esperando a la gente. Si un número grande de personas ingresaran a la cola juntas, la mayoría o quizá todas podría subirse a los carros y correr al mismo tiempo por la pista.

El número de carros disponibles sería un bucket, que significa literalmente cubeta. Un bucket contiene un número de tokens y puede usar todos los tokens en ella sin considerar el paso del tiempo. También es importante mencionar que cada bucket tiene un límite de tokens, y más adelante se entrará en detalle en estos conceptos cuando se hable de las disciplinas de cola.

En el caso que una cola no necesite tokens inmediatamente, los tokens pueden ser recogidos hasta que sean necesitados. Recoger tokens indefinidamente, negaría cualquier beneficio de shaping (o darle forma al tráfico), así que los tokens son recogidos hasta que un cierto número de ellos hayan sido alcanzados. Ahora, la cola tiene tokens disponibles para un gran número de paquetes o bytes, los cuales necesitan ser quitados de la cola.

Esos tokens intangibles son almacenados en una bucket intangible y el número de tokens que puede ser almacenado depende del tamaño del bucket. Esto quiere decir que una bucket llena de tokens, pueda estar disponible en cualquier instante.

En resumen, los tokens son generados en una tasa y un valor máximo de tokens de una bucket puede ser recogido. Los conceptos de tokens y buckets están íntimamente relacionados y son usados en las disciplinas de cola Token Bucket

Filter (TBF, por sus siglas en inglés) y Hierarchical Token Bucket (HTB), las cuales se discutirán más adelante.

2.4.3. Disciplinas de cola

El sistema operativo GNU/Linux nos ofrece una gran variedad de herramientas para controlar el tráfico que pasa por la red. Esto se refleja en las disciplinas de colas que nos permite implementar para indicarle a la tarjeta de red la forma en que se enviarán los datos, dando una prioridad mayor a cierto tipo de paquetes, esto de acuerdo a las necesidades que se tengan.

2.4.3.1. Qdisc

Correlacionando, una cola es un scheduler. Aquí se intenta introducir respecto de los mecanismos de colas. La palabra qdisc es el acrónimo de las palabras queue discipline (disciplina de cola). Cada interfaz de salida necesita de un scheduler de algún tipo y el scheduler por defecto en un sistema GNU/Linux es una cola FIFO. Otras qdisc disponibles bajo Linux reordenarán los paquetes que ingresan en la cola de acuerdo con las reglas del scheduler. Existen dos tipos de qdisc, las classful (disciplinas con clases) y las classless (disciplinas sin clases). Una cola classful puede contener clases (que se explicarán a continuación) y proveer un identificador (handle) a la cual atarle filtros. En otras palabras, una disciplina de este tipo tiene la flexibilidad de tener filtros y disciplinas de cola hijas asociadas a ella. Generalmente

contienen al menos una hija, debido a que si se utiliza sin descendientes, suele consumir ciclos y otros recursos del sistema sin ningún beneficio. Una qdisc sin clase o classless no contiene clases o filtros asociadas a ella; debido a que no tiene hijos, no existe la utilidad de clasificación en este tipo de colas.

Una fuente de confusión es el uso de los términos root e ingress qdisc. En realidad estos no son disciplinas de cola, pero hacen referencia al lugar donde las estructuras del traffic control son colocadas: egress (egreso, cuando el tráfico sale de la interfaz) e ingress (ingreso, cuando el tráfico ingresa a la interfaz). Cada interfaz contiene ambos lados (egreso e ingreso) y el tráfico tiene que atravesar forzosamente los dos; la más común y principal de las colas es la de egreso, conocida como cola raíz o root qdisc y contiene cualquier disciplina de cola con posibilidad de tener clases y filtros. Para el tráfico que es aceptado en una interfaz, la cola de ingreso actúa sobre él, con la limitación de que no permite crear clases hijos y sólo existe como un objeto sobre el cual un filtro puede ser atado. En resumen, se puede hacer mucho más con una qdisc de egreso porque contiene una cola real y el poder de un sistema de control de tráfico.

2.4.3.2. Class

Una clase sólo existe dentro de una disciplina classful y es un mecanismo a través del cual se puede realizar el denominado shaping o controlar la tasa de cierto tipo de tráfico. Las clases son inmensamente flexibles y pueden siempre contener múltiples clases hijos o uno solo. Desde luego que no hay ninguna regla que prohíba a

una clase que contenga una misma clase classful, lo cual facilita tremendamente los escenarios complejos de traffic control. Cualquier clase puede tener un número arbitrario de filtros atados a ella, los cuales permiten la selección de una clase hija o el uso de un filtro para volver a clasificar o descartar el tráfico que ingresa a una clase particular.

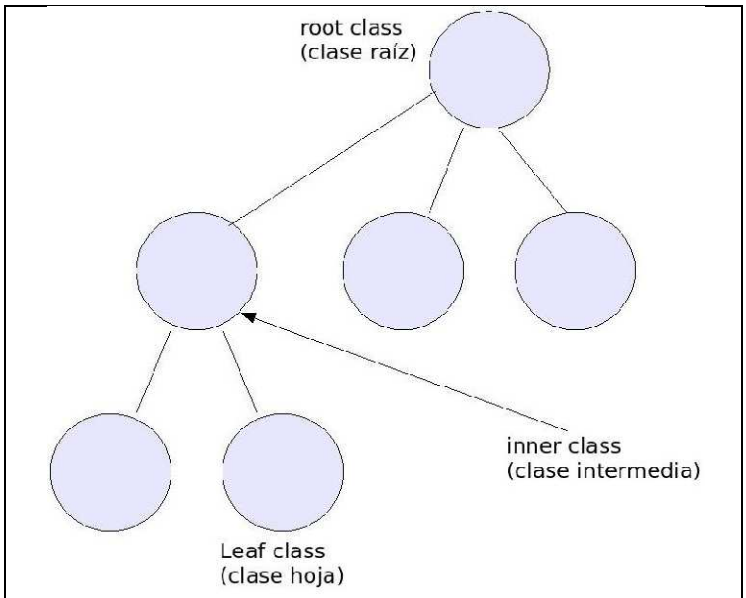


Figura II.10.- Tipos de clases

Una clase hoja (leaf class), es una clase terminal en una disciplina de cola que contiene una disciplina y nunca contendrá una clase hijo, cualquier clase que contenga una clase hija es llamada una clase interna (inner class). Tal interpretación se muestra en la figura II.10.

2.4.3.3. Filter

El filtro es el más complejo componente en el sistema de control de tráfico de Linux. El filtro provee un mecanismo conveniente para juntar varios de los elementos clave del traffic control.

El papel más simple y obvio de un filtro es clasificar paquetes. Los filtros de Linux permiten a los usuarios clasificar paquetes dentro de una cola de salida con diferentes o un único filtro.

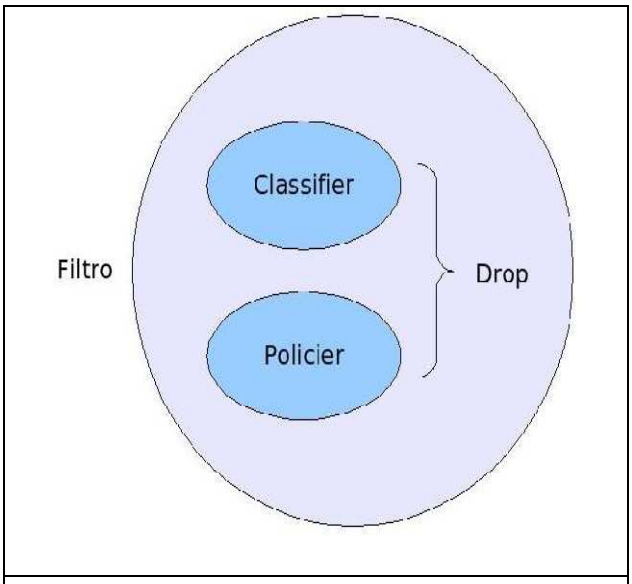


Figura II.11.- Filtro

Un filtro debe contener un classifier al menos y puede tener políticas (policier). Los filtros pueden ser atados a disciplinas de cola classful o classless, sin embargo, el paquete que se está metiendo a la cola siempre ingresa a la cola root primero. Después de que el filtro atado a la cola root haya sido atravesado, el paquete puede

ser dirigido a alguna subclase (la cual puede tener su propio filtro). En la figura II.11 se ve el contenido de un filtro.

2.4.3.4. Handle

Cada clase y classful requiere un único identificador dentro de la estructura del traffic control. Este identificador único es conocido como un "handle" y tiene dos números que lo constituyen; esos números pueden ser asignados arbitrariamente por el usuario de acuerdo con las siguientes reglas:

- Número mayor (major): este parámetro está completamente libre de significado para el kernel. El usuario puede usar un esquema de números arbitrarios, sin embargo todos los objetos en la estructura del traffic control con el mismo padre deben compartir un número major. Las convenciones para este número comienzan en 1 (uno) para objetos atados a la disciplina raíz (root).

- Número menor (minor): este parámetro identifica ambiguamente al objeto como una qdisc si el número minor es cero. Cualquier otro valor identifica al objeto como a una clase. Todas las clases comparten un padre deben tener números únicos de este tipo. El handle especial ffff:0 esté reservado para la qdisc de ingreso.

A continuación se explicará la disciplina de cola con clase PRIO y HTB que son las utilizadas en el desempeño de este algoritmo

2.4.4. PRIO

Esta disciplina no hace ajustes referentes a las tasas de transmisión, lo que hace es subdividir el tráfico en base a los filtros que se hayan creado, para darles prioridades de transmisión a cada uno de estos tipos. Esta disciplina de cola consta de tres clases por defecto. Cada una de estas clases es una cola FIFO, pero estas pueden ser sustituidas por otras que más se ajusten a las necesidades que se requieran resolver.

Cuando ya hay paquetes en la cola, los paquetes que se quitaron de la cola primero son los de la banda. Si la banda 1 contiene paquetes y la banda 2 también quiere sacar paquetes de la cola, esta no podrá realizar transmisión alguna hasta que la banda 1 esté vacía. Lo mismo ocurre cuando la banda 3 es la quiere transmitir, lo primero que hace es verificar que la banda uno esté vacía, de ser así ahora verifica que la banda 2 no tenga paquetes, si las dos cola están vacías entonces si puede sacar los paquetes, de lo contrario debe de esperarse hasta que las dos bandas se queden vacías. Los parámetros con los que cuenta esta disciplina de colas son los siguientes:

- Bands: indica el número de bandas a crear. Donde cada banda es una clase.

- Priomap: indica a qué banda se debe ir el tipo de tráfico que llega a la cola de acuerdo a la prioridad que se haya asignado. La tabla II.1 muestra las bandas a las que se va un paquete de acuerdo a las prioridades asignadas al campo TOS.

Tabla II.1 *Prioridades y bandas de la cola PRIO*

Band	TOS	Bits	Means	Linux Priority
1	0x0	0	Normal Service	0 Best Effort
2	0x2	1	Minimize Monetary Cost (mmc)	1 Filler
1	0x4	2	Maximize Reliability (mr)	0 Best Effort
1	0x6	3	Mmc+mr	0 Best Effort
2	0x8	4	Maximize Throughput (mt)	2 Bulk
2	0xa	5	mmc+mt	2 Bulk
2	0xc	6	mr+mt	2 Bulk
2	0xe	7	mmc+mr+mt	2 Bulk
0	0x10	8	Minimize Delay (md)	6 Interactive
0	0x12	9	mmc+md	6 Interactive
0	0x14	10	mr+md	6 Interactive
0	0x16	11	mmc+mr+md	6 Interactive
1	0x18	12	mt+md	4 Int. Bulk
1	0x1a	13	mmc+mt+md	4 Int. Bulk
1	0x1c	14	mr+mt+md	4 Int. Bulk
1	0x1e	15	mmc+mr+mt+md	4 Int. Bulk

Fuente: www.openbsd.org

2.4.5. HTB (Hierarchical Token Bucket)

El funcionamiento de esta disciplina permite hacer un ajuste (Shaping) a la velocidad a la que transmite, se basa en determinar y tener controlado el tiempo en que el enlace está sin transmitir datos, para que sólo se reduzca el ancho de banda cuando sea necesario. Pero no considera el tipo en el que el enlace está sin transmitir datos, sino que se basa en clases con Token Bucket Filter. Para asignar prioridades al tipo de tráfico que llegan a la cola es necesaria la implementación de filtros que indiquen a qué clases hijas se debe de dirigir el tráfico. Los filtros se pueden realizar mediante la opción que proporciona tc, que se explicará después, o mediante la marcación de paquetes con iptables. Se pueden realizar filtros por direcciones ip, puertos origen y destinos, el campo TOS (Type of Service), entre otros.

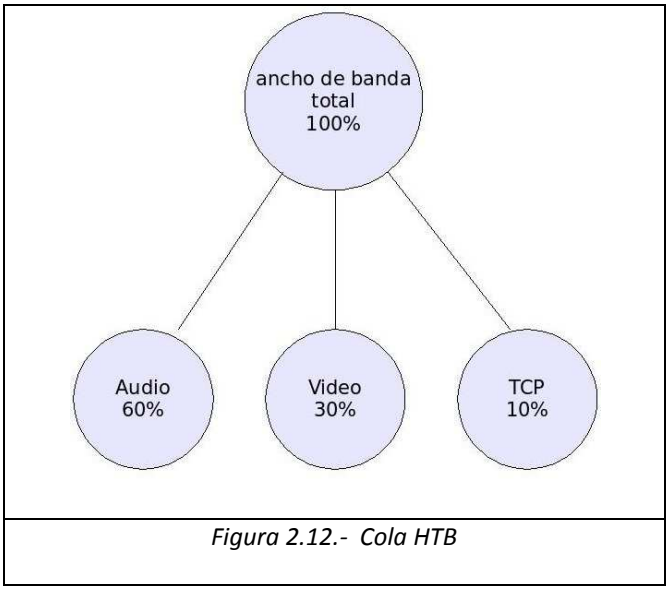


Figura 2.12.- Cola HTB

Esta disciplina de cola se preocupa de garantizar un ancho de banda al tipo de tráfico definido por clase.

Es por eso que es necesario tener un buen análisis sobre las prioridades y el ancho de banda físico con el que cuenta nuestra red. Una cola HTB garantiza un porcentaje de ancho de banda a cada uno de los tráficos que pasan por la red.

Los parámetros que se pueden configurar en HTB son los que se explican a continuación:

- Rate: define el ancho de banda al que puede trabajar la clase raíz o clases hijas.(kbps).
- Ceil: determina el ancho de banda máximo que la clase puede usar. (kbps).
- Prio: se utiliza para dar prioridades a las clases, dependiendo del tráfico que almacena cada una de estas.

CAPÍTULO III

MARCO PROPOSITIVO

IMPLEMENTACIÓN DE MIDDLEWARE BAJO GNU/LINUX PARA CALIDAD DE SERVICIO

3.1. Pre-requisitos en GNU/Linux para garantías de calidad de servicio

Un driver o controlador es el encargado de actuar como una interfaz entre el sistema operativo (hardware) y los dispositivos que conforman al equipo (Ver figura 3.1). Se considera a los drivers como módulos dinámicos que se pueden cargar y descargar cuando se necesite, esto mediante la compilación de un nuevo kernel o recompilación del que se tenga en uso. El kernel de Linux diferencia tres tipos de drivers: drivers de

carácter (char devices), drivers de bloque (block devices), drivers de red (network devices).

El objetivo principal de un driver es permitir el acceso a los dispositivos de hardware, sin imponer restricciones arbitrarias a los que usan el driver. En Unix esto es considerado como una regla de diseño, que se conoce como separación de mecanismos y políticas.

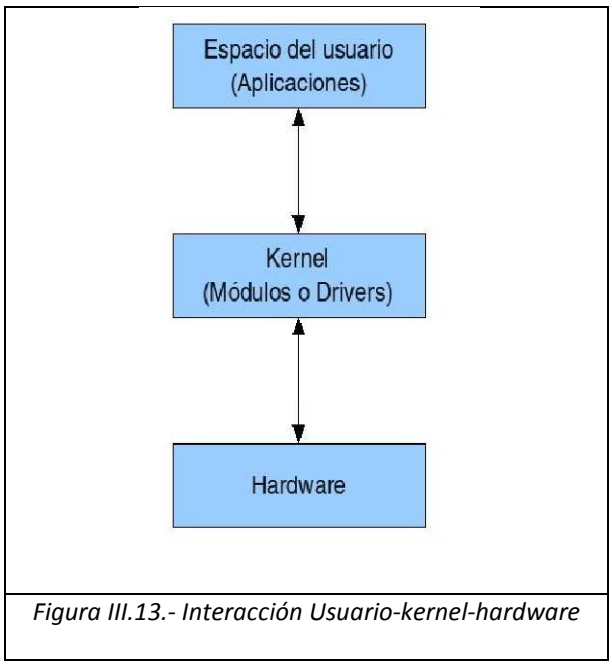


Figura III.13.- Interacción Usuario-kernel-hardware

GNU/Linux ofrece una gran cantidad de drivers para varios dispositivos, sin embargo en este trabajo se enfocará en los drivers que ofrece para algunas tarjetas inalámbricas para la creación de WLAN. Una instalación de un driver en GNU/Linux se puede realizar de varias maneras. Dependiendo si el kernel del sistema tiene el soporte necesario para tal dispositivo. En el caso de una tarjeta inalámbrica, el proceso de instalación se hace cuando se compila o recompila el

Kernel, el cual se explica a continuación. Es por ello que el primer requisito para dar pie a ofrecer garantías de calidad de servicio en redes WLAN es un driver y la instalación de una tarjeta inalámbrica en GNU/Linux.

Dentro de los drivers más importantes en GNU/Linux se encuentra Multiband Atheros Driver for Wireless Fidelity o MADWiFi , Aironet, Realtek, Orinoco. Se dice que un driver es considerado software libre si este proporciona el código fuente, ya sea para estudiarlo y entender su funcionamiento o para la modificación del mismo, en caso de ser necesario. El driver sobre el cual se trabajará y abarcará en esta tesis es MADWiFi, el cual funciona bajo el chipset Atheros; este driver soporta la API de wireless extension.

La razón primordial por la cual se decidió utilizar este driver es porque ya se contaba con una tarjeta inalámbrica con un chipset Atheros, soportado por este driver, además de que MADWiFi permite la creación de un punto de acceso. Es distribuido bajo una licencia doble, ya que está basada en las licencias GPL versión 2 y BSD. Considerado como uno de los drivers más avanzados en Linux, es soportado por dispositivos PCI, miniPCI. En lo que respecta a la seguridad MADWiFi permite las siguientes formas de cifrado: WEP, WAP, WAP2.

Una tarjeta inalámbrica puede trabajar en diferentes modos. Un modo es el estado sobre el cual una tarjeta inalámbrica trabaja en una red WLAN y que está limitada a

ciertas funcionalidades que el mismo modo ofrece. El más conocido de todos es el modo estación o cliente. Este driver permite ser operado en diferentes modos:

- Access Point (ap): la computadora funciona como un punto de acceso, es decir que los clientes se pueden conectar a los servicios que el access point o punto de acceso ofrece, como puede ser servicio de internet.
- Ad hoc (adhoc): en este modo, la tarjeta se puede conectar con otro cliente directamente para establecer una comunicación punto a punto.
- Estación (sta): la tarjeta wireless se conecta a un punto de acceso con la finalidad de establecer comunicación con otros clientes conectados con el access point mismo.
- Monitor (monitor): Permite capturar paquetes sin tener que conectarse a un AP o a una red ad-hoc.
- wds (wds): este modo permite conectarse con otros puntos de acceso con diferentes fines, tal es el caso de hacer un puente para crear una cobertura de la red más amplia.

Otro requisito ya mencionado arriba es la compilación de un kernel con las opciones necesarias para el soporte de redes inalámbricas. Además, se necesitará soporte para Calidad de Servicio y todos los mecanismos que se derivan de ella y que se explicarán en las siguientes secciones.

3.2. Instalación de la distribución DEBIAN/ETCH

GNU/Linux se ha desarrollado en muchas variantes, llamadas distribuciones, algunas de las cuales han sido pensadas para solventar necesidades específicas, por ejemplo se encuentran distribuciones con todas las herramientas para implementar seguridades en redes, otras que contienen exclusivamente software educativo, etc.; sin embargo en líneas generales se podría decir que hay dos grandes distribuciones de las cuales se desprenden las demás: DEBIAN y RED HAT/FEDORA.

De estas dos Red Hat se transformó en un sistema de pago y su contraparte Fedora puede ser considerada como la versión de prueba y gratuita. Debian es totalmente libre y gratuito y una de las más reconocidas y apreciadas por la comunidad de software libre por su escalabilidad y robustez.

Para el desarrollo de esta tesis se eligió Debian precisamente por estas características, además del hecho de su filosofía de desarrollo todo el software a instalarse sobre éste puede ser encontrado en fuentes, es decir, el usuario lo puede compilar e instalar según sus necesidades.

El proceso de instalación del sistema operativo se puede desarrollar de dos maneras, ya sea descargando los archivos en formato ISO desde algún repositorio FTP para luego grabarlos en un CD o directamente desde Internet.

Debian se subdivide a su vez en tres versiones: estable, de prueba e inestable, cada una de las cuales recibe un nombre código. Al momento del desarrollo de este trabajo la versión estable se denomina LENNY pero se trabajó con la versión anterior denominada ETCH por su alcance a la misma, la versión de prueba es squeeze y la inestable sid. La versión estable ha sido probada muchos meses antes de ser lanzada y efectivamente suele tener un comportamiento muy robusto, la mayoría de los problemas y bugs han sido solucionados, hay documentación disponible sobre todas sus características e incorpora una gran variedad de software. La versión de prueba en cambio suele tener problemas sobre ciertas plataformas y no tiene la ventaja de correr perfectamente en entornos con configuraciones no comunes o software nuevo. La versión inestable como su nombre lo indica puede generar más de un problema, aunque hay muchos usuarios que lo prefieren pues incorpora las últimas versiones del software disponible.

Se eligió Debian ETCH porque el software necesario para desarrollar esta tesis y el hardware disponible así lo requerían, al inicio se probó con CentOS pero se presentaron problemas de incompatibilidad entre las versiones del software disponible que se solucionaron con ETCH.

En primer lugar fue necesario descargar desde www.debian.org el primer CD de instalación, luego particionar el disco duro de la PC sobre la que se iba a instalar el sistema operativo y destinar la nueva partición para la raíz del sistema. En principio se eligió una instalación básica, es decir únicamente el sistema base.

Posteriormente se actualizó la base de datos del software disponible con el comando

#apt-dis update

Se modificó el archivo */etc/apt/sources.list* para que contenga:

deb ftp://ftp.debian.org/debian/ testing main
deb-src ftp://ftp.debian.org/debian/ testing main

deb ftp://ftp.es.debian.org/debian/ testing main
deb-src ftp://ftp.es.debian.org/debian/ testing main

deb http://security.debian.org/testing/ update main contrib

Estas son las direcciones web de los repositorios ftp Debian Etch. Con esto se asegura que al actualizar la distribución efectivamente se descarga e instale ETCH, desde el sitio de Debian en los Estados Unidos ftp.debian.org, y si no está disponible, entonces desde el sitio de España ftp.es.debian.org. La última línea hace alusión a un sitio especial, desde donde se descargaran parches de seguridad para el sistema.

Se ejecuta nuevamente apt-dist update y luego apt-dist update, este último comando es el encargado de actualizar la distribución. El proceso en total tardó 2 horas. Al final se consiguió un sistema operativo actualizado y compatible con los requerimientos para este desarrollo.

3.3. Compilación del kernel de Linux

El kernel o núcleo de un sistema operativo es la base sobre cual el software y el hardware se comunican entre sí para poder realizar todas las operaciones básicas de un sistema operativo, siendo realmente como el corazón del sistema, está compuesto por miles de millones de líneas de código, programadas a un bajo nivel, puesto que trabajan con el hardware de una computadora. En cuanto al núcleo de Linux, se puede decir que es de tipo monolítico, es decir, un kernel grande que engloba todas las tareas ya mencionadas en un solo "software". También posee una ventaja muy importante respecto a otros sistemas operativos de tipo privativo o propietarios: es software libre. Esto quiero decir que el código del mismo se encuentra de manera libre para ser descargada y vista por cualquier persona. De igual forma, existe la libertad de distribuir el código y modificarlo sin ningún inconveniente por parte de los desarrolladores. Al contrario, cualquier aportación es recibida de manera positiva por la comunidad GNU/Linux.

Actualmente Linux se encuentra en su versión 2.6.31.12, la cual es la última versión estable, esto quiere decir que se ha probado de errores lo más posible y está lista para su liberación y descarga. Hasta que empezó el desarrollo de la serie 2.6 del núcleo, existieron dos tipos de versiones del núcleo. La versión de producción, es lo que ahora se le conoce como versión estable. Las versiones de desarrollo se pueden comparar con aplicaciones denominadas betas, o lo que es lo mismo,

versiones del kernel que aún no ha sido probadas al cien por ciento o que están en pruebas y se experimenta aún con ellas.

La versión de kernel que etch trae consigo por defecto es la 2.6.18.5. Cada vez que una versión es liberada se agregan nuevas características y como es de esperarse la última versión estable posee nuevos aspectos. Lo que compete, en este caso, son los elementos necesarios para dar soporte de redes inalámbricas, calidad de servicio y traffic shaper.

El soporte para QoS en versiones recientes del kernel en Linux provee una plataforma para la implementación de varias tecnologías de calidad de servicio como servicios integrados y servicios diferenciados. Para el soporte de QoS en el kernel de Linux se necesita realizar un proceso de re-compilación del kernel, puesto que la mayoría de las distribuciones no tienen habilitado esta opción cuando se instala el sistema. Debian, por defecto, tampoco trae consigo esta opción. El proceso de recompilación de un kernel significa en términos generales ajustar un sistema operativo GNU/Linux a la medida mediante una serie de pasos que se explicarán a continuación.

Cabe destacar que el proceso de compilación de kernel en Debian es un poco distinto respecto a otras distribuciones pero no se entrará en detalle acerca de la compilación en otra distribución. Lo primero que hay que hacer es descargar la versión 2.6.23 de la página oficial donde están todas las versiones de los

kernel de Linux: http://www.kernel.org/pub/linux/kernel/v2.6/, en formato .tar.gz, ya sea con el comando wget y la URL del kernel a descargar o mediante un navegador Web, específicamente, se necesita descargar el archivo linux-2.6.23.tar.gz.

Ahora se procede a copiar el archivo comprimido dentro del directorio */usr/src/;* esto se logra mediante el comando *cp /ruta/ linux-2.6.23.tar.gz /usr/src/*. Ya estando dentro del directorio especificado, se descomprime utilizando la línea de comandos, de esta forma se escribirá:

tar xvzf linux-2.6.23.tar.gz

Ya descomprimido el archivo, se creará automáticamente un directorio llamado linux-2.6.23, en donde tendrá ya las fuentes del código necesarias para comenzar a compilar.

Ahora, el siguiente paso es instalar algunas aplicaciones que servirán durante este proceso de compilación: *apt-get install kernel-package libncurses5-dev fakeroot build-essential.* Con este comando estamos instalando primero el paquete kernel-package, el cual es utilizado para crear paquetes personalizados del tipo de Debian de un kernel que se haya compilado como se desee. libncurses5-dev es un conjunto de librerías utilizadas para los desarrolladores para escribir interfaces de usuario en terminales independientes. libncurses optimiza la pantalla de cambios, a fin de reducir la latencia experimentada cuando se usan los accesos remotos y

algunos otros aspectos relacionados; esta se ocupará para proporcionar un menú de configuración del kernel. El paquete fakeroot funciona para crear paquetes o algunos archivos y darle a los mismos privilegios de superusuario (root) sin tenerlos, de ahí el nombre de fakeroot o falso root. Por último, está el conjunto de paquetes build-essential, los cuales contienen los compiladores de C y C++ necesarios para generar los paquetes de instalación del nuevo kernel a instalar.

Posteriormente se eligen las opciones que con las que el nuevo kernel arrancará cuando se haya compilado. Si se desea se puede hacer una copia del archivo de configuración del kernel en uso para tener una base para continuar configurando. Cuando una distribución Debian se instala, en este caso Etch, con las opciones genéricas de un kernel, las configuraciones de tal kernel se guardan en el archivo */boot/config-2.6.18-5-486*.

Si se decidió por copiar el archivo a las fuentes del nuevo kernel entonces se escribirá una línea como la siguiente:

server:/usr/src#cp config- 6.18-5-486 ./.config

Después de copiarse esto, ahora se procede a ver el menú de edición de opciones del núcleo, esto se realiza tecleando el comando siguiente y estando siempre dentro del directorio */usr/src/:*

server:/usr/src#make menuconfig

De esta forma saldrá la pantalla principal, la cual se muestra a continuación en la figura III.14

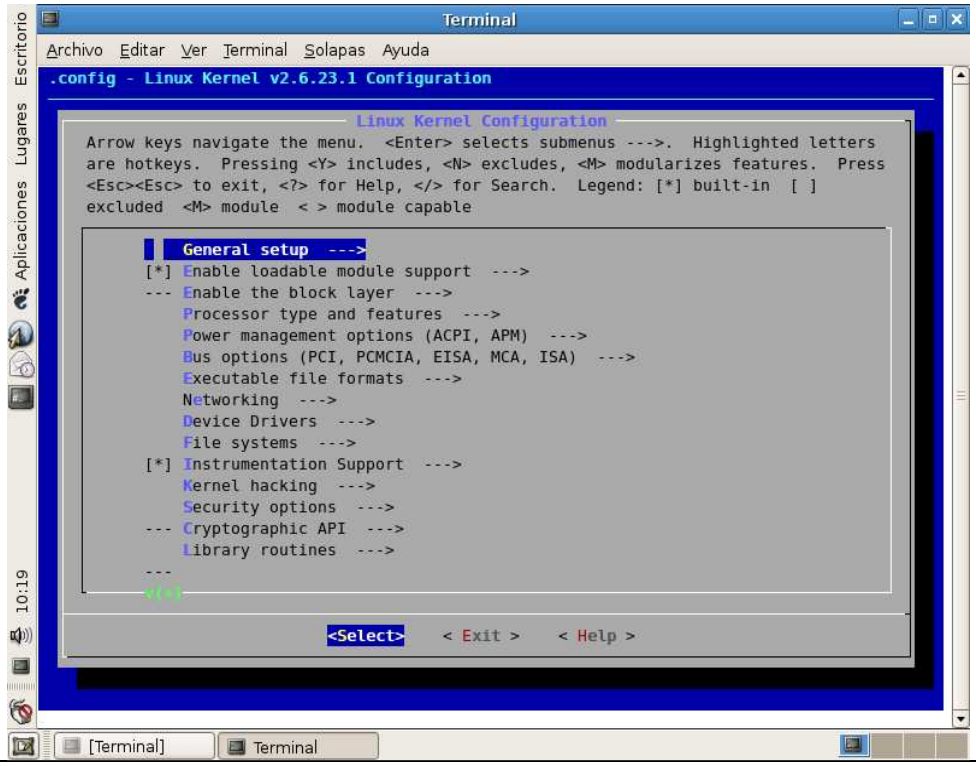


Figura III.14- . Menú principal de configuración del kernel versión 2.6.23

Estando en el menú principal, se cargará el archivo de configuración que se respaldó. Por tanto hay que elegir la opción “Load an Alternate Configuration File”; como el archivo .config se encuentra en el mismo directorio, no hay que especificar ruta, simplemente el nombre del archivo como se muestra en la figura III.15.

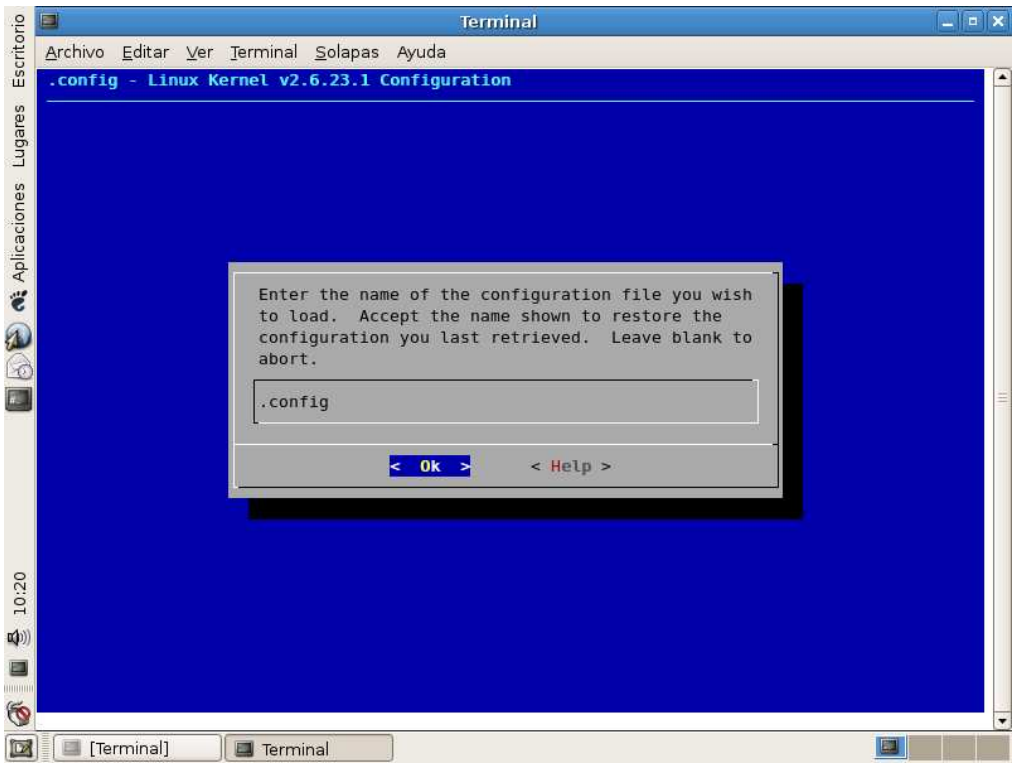


Figura III.15.- Cargar un archivo de configuración existente.

Antes de compilar, primero hay que escoger las opciones que se requieran para ajustar un kernel a la medida; en este caso las opciones requeridas para soporte de QoS o calidad de servicio son las siguientes con un kernel versión 2.6.23.

Netwoking → Networking Options → QoS and/or fair queueing. Esta es la ruta que se debe seguir en el menú para navegar a la opción de QoS and/or fair queueing.

De esta forma se habilitarán cada una de las opciones como integradas al kernel, lo cual significa que al arrancar el sistema operativo, las opciones estarán listas para utilizarse. Para ello se debe seleccionar la opción a elegir y presionar la tecla “Y”. Así se pondrá un asterisco (*) en la parte izquierda de cada elemento. Si en lugar del

asterisco aparece una M quiere decir que el elemento cargará como un módulo del kernel y se tendrá que cargar dinámicamente cada vez que se necesite.

Las opciones que se habilitan en la ruta de arriba servirán posteriormente para realizar las modificaciones pertinentes respecto del mecanismo DRAP. En estas opciones que se enlistan a continuación hay diferentes disciplinas de colas prioritarias para el tráfico saliente. También existen opciones para el control de la tasa de transferencia y clasificadores para el tipo de tráfico. En las figuras III.16 Y III.17 se aprecian estas opciones:

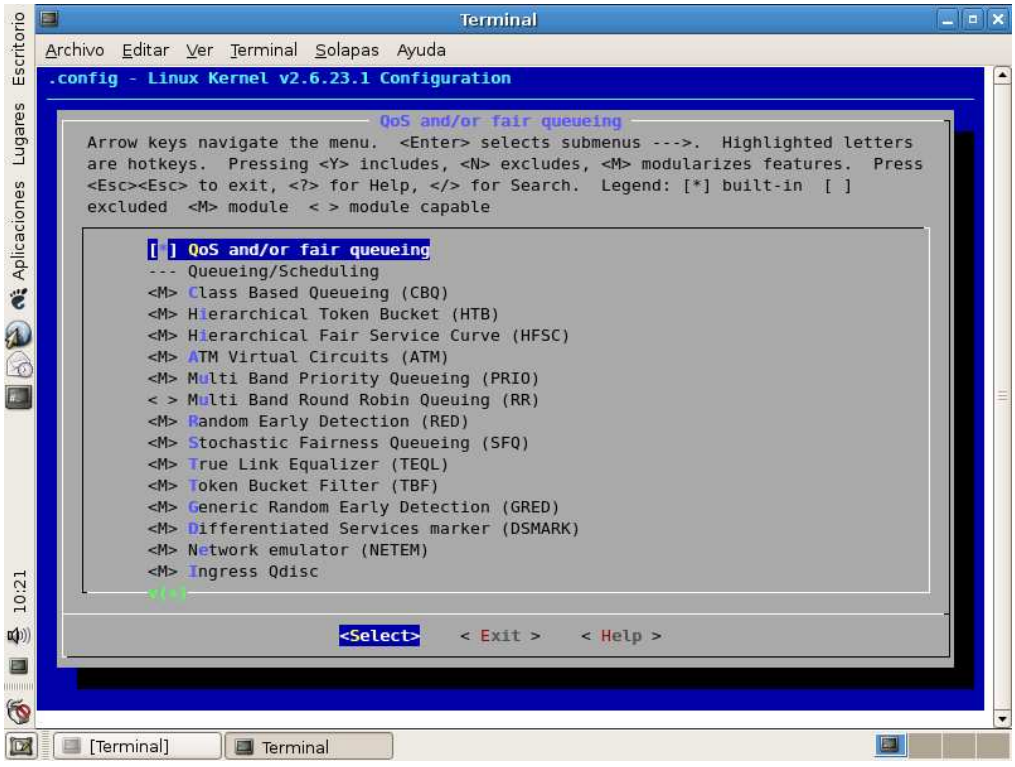


Figura III.16.- Opciones del menú QoS and/or fair queueing, parte 1

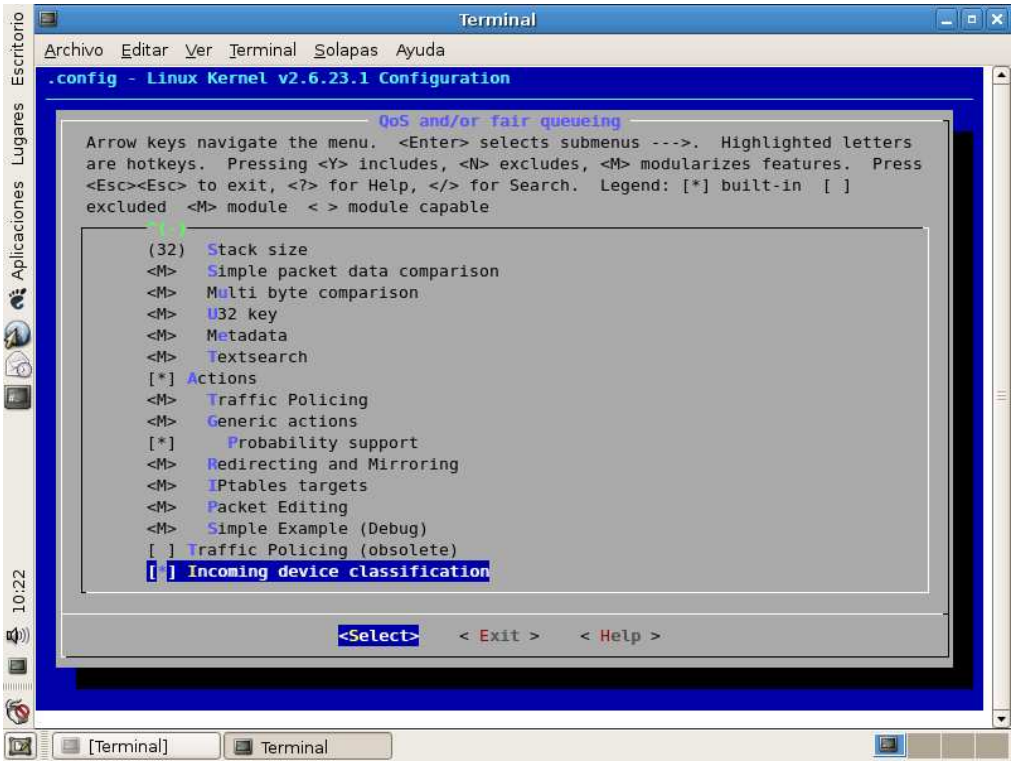


Figura III.17.- Opciones del menú QoS and/or fair queueing, parte 2

Posteriormente se encuentra el traffic shaper, un programa que permite controlar el ancho de banda mediante algunos scripts que vienen con la instalación del mismo. Siempre y cuando la opción Traffic Shaper esté habilitada en el kernel, tal como en la figura III.18, a través de Device Drivers → Networking Device Support → Traffic shaper

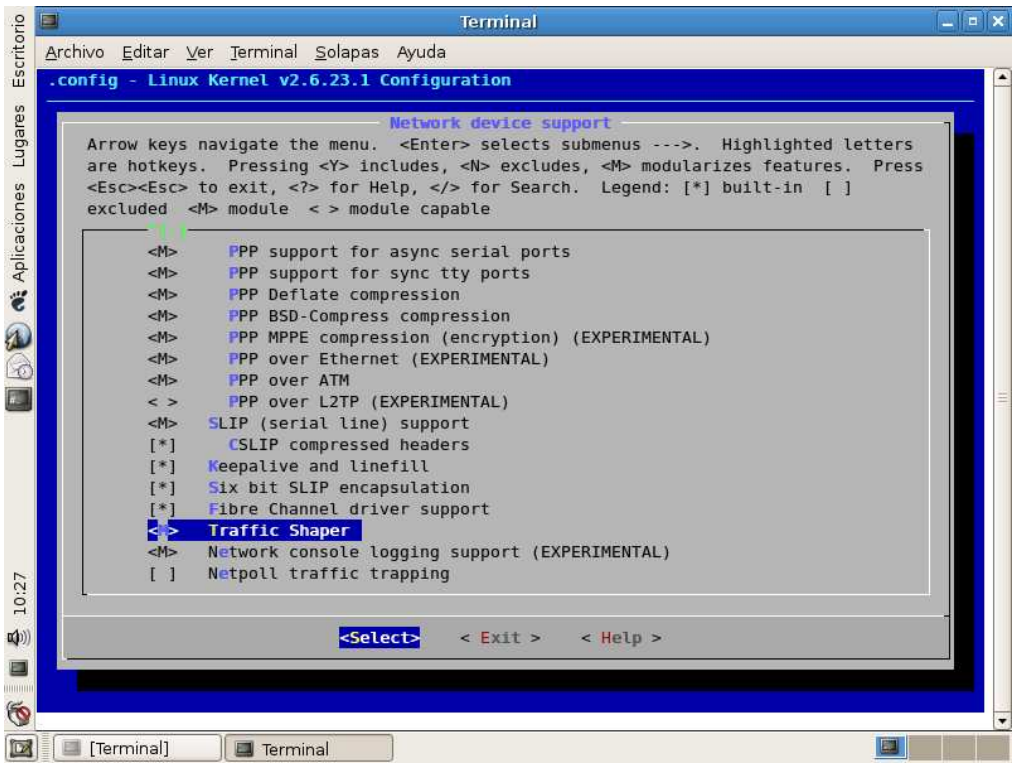


Figura III.18. La opción Traffic Shaper de Linux

Finalmente, se presiona dos veces seguidas la tecla Esc para salir de este menú. Aquí se mostrará un diálogo que preguntará si se desea guardar o no guardar la nueva configuración, por lo que se responderá que sí a la pregunta y se dejará el mismo nombre que maneja por defecto: .config, tal como se muestra en la figura III.19.

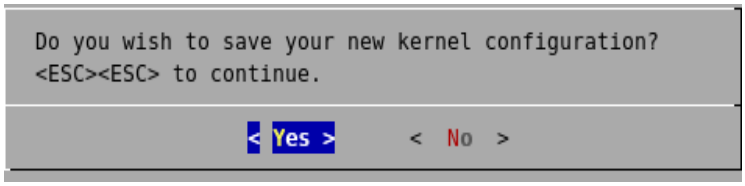


Figura III.19. Diálogo de guardar configuración

El siguiente paso es construir el kernel. Aquí se compilará y construirá el código fuente, generando los paquetes de Debian que se instalarán finalmente. Para compilar las fuentes se debe escribir los siguientes comandos, como siempre, en la línea de comandos:

make-kpkg clean

fakeroot make-kpkg --initrd kernel_image kernel_headers

La opción --initrd crea una imagen initrd en el paquete que se guardará en /boot cuando se instala el kernel. El resultado del comando anterior va a ser crear dos paquetes con extensión .deb en el directorio superior de donde está el código fuente del kernel. Un paquete va a ser el kernel completo y el otro va a ser los kernel-headers o cabeceras. La opción de kernel_headers de la compilación es opcional, pero es recomendable debido a que muchos programas y módulos necesitan tener las cabeceras del kernel que se está usando para poder ser instalados. Este es el caso de MADWiFi. El tiempo de compilación puede variar, dependiendo de la velocidad de procesador que se tenga y de las opciones que se haya habilitado, ya sea como módulos o integrados. Puede realizarse desde media hora hasta dos horas o más.

Una vez que terminó la compilación de los paquetes, se debe subir un nivel en el directorio donde se encuentre. Así, en la ruta /usr/src/ deben existir dos

paquetes con extensión .deb. Para instalar los dos paquetes de Debian generados en el proceso anterior, se utiliza la instrucción dpkg, de esta forma:

dpkg -i linux-image-2.6.23.9_10.00.Custom_i386.deb

dpkg -i linux-headers-2.6.23.9_10.00.Custom_i386.deb

Finalmente, se reinicia la máquina con la instrucción shutdown –r now o init 6 y se selecciona la primer opción del menú de inicio, ya que aparecerá la versión del kernel nuevo, en este caso la 2.6.23.9. De esta forma se finaliza el proceso de re-compilación de un kernel para dar soporte a herramientas inalámbricas y de calidad de servicio.

3.4. Instalación de Wireless Tools

Estas son una serie de suite de herramientas que permiten la comunicación eficiente y rápida del usuario con las distintas opciones de configuración de las tarjetas inalámbricas, estas pensadas precisamente para facilitar las tareas de instalación, configuración y monitoreo de este tipo de dispositivos.

Desde la página oficial de Jean Tourrilhes se descarga el fichero de fuentes wireless_tools.28.pre16.tar.gz. (http://www.hpl.hp.com/personal/Jean_Tourrilhes)

Se ingresa al directorio donde se haya descargado las wireless tools, en este caso bajo usr/src, y se ejecuta la siguiente secuencia de órdenes para descomprimirlo:

debian:#cd /usr/src/

debian:#tar –xzvf wireless_tools.28.pre16.tar.gz

Se crea un nuevo directorio llamado wireless_tools.28.pre16/ en donde se encuentran los ficheros fuente C, se ejecutó el siguiente comando para compilarlos e instalarlos:

debian:/usr/src/#cd wireless_tools.28.pre16

debian:/usr/src/#make

debian:/usr/src/#make install

Se ingresa en el archivo: */etc/ld.so.conf* y se añade la siguiente línea:

/usr/local/lib

Para permitir que el sistema reconozca y ejecute los nuevos comandos.

Por último se ejecuta

debian:#ldconfig

Se verifica el normal funcionamiento con la orden iwconfig; que muestra algo parecido a:

lo no wireless extensions

eth0 no wireless extensions

Es decir el sistema encuentra dos interfaces de red, la de loopback y eth0 o Ethernet, ninguna de las cuales es un adaptador inalámbrico por lo que no soportan wireless extensions, las extensiones del kernel para el manejo de los dispositivos inalámbricos que anteriormente se activaron al compilarlo.

Únicamente cuando wireless tools se hayan compilado e instalado se puede proceder a instalar el driver MADWifi.

3.3. Instalación del Software MADWiFi. (Driver tarjeta inalámbrica)

MADWifi es la contracción de Multiband Atheros Driver for Wifi; driver crado para manipular vía software las tarjetas inalámbricas con chipset Atheros. Este chipset no está soportado nativamente por el kernel de Linux, por lo que es necesario compilarlo e instalarlo.

Atheros es una compañía norteamericana especializada en tecnología inalámbrica que desarrolló este chipset con una propiedad particular: soporta varias bandas de frecuencia simultáneamente y permite crear puntos de acceso para redes inalámbricas. El primer firmware fue desarrollado para la plataforma BSD y luego fue trasladado para la plataforma Linux.

Al momento hay varias generaciones del Chip Atheros, sin embargo las más importantes son:

- 5210 Soporta 802.11a y encriptación WEP vía hardware
- 5211 Soporta 802.11 a y b además de encriptación WEP y AES/OCB vía hardware
- 5212 Soporta 802.11 a, b y g; encriptaciones WEP, TKIP, y AES vía hardware.

El driver se basa en un archivo llamado HAL (Hardware Access Layer, capa de acceso al hardware) que hace las veces de firmware actuando como módulo en el kernel del sistema. HAL se distribuye en un archivo de solo lectura debido a que chipset Atheros puede funcionar fácilmente en frecuencias fuera de las bandas ISM, las mismas que requieren licencias y por razones de seguridad y reglamentación es preferible suministrar este archivo sin permitir que los usuarios lo puedan manipular. Razón por la cual MADWifi no es un código con licencia GPL ya que HAL es propietario.

HAL está conformado por dos archivos, un binario pre-compilado y ah_osdec.c el cual actúa como intermediario entre el kernel y el binario. Los dos archivos trabajan juntos y forman el módulo kernel ath_hal.ko.

MADWifi depende de dos módulos:

- wlan.ko: Contiene toda la parte de soporte del protocolo 802.11. Esta derivado del primer código incluido en NetBSD y FreeBSD
- ath_hal.ko: Contiene las instrucciones específicas del hardware Atheros.

El driver funciona como un dispositivo normal de red, por lo que está al 100% integrado en el sistema. Hay un solo driver que soporta tanto tarjetas PCI como PCMCIA. Soporta tanto modo máster (Access Point), como modo managed (cliente red inalámbrica con Access Point), y ad-hoc (cliente de red inalámbrica entre iguales); además que es posible poner al adaptador inalámbrico en modo monitor, es decir que actúe como sniffer de una red inalámbrica para detectar y capturar paquetes. Ha sido testeado tanto con kernels 2.4 como 2.6 y requiere extensiones wireless tools posteriores a 14 (kernel 2.4.28 o posterior). Antes de instalarlo y para asegurarse que efectivamente la tarjeta será soportada por MADWifi es recomendable ejecutar lspci, este comando mostrará todos los dispositivos con interfaz PCI conectados en el sistema y arroja un resultado parecido a la figura III.20.

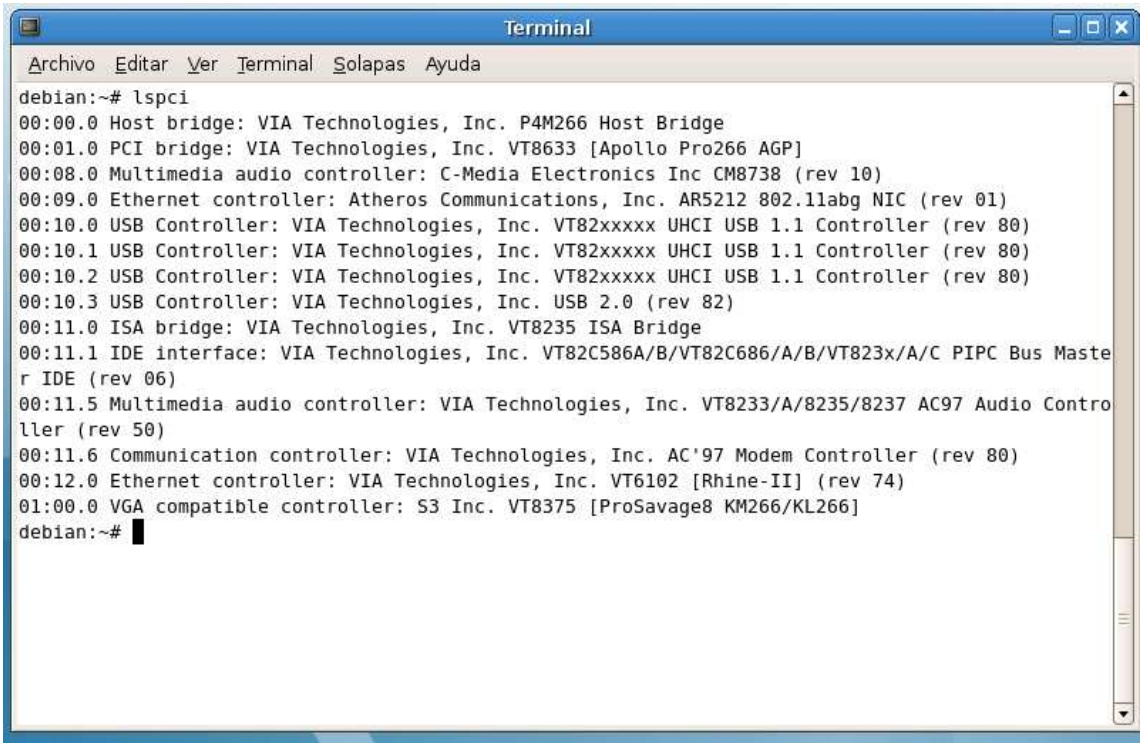


Figura III.20.- Resultado ejecución lspci en PC con tarjeta inalámbrica instalada

Obsérvese la descripción del cuarto dispositivo encontrado, el sistema detecta que efectivamente se trata de una tarjeta de red (NIC) con chipset Atheros del tipo 5212 que soporta los protocolos a, b y g (es importante anotar aquí que algunos fabricantes cambian el chipset pero sin cambiar la denominación que detectará el sistema, por lo que es muy posible que la tarjeta de trabajo aunque aparente ser Atheros en realidad no lo sea).

Con este resultado se procede entonces a la obtención e instalación del driver. Este se puede obtener desde varios lugares, sin embargo el método más confiable y donde siempre se podrá descargar la última versión es a través de la página oficial del proyecto en www.madwifi.org.

Se mencionarán los pasos a seguir para la instalación de este driver previa re-compilación del kernel , los requisitos necesarios para la instalación son los siguientes:

- Sistema operativo GNU/Linux (Debian Etch).
- Kernel 2.4.x o 2.6.x (Cabeceras y Fuentes).
- Tarjeta inalámbrica con chipset Atheros (En este trabajó se uso una D-Link AirPlus XtremeG Mod. DWL-G520).
- Herramientas de Linux (gcc, make, wireless-tools)
- Fuentes del driver (Se utilizó MADWiFi 0.9.3.3)

Cabe mencionar que el proceso de instalación de una tarjeta inalámbrica en GNU/Linux puede variar dependiendo de la marca de la misma y del driver que se utiliza.

Los pasos para la instalación de MADWiFi son los siguientes:

1. Descomprimir el archivo .tar.gz

   debian:/usr/src# tar -xvzf madwifi-0.9.3.3.tar.gz

2. Compilar el driver, al teclear make aparecerá la salida siguiente.

   debian:/usr/src /madwifi-0.9.3.3# make

   Checking requirements... ok.
   Checking kernel configuration... ok.
   make -C /lib/modules/2.6.23/build SUBDIRS= debian:/usr/src/madwifi-0.9.3.3 modules
   make[1]: Entering directory `/usr/src/linux-2.6.23'
   CC [M] /usr/src /madwifi-0.9.3.3/ath/if_ath.o
   CC [M] /usr/src /madwifi-0.9.3.3/ath/if_ath_pci.o
   LD [M] /usr/src /madwifi-0.9.3.3/ath/ath_pci.o
   CC [M] / usr/src /madwifi-0.9.3.3/ath_hal/ah_os.o
   HOSTCC / usr/src /madwifi-0.9.3.3/ath_hal/uudecode
   UUDECODE / usr/src /madwifi-0.9.3.3/ath_hal/i386-elf.hal.o
   LD [M] / usr/src /madwifi-0.9.3.3/ath_hal/ath_hal.o
   CC [M] / usr/src /madwifi-0.9.3.3/ath_rate/amrr/amrr.o
   LD [M] / usr/src /madwifi-0.9.3.3/ath_rate/amrr/ath_rate_amrr.o
   CC [M] / usr/src /madwifi-0.9.3.3/ath_rate/onoe/onoe.o
   LD [M] / usr/src /madwifi-0.9.3.3/ath_rate/onoe/ath_rate_onoe.o
   CC [M] / usr/src /madwifi-0.9.3.3/ath_rate/sample/sample.o
   LD [M] / usr/src /madwifi-0.9.3.3/ath_rate/sample/ath_rate_sample.o
   CC [M] / usr/src /madwifi-0.9.3.3/net80211/if_media.o
   CC [M] / usr/src /madwifi-0.9.3.3/net80211/ieee80211.o
   CC [M] / usr/src /madwifi-0.9.3.3/net80211/ieee80211_beacon.o
   CC [M] / usr/src /madwifi-0.9.3.3/net80211/ieee80211_crypto.o
   CC [M] / usr/src /madwifi-0.9.3.3/net80211/ieee80211_crypto_none.o
   CC [M] / usr/src /madwifi-0.9.3.3/net80211/ieee80211_input.o

```
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_node.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_output.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_power.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_proto.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_scan.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_wireless.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_linux.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_monitor.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_rate.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_acl.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_crypto_ccmp.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_scan_ap.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_scan_sta.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_crypto_tkip.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_crypto_wep.o
CC [M]  / usr/src /madwifi-0.9.3.3/net80211/ieee80211_xauth.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan_wep.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan_tkip.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan_ccmp.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan_acl.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan_xauth.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan_scan_sta.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan_scan_ap.o
Building modules, stage 2.
MODPOST 13
modules
CC      / usr/src /madwifi-0.9.3.3/ath/ath_pci.mod.o
LD [M]  / usr/src /madwifi-0.9.3.3/ath/ath_pci.ko
CC      / usr/src /madwifi-0.9.3.3/ath_hal/ath_hal.mod.o
LD [M]  / usr/src /madwifi-0.9.3.3/ath_hal/ath_hal.ko
CC      / usr/src /madwifi-0.9.3.3/ath_rate/amrr/ath_rate_amrr.mod.o
LD [M]  / usr/src /madwifi-0.9.3.3/ath_rate/amrr/ath_rate_amrr.ko
CC      / usr/src /madwifi-0.9.3.3/ath_rate/onoe/ath_rate_onoe.mod.o
LD [M]  / usr/src /madwifi-0.9.3.3/ath_rate/onoe/ath_rate_onoe.ko
CC      / usr/src /madwifi-0.9.3.3/ath_rate/sample/ath_rate_sample.mod.o
LD [M]  / usr/src /madwifi-0.9.3.3/ath_rate/sample/ath_rate_sample.ko
CC      / usr/src /madwifi-0.9.3.3/net80211/wlan.mod.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan.ko
CC      / usr/src /madwifi-0.9.3.3/net80211/wlan_acl.mod.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan_acl.ko
CC      / usr/src /madwifi-0.9.3.3/net80211/wlan_ccmp.mod.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan_ccmp.ko
CC      / usr/src /madwifi-0.9.3.3/net80211/wlan_scan_ap.mod.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan_scan_ap.ko
CC      / usr/src /madwifi-0.9.3.3/net80211/wlan_scan_sta.mod.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan_scan_sta.ko
CC      / usr/src /madwifi-0.9.3.3/net80211/wlan_tkip.mod.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan_tkip.ko
CC      / usr/src /madwifi-0.9.3.3/net80211/wlan_wep.mod.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan_wep.ko
CC      / usr/src /madwifi-0.9.3.3/net80211/wlan_xauth.mod.o
LD [M]  / usr/src /madwifi-0.9.3.3/net80211/wlan_xauth.ko
make[1]: Leaving directory `/usr/src/linux-2.6.23'
make -C ./tools  all || exit 1
```

```
make[1]: Entering directory `/ usr/src /madwifi-0.9.3.3/tools'
gcc -o athstats -g -O2 -Wall -I. -I../hal -I.. -I../ath  athstats.c
gcc -o 80211stats -g -O2 -Wall -I. -I../hal -I..  80211stats.c
gcc -o athkey -g -O2 -Wall -I. -I../hal -I..  athkey.c
gcc -o athchans -g -O2 -Wall -I. -I../hal -I..  athchans.c
gcc -o athctrl -g -O2 -Wall -I. -I../hal -I..  athctrl.c
gcc -o athdebug -g -O2 -Wall -I. -I../hal -I..  athdebug.c
gcc -o 80211debug -g -O2 -Wall -I. -I../hal -I..  80211debug.c
gcc -o wlanconfig -g -O2 -Wall -I. -I../hal -I..  wlanconfig.c
make[1]: Leaving directory `/ usr/src /madwifi-0.9.3.3/tools'
```

La compilación generará tres ficheros importantes:

- ath_pci.ko (driver para PCI/PCMCIA)
- ath_hal.ko (Atheros HAL)
- wlan.ko (soporte 802.11)

3. Para instalar el driver es necesario tener los privilegios de superusuario (root).

debian:/ usr/src /madwifi-0.9.3.3# make install

```
sh scripts/find-madwifi-modules.sh 2.6.23
for i in ./ath ./ath_hal ./ath_rate ./net80211; do \
        make -C $i install || exit 1; \
    done
make[1]: Entering directory `/ usr/src /madwifi-0.9.3.3/ath'
test -d //lib/modules/2.6.23/net || mkdir -p //lib/modules/2.6.23/net
install ath_pci.ko //lib/modules/2.6.23/net
make[1]: Leaving directory `/ usr/src /madwifi-0.9.3.3/ath'
make[1]: Entering directory `/ usr/src /madwifi-0.9.3.3/ath_hal'
test -d //lib/modules/2.6.23/net || mkdir -p //lib/modules/2.6.23/net
install ath_hal.ko //lib/modules/2.6.23/net
make[1]: Leaving directory `/ usr/src /madwifi-0.9.3.3/ath_hal'
make[1]: Entering directory `/ usr/src /madwifi-0.9.3.3/ath_rate'
for i in amrr/ onoe/ sample/; do \
        make -C $i install || exit 1; \
    done
make[2]: Entering directory `/ usr/src /madwifi-0.9.3.3/ath_rate/amrr'
test -d //lib/modules/2.6.23/net || mkdir -p //lib/modules/2.6.23/net
install ath_rate_amrr.ko //lib/modules/2.6.23/net
make[2]: Leaving directory `/ usr/src /madwifi-0.9.3.3/ath_rate/amrr'
```

```
make[2]: Entering directory `/ usr/src /madwifi-0.9.3.3/ath_rate/onoe'
test -d //lib/modules/2.6.23/net || mkdir -p //lib/modules/2.6.23/net
install ath_rate_onoe.ko //lib/modules/2.6.23/net
make[2]: Leaving directory `/ usr/src /madwifi-0.9.3.3/ath_rate/onoe'
make[2]: Entering directory `/ usr/src /madwifi-0.9.3.3/ath_rate/sample'
test -d //lib/modules/2.6.23/net || mkdir -p //lib/modules/2.6.23/net
install ath_rate_sample.ko //lib/modules/2.6.23/net

make[2]: Leaving directory `/v/madwifi-0.9.3.3/ath_rate/sample'
make[1]: Leaving directory `/ usr/src /madwifi-0.9.3.3/ath_rate'
make[1]: Entering directory `/ usr/src /madwifi-0.9.3.3/net80211'
test -d //lib/modules/2.6.23/net || mkdir -p //lib/modules/2.6.23/net
for i in wlan.o wlan_wep.o wlan_tkip.o wlan_ccmp.o wlan_acl.o wlan_xauth.o wlan_scan_sta.o
wlan_scan_ap.o; do \
        f=`basename $i .o`; \
        install $f.ko //lib/modules/2.6.23/net; \
    done
make[1]: Leaving directory `/ usr/src /madwifi-0.9.3.3/net80211'
(export KMODPATH=/lib/modules/2.6.23/net; /sbin/depmod -ae 2.6.23)
make -C ./tools  install || exit 1
make[1]: Entering directory `/ usr/src /madwifi-0.9.3.3/tools'
install -d /usr/local/bin
for i in athstats 80211stats athkey athchans athctrl athdebug 80211debug wlanconfig; do \
        install $i /usr/local/bin/$i; \
        strip /usr/local/bin/$i; \
    done
install -d /usr/local/man/man8
install -m 0644 man/*.8 /usr/local/man/man8
make[1]: Leaving directory `/usr/src/madwifi-0.9.3.3/tools'
```

4. Ahora lo que se hace es cargar lo módulos del driver MADWiFi al sistema operativo. Después de teclear el modprobe deberá aparecer la nueva interfaz (ath0).

```
debian:/ usr/src /madwifi-0.9.3.3# modprobe ath_pci

debian:/ usr/src /madwifi-0.9.3.3# iwconfig ath0
```

Se ejecuta la orden iwconfig ath0 (las interfaces inalámbricas, para poder diferenciarlas de las interfaces de red Ethernet denominadas ethx suelen denominarse

athx, correspondiendo x al número de interfaz usada). Se obtiene una salida parecida a la mostrada en la figura III.21.

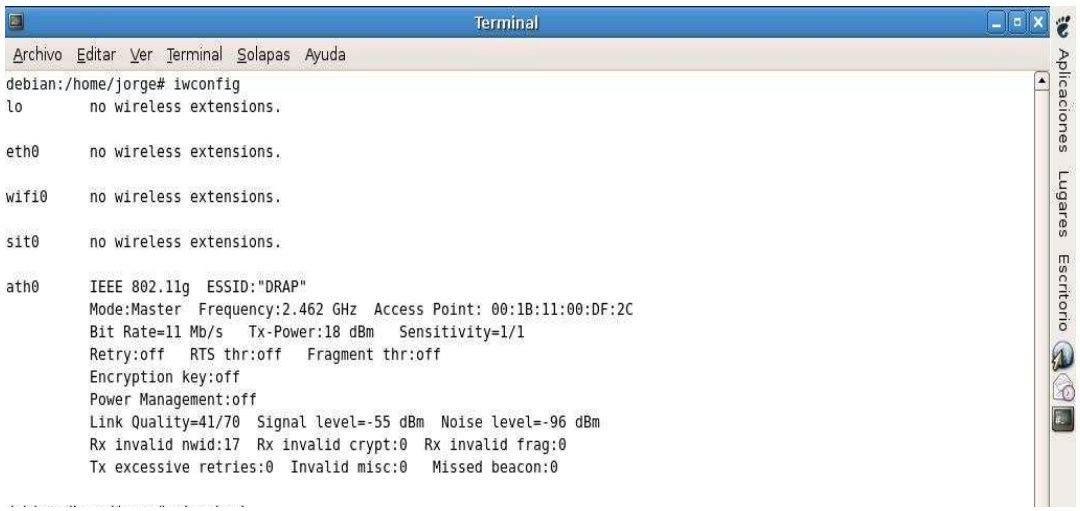


Figura III.21.- Resultado ejecución iwconfig

La pantalla de la figura muestra que se han encontrado tres interfaces de red, eth0 (tarjeta de red Realtek), lo (interfaz de red de loopback) y ath0 (tarjeta de red inalámbrica), donde las dos primeras no soportan las wireless tools y la tarjeta de red ath0 está levantada con la configuración por defecto.

5. Configurar la interfaz inalámbrica, mediante el siguiente comando.

debian:/ usr/src /madwifi-0.9.3.3# iwconfig ath0 essid "NOMBRE DE LA RED" KEY "CONTRASEÑA EN CASO DE TENER" open

Ahora bien, para configurar la interfaz inalámbrica, iwconfig provee los siguientes parámetros (no todos están desarrollados al momento).

3.3.1. ***iwconfig*** interface [essid X] [freq F][channel C] [sens S] [ap A] [rate R] [rts RT] [frag FT] [txpower T] [enc E] [key K] [retry R]

3.3.2. essid o Nombre de la red.- Configura el nombre de la red inalámbrica, si se usa en modo managed (cliente de una red inalámbrica de infraestructura) indica la red a la que se conectará la PC del usuario, si se usa en modo máster (punto de acceso).

3.3.3. freq/channel (frecuencia o canal de uso).- Se puede utilizar cualquiera (pero no ambos) de los dos parámetros e indican la frecuencia en la cual funciona el access point o el canal de uso. Si se ingresa un valor entre 1 y 11 el sistema lo interpretara como el canal que se va a usar. Si por contrario se desea configurar una frecuencia de uso entonces se ingresa esta frecuencia en Kilohertz, Megahertz o Gigahertz, siempre y cuando esta sea una frecuencia válida dentro del espectro de las redes inalámbricas.

3.3.4. sens (Umbral de sensibilidad).- No está implementado en las distribuciones existentes pero se espera que se determine el menor nivel de señal para el cual se recibirán paquetes.

3.3.5. ap (Uso específico de un ap).- Si la interfaz está configurada como managed significa que será cliente de algún punto de acceso, esta opción requiere el ingreso de una dirección MAC del AP al cual se conecta.

3.3.6. rate (Velocidad de transmisión).-Las tarjetas Atheros al soportar diversos protocolos soportarán también distintas velocidades de transmisión: 2 Mbps, 11 Mbps, 54 Mbps, 108 Mbps, etc. Con este parámetro se determina la velocidad teórica máxima a la cual la red inalámbrica trabajará; se digita el valor deseado acompañado de la unidad de medida correspondiente. Es posible utilizar también el modificador *auto* con el cual se permite que sea la interfaz la que determine la mejor velocidad a la cual trabajar.

3.3.7. rts (Umbral rts/cts).- Especifica cuál será la longitud del paquete para el sistema que enviará peticiones rts. Se debe tener cuidado con esta configuración pues puede llegar a ralentizar la red.

3.3.8. frag (Umbral de fragmentación).- Determina la longitud de fragmentación de los paquetes.

3.3.9. key/enc.- Sirven para manipular los distintos tipos de encriptación y niveles de seguridad especificados en el estándar.

Key manipula las claves WEP y los modos de autenticación, enc manipula los tipos de encriptación.

3.3.10. txpower.- Configurará la potencia de transmisión de la interfaz a X dBm. No está implementado.

3.3.11. retry.- Configurará el número máximo de retransmisiones a usarse. No está implementado.

Después de este paso, se está comprobando que la tarjeta está funcionando correctamente en modo cliente y, por consiguiente, fue instalada exitosamente.

3.4. Instalación y configuración de un servidor para control de Calidad de Servicio en Redes Inalámbricas de Área Local.

3.4.1 Creación de un Hotspot en GNU/Linux

Un punto de acceso va a ser el modo indicado para este trabajo de tesis por ello lo que prosigue en esta etapa del proyecto es la creación de un Hotspot con la tarjeta inalámbrica que se tiene ya instalada. Un Hotspot es una zona geográfica con cobertura de red WiFi, en donde un access point provee de servicios de red (generalmente servicio de internet) a los clientes conectados dentro de la zona de cobertura de un Hotspot y generalmente se encuentran abiertas, es decir, no

requieren de algún tipo de cifrado para poder accederlos, dependiendo de las políticas del lugar.

Se configurará los elementos necesarios para la creación de un Hotspot bajo GNU/Linux. Se necesitará una tarjeta inalámbrica trabajando en modo access point a una velocidad de conexión de 11 o 54 Mbps, al igual que sus clientes. Tal hotspot debe brindar a sus clientes la posibilidad de conectarse a este y tener acceso a internet de una forma transparente. Para efecto del proyecto, quizá no sea tan necesario dar conexión a internet a los clientes ya que el tráfico que se analizará será local pero se propondrá como un servicio extra ya puede utilizarse internet para generar más tráfico.

El proceso de creación de un punto de acceso no es del todo complicado. Lo primero que hay que hacer es poner a trabajar a la tarjeta inalámbrica en modo en punto de acceso. Para esto, se utiliza el comando wlanconfig, el cual se encuentra en el directorio tools/ del código fuente del driver MadWiFi y que se instala de forma conjunta cuando el driver mismo se instala en la forma ya especificada. El comando wlanconfig funciona para crear, destruir y manipular interfaces virtuales MADWiFi o Virtual Access Points (VAP); esto quiere decir que principalmente se ocupa para crear interfaces en modo de punto de acceso, pero puede también crear interfaces en modo cliente. Una VAP es una instancia de una interfaz en modo cliente o punto de acceso y se puede crear más de una interfaz en diferentes modos; para esta

finalidad, sólo se ocupará la tarjeta trabajando en un modo particular: punto de acceso.

Las tarjetas inalámbricas instalados con el driver MADWiFi no pueden cambiar su modo de operación con el comando iwconfig, es por ello que se utiliza este comando como herramienta alternativa, de cualquier modo, el resultado es el mismo. Para manejar una interfaz inalámbrica en modo punto de acceso se utiliza los comandos siguientes:

ifconfig ath0 down

wlanconfig ath0 destroy

wlanconfig ath0 create wlandev wifi0 wlanmode ap

El primero de ellos servirá para dar de baja antes que todo, la interfaz que ya se instaló correctamente pero que está funcionando en modo cliente o estación; forzosamente se tiene que darse de baja para poder destruirla. Dar de baja una interfaz es como hacerla inactiva o deshabilitarla, de modo que quede en un estado en el que existe pero no se cuenta con ella para trabajar. El siguiente comando es destruir la interfaz, para que no se libere del sistema operativo y no exista por el momento ningún tipo de interfaz trabajando. Finalmente, se crea la interfaz inalámbrica en modo access point, o modo master, con el tercer comando. Tal comando tiene como parámetros la interfaz inalámbrica, el dispositivo base a través

del cual se creará la nueva interfaz y el modo en el cual trabajará, que en esta caso es con las letras ap (access point).

Con estos tres comandos fundamentales, tenemos una interfaz inalámbrica funcionando como punto de acceso. Pero aún falta un elemento fundamental, se debe asignar una dirección IP, una máscara de red y una dirección de broadcast para que los clientes se conecten, puedan trabajar en la red y accedan al servicio de internet que prestará esta estación base. La red de esta WLAN pertenece a direcciones privadas, en este caso a la red 192.168.10.0 con una máscara de red por defecto de este tipo de redes de clase C: 255.255.255.0. La dirección de broadcast será la 192.168.10.255. Como es deseable que cada vez que inicie la computadora, tenga ya esa dirección IP fija la interfaz inalámbrica y el modo de trabajo, entonces se procedió a unir el direccionamiento IP junto con la creación de la interfaz en modo master y se colocan ambos en un archivo de configuración muy importante para las interfaces de red bajo GNU/Linux. Este archivo de configuración se encuentra en la ruta */etc/network/interfaces* y contiene todo lo referente a la configuración de cada una de las interfaces existentes en el sistema como direccionamiento IP, la forma de obtener dirección de red, y en este caso la forma de trabajar (master), el canal de comunicación (canal número 11); también se puede observar en el archivo mostrado abajo, que el identificador para el punto de accesso es denominado DRAP.

Después de esto se coloca todos los datos del direccionamiento de la red inalámbrica de área local. Finalmente, otro aspecto relevante de este archivo, es que al final del mismo se colocan las instrucciones para crear la interfaz en modo access point antecedidas por la instrucción pre-up, con lo cual se está diciendo al archivo de configuración que antes que cualquier otra cosa, ejecute esas tres instrucciones. A continuación se muestra el archivo de configuración utilizado para poner a punto la interfaz inalámbrica.

auto lo
iface lo inet loopback
iface eth0 inet static

auto ath0
iface ath0 inet static
wireless-mode master
wireless-channel 11
wireless-essid DRAP

address 192.168.10.2
netmask 255.255.255.0
broadcast 192.168.10.255

pre-up ifconfig ath0 down
pre-up wlanconfig ath0 destroy
pre-up wlanconfig ath0 create wlandev wifi0 wlanmode ap

Posteriormente, para proporcionar a los clientes conexión a internet se instala un servidor proxy, el cual hace referencia a un programa o dispositivo que realiza una acción en representación de otro. En este caso un servidor proxy servirá para permitir el acceso a internet a todos los clientes cuando sólo se puede disponer de un único equipo conectado, esto es, una única dirección de red. Como el objetivo de esta tesis

no es el de explicar cómo se configura a detalle un servidor proxy, no se entrará a fondo. Para instalar un proxy en Debian se utiliza el comando apt-get install seguido del nombre del paquete; squid es el nombre del proxy más utilizado en los sistemas GNU/Linux. Después de la instalación del mismo, se configuró de forma sencilla para que cualquier cliente de la red tenga acceso a internet sin ningún tipo de prohibición.

Finalmente, para que un cliente no tenga que configurar cada uno de sus navegadores o aplicaciones que requieran internet, se creó un proxy transparente.

El concepto de transparente se refiere a que los usuarios no se den cuenta que están bajo un proxy, dando la apariencia de que están conectados directamente a internet. Para esto se requiere la ayuda de iptables, el firewall por defecto de Linux y tablas de traducción de direcciones de red: NAT (Network Address Translation). Como tampoco es la finalidad explicar cómo se configura un proxy transparente a través de la traducción de direcciones de red entonces se da por terminado la creación y puesta a punto de un hotspot bajo GNU/Linux.

3.4.2. Creación de middleware para calidad de servicio

Después de haber configurado el hotspot, el paso que sigue es la configuración del middleware que interactúe con el kernel para que pueda proporcionar QoS. Para

tal configuración se hará uso del marcado de paquete y disciplinas de colas. El middleware a implementar pretende hacer una distinción entre el trafico UDP del trafico TCP y más específicamente entre los paquetes de audio y vídeo, generados por un servidor de voz sobre IP, con lo cual se dará un trato diferente a cada uno de estos.

I. La distinción del tráfico se hará mediante el marcado de los paquetes que se generan en el hotspot. La distinción del tráfico se basa en diferenciar el trafico UDP del TCP. A su vez el tráfico UDP será dividido en dos tipos: audio y vídeo. La división del trafico UDP depende del tamaño de los paquetes. En base a los análisis realizados mediante Wireshark, uno de las herramientas de rastreo de tráfico más potentes que existen, se concluyó que el tamaño de los paquetes de audio se encuentra entre el rango de 64 y 300 bytes, mientras que los paquetes de vídeo serán aquellos que oscilen entre 301 y 1500 bytes. El marcado de paquetes se realizará mediante iptables, el cual permite hacer la distinción del tráfico dependiendo del protocolo, tamaño del paquete y puertos (origen y destino). Cuando se realice el marcado de los paquetes se modificará el campo TOS para asignarles prioridades diferentes, y así dar un trato diferente a cada unos de estos. Los valores que se asignan al campo TOS según el tipo de tráfico se presentan en la tabla III.2.

Tabla III.1 Prioridades de paquetes

Solo distinción de tráfico TCP del UDP		
0x10	Mínimo Retardo	Se asignará este valor al campo TOS a todos los paquetes UDP sin distinción de paquetes de audio y video
0x08	Máximo Rendimiento	Se asignará este valor al campo TOS a todos los paquetes TCP
Distinción de paquetes de Audio y Video		
0x10	Mínimo Retardo	Se asignará este valor al campo TOS a todos los paquetes UDP que tengan un tamaño entre el rango 64 y 300 bytes
0x04	Máximo Retardo	Se asignará este valor al campo TOS a todos los paquetes UDP que tengan un tamaño entre el rango 301 y 1500 bytes
0x08	Máximo Rendimiento	Se asignará este valor al campo TOS a todos los paquetes TCP
Fuente: Autores		

Al tener marcado los paquetes que viajan por la red, se pueden implementar disciplinas de colas para manejar de diferente forma los paquetes dependiendo del valor que contenga el campo TOS. Las disciplinas de colas que se manejarán para la implementación del middleware son dos: PRIO y HTB, ambas son disciplinas de colas con clases. Los paquetes de audio tendrán la más alta prioridad, con lo anterior

se garantiza que en una conferencia el audio siempre sea constante. Todas estas implementaciones de schedulers o colas se realizan, por supuesto, en la interfaz de red inalámbrica que está conectada a los clientes y que es la que necesita proporcionar calidad de servicio.

La disciplina de cola PRIO es una cola de prioridad absoluta. PRIO maneja tres bandas, en donde las bandas de prioridad menor no podrán transmitir datos mientras las bandas de mayor prioridad contengan datos. La banda número 0 (Alta prioridad) contendrá todos los paquetes que tengan en el campo TOS el valor 0x10, es decir, los paquetes de audio. Mientras que la banda número 1 (Prioridad media) almacenará todos los paquetes con valor TOS 0x04 que son los paquetes de vídeo. Por último la banda 3 (Baja prioridad) contendrá todos los paquetes TCP, que tienen en el campo TOS el valor 0x08. Cabe aclarar que al hablar de disciplinas de colas PRIO, se está haciendo referencia a la primera aportación realizada en el transcurso del proceso de investigación de DRAP, el cual se basa en la implementación de colas de prioridad absoluta. A continuación se observa un ejemplo de cómo se configuró una disciplina de cola PRIO.

iptables -t mangle -A POSTROUTING -p tcp -j TOS --set-tos Maximize-Throughput

iptables -t mangle -A POSTROUTING -p tcp --sport 20:21 --dport 20:21 -j TOS --set-tos Maximize-Throughput

iptables -t mangle -A POSTROUTING -p icmp -j TOS --set-tos Maximize-Throughput

iptables -t mangle -A POSTROUTING -m length --length 50:300 -p udp -j TOS --set-tos Minimize-Delay

iptables -t mangle -A POSTROUTING -m length --length 301:1500 -p udp -j TOS --set-tos Maximize-Reliability

Primero se tienen que marcar los paquetes conforme a lo explicado. Esto se logra con la aplicación iptables y mediante la tabla mangle, la cual sirve para modificar ciertos campos de un paquete de red. En las instrucciones de arriba se marcan cada uno de los tipos utilizados durante la prueba de laboratorio, es decir, tráfico icmp y tcp, que incluye dentro de éste descargas mediante ftp, así como tráfico de audio y de vídeo. Algunos de ellos se distinguen por el puerto como el ftp y otros, como el audio y vídeo además de especificar el protocolo, se proporcionó un rango de tamaño de paquetes. Todo el procedimiento de marcado se hace en la fase de POSTROUTING, o lo que es lo mismo a justo antes de colocar los paquetes en las colas. Cuando simplemente se distingue el tráfico UDP y TCP, entonces se ejecutaron instrucciones como las siguientes, donde el UDP se clasifica en general:

iptables -t mangle -A POSTROUTING -p tcp -j TOS --set-tos Maximize-Throughput

iptables -t mangle -A POSTROUTING -p tcp --sport 20:21 --dport 20:21 -j TOS --set-tos Maximize-Throughput

iptables -t mangle -A POSTROUTING -p icmp -j TOS --set-tos Maximize-Throughput

iptables -t mangle -A POSTROUTING -p udp -j TOS --set-tos Minimize-Delay

Posteriormente, para la aplicación de una cola (PRIO) se utiliza la herramita tc. En las instrucciones que siguen se está agregando una disciplina de cola raíz a la interfaz inalámbrica ath0, y enseguida se atan filtros a tal disciplina mediante el parámetro *parent 1:* y se indica la condición de cada filtro. Las coincidencias son por el campo Type of Servicio con diferentes prioridades cada una. Las clases a las cuales se mandará el tráfico clasificado no se especifican porque se usan las bandas de las colas por defecto que el sistema Linux, y se comportarán como lo hace una disciplina de cola con prioridad absoluta:

```
tc qdisc add dev ath0 root handle 1: prio
tc filter add dev ath0 parent 1: prio 1 protocol ip u32 match ip tos 0x10
0xff flowid 1:1
tc filter add dev ath0 parent 1: prio 2 protocol ip u32 match ip tos 0x04
0xff flowid 1:2
tc filter add dev ath0 parent 1: prio 3 protocol ip u32 match ip tos 0x08
0xff flowid 1:3
```

La disciplina de cola HTB permite garantizar un ancho de banda a cada tipo de tráfico que pasa por la red del hotspot. HTB contará con tres clases que almacenará los tres tipos de tráfico que maneja el middleware (audio, vídeo y TCP o best-effot). La clase uno contendrá los paquetes de audio, en donde su campo TOS será igual al valor 0x10, la cual garantizará un 60 % del ancho de banda total de la red y tendrá la más alta prioridad.

Mientras que la segunda clase contendrá los paquetes de vídeo (TOS=0x04), la cual tendrá 30 % del ancho de banda total de la red. La tercera clase sólo contará con el 10

% del ancho de banda, por lo cual todos los paquetes TCP (TOS=0x08) estarán en esta clase. HTB permite ocupar más ancho de banda del que se defina en la clase, siempre y cuando las otras clases no lo estén ocupando, con lo anterior se puede decir que el tráfico TCP no puede usar el 10%, sino que puede aumentar cuando no existan paquetes UDP en la red. El marcado de los paquetes se realiza de la misma forma al diferenciar los paquetes de audio, vídeo y best-effort. Por lo tanto, lo único que resta por explicar es la aplicación de la disciplina de cola Hierarchical Token Bucket, que junto que el marcado de los paquetes de red conforma el middleware que se denomina "Semi Dual-Queue Rate-Controlled Access Point" o simplemente "SDRAP", por sus siglas en inglés. Las instrucciones pertinentes son las siguientes:

```
tc qdisc add dev ath0 root handle 1:0 htb default 13
tc class add dev ath0 parent 1:0 classid 1:1 htb rate 256kbps
tc class add dev ath0 parent 1:1 classid 1:11 htb rate 160kbps ceil
256kbps prio 0
tc class add dev ath0 parent 1:1 classid 1:12 htb rate 80kbps ceil
256kbps prio 1
tc class add dev ath0 parent 1:1 classid 1:13 htb rate 16kbps ceil
256kbps prio 2
tc qdisc add dev ath0 parent 1:11 handle 11: sfq perturb 10
tc qdisc add dev ath0 parent 1:12 handle 12: sfq perturb 10
tc qdisc add dev ath0 parent 1:13 handle 13: sfq perturb 10
tc filter add dev ath0 parent 1: prio 1 protocol ip u32 match ip tos 0x10
0xff flowid 1:11
tc filter add dev ath0 parent 1: prio 2 protocol ip u32 match ip tos 0x04
0xff flowid 1:12
tc filter add dev ath0 parent 1: prio 3 protocol ip u32 match ip tos 0x08
0xff flowid 1:13
```

Como primera instrucción fundamental, se agrega una disciplina de cola raíz que será de tipo HTB y que por defecto, contendrá una cola a donde irán todos los paquetes que no hayan sido marcados o clasificados, en este caso tendrá el identificador 13. En el segundo bloque de instrucciones se agrega las clases necesarias, las cuales servirán para definir la cantidad de ancho de banda que se asignará a cada flujo de datos. La primera de ellos gestionará todo el ancho de banda y será asignada directamente a la disciplina de cola razón, de ahí que se le agregue el parámetro *parent 1:0.* Las demás clases tendrán como clase padre a la que se acaba de definir y se les dará para el parámetro rate, la cantidad de ancho de banda disponible por flujo; el tráfico de vídeo contará con 160 de 256 kbps (definidas en kilobytes por segundo), 80 para el vídeo y 16 para el tráfico TCP, con sus respectivas prioridades. Si las primeras clases no usan su ancho de banda garantizado, la 1:13 podrá usarlo todo y llegar hasta 256kbps (que es el parámetro ceil), pero si alguna clase superior reclama su ancho de banda garantizado, la 1:13 deberá bajar a 16 kbps y dejar que las superiores tomen lo que les corresponde. La clase padre 1:1 se encargará de reasignar el ancho de banda que sobre en función de la prioridad de las hijas.

Después de crear las clases, lo que falta es crear disciplinas de colas que programen los paquetes, esto es, que a las clases en sí no les llegan paquetes porque que son simplemente gestoras y no colas en sí. Por consiguiente, se asignará una cola independiente a cada clase. Se quiere que dentro de una misma clase todos los paquetes sean iguales, por tanto se utilizan disciplinas de cola de tipo SFQ (Stochastic

Fairness Queuing), para que los paquetes sean clasificados por conexiones y se intente que "viajen juntas" en la medida de lo posible y de forma igualitaria entre todos. Cada 10 segundos, el algoritmo utilizado por esta cola realizará un ajuste si existe tráfico que está llegando y que no esté siendo atendido.

Finalmente, para los filtros, se aplican los mismos en la disciplina PRIO, dependiendo de las coincidencias que haya en el marcado, se enviarán a la clase y colas correspondiente

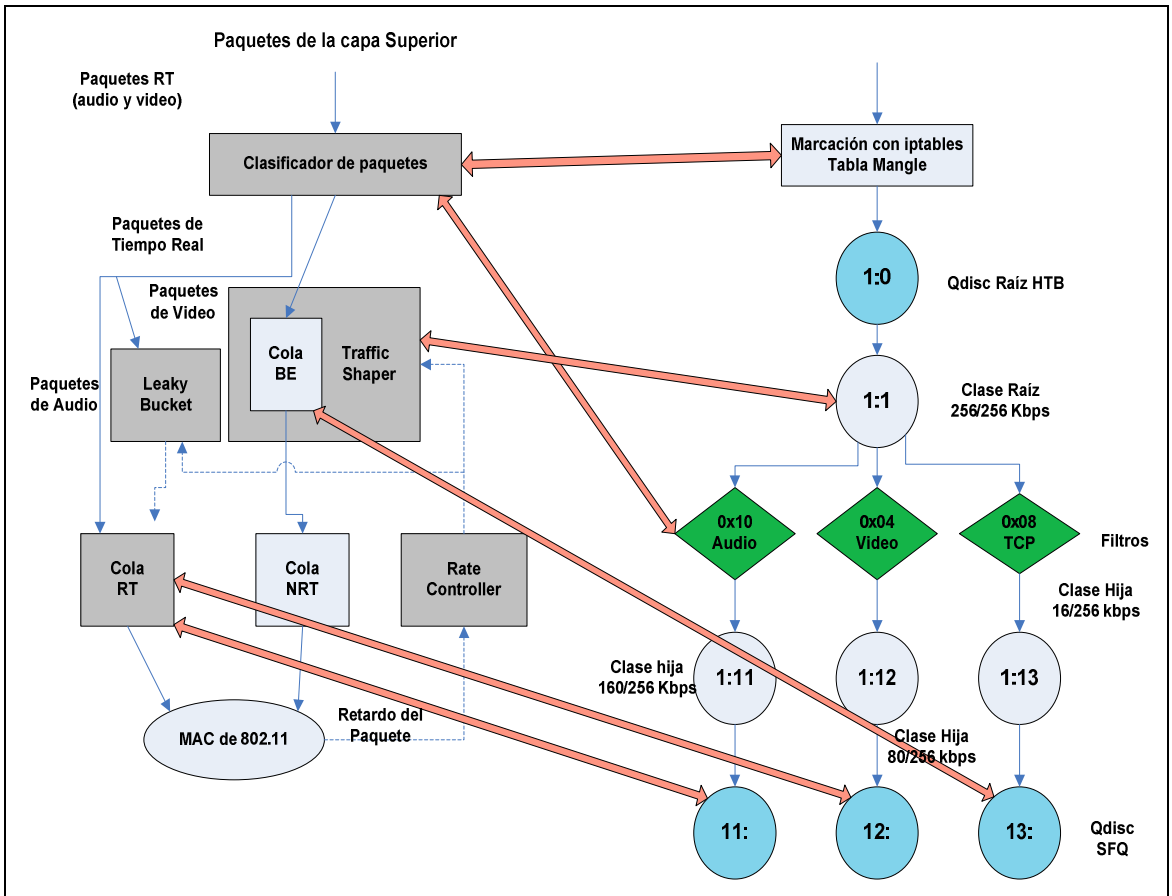


Figura III.22.- Comparativa entre DRAP y SDRAP

Se muestra el esquema de este middleware, además de hacer una comparativa con el modelo DRAP, del cual se basó este mecanismo SDRAP. Aquí se puede observar la correspondencia entre ambos modelos y los módulos faltantes, en este caso son el leaky bucket para ir descartando paquetes de video y el rate controller de forma dinámica.

3.5. Diseño e Implementación de un Testbed (escenario de pruebas) para Wlan con QoS

Después de realizar un middleware para calidad de servicio, siguiendo el modelo Dual-Queue Rate-Controlled Access Point, con todos los elementos que requiere, el paso siguiente es realizar las pruebas pertinentes para corroborar o refutar la hipótesis de esta tesis. Las pruebas se realizan implementando lo que se conoce como "testbed", algunas personas traducen esta palabra literalmente como una cama de prueba pero para efectos de este trabajo, se considera como una plataforma para experimentación para largos proyectos de desarrollo. Los testbed están hechos para teorías científicas rigurosas, transparentes y replicables, para herramientas computacionales y otras nuevas tecnologías".

3.5.1. Requisitos para el diseño de un testbed

Como ya se ha mencionado al inicio de este capítulo, se hace un enfoque respecto del concepto de un testbed y todo lo que implica. Las herramientas que se necesitan para

evaluar los datos que se obtienen con el testbed. Para obtener el retardo y la pérdida de paquetes son necesarias las siguientes herramientas: Tcpdump, Ethereal o Wireshark, Microsoft Access y Excel.

Tcpdump es una herramienta que permite capturar los paquetes que viajan a través de la WLAN. Es creada originalmente para Linux y es una aplicación muy potente para los fines que se ocupa. La captura de paquetes con tcpdump se puede realizar desde línea de comandos o con las herramientas gráficas que han surgido en base a este programa (Ethereal o Wireshark).

Ethereal o Wireshark es una herramienta gráfica diseñada para capturar los paquetes que viajan en la red de una manera fácil. Con esta herramienta sólo basta con elegir la interfaz de red en la que se quiere realizar la captura. Todos los paquetes capturados son almacenados en un archivo con extensión *pcap.* Con esta herramienta es muy fácil ver los campos que conforman el paquete IP, algo que con sólo tcpdump es más difícil. Mediante esta herramienta se hará el análisis del etiquetado de cada uno de los paquetes además de la perdida y el jitter.

Microsoft Access, una aplicación de base de datos, en este testbed es necesario para importar los archivos txt generados del análisis de paquetes RTP, la importación se realiza para crear tablas para su posterior análisis. Con las tablas creadas se podrá obtener el promedio, máximo y mínimo de los jitters capturados.

3.5.2. Diseño de un testbed

Teniendo los elementos necesarios para realizar la prueba, el siguiente paso es diseñar e implementar una prueba de laboratorio, con el fin de probar la funcionalidad del middleware. El diseño de la prueba requiere de los siguientes elementos:

- Una computadora de escritorio funcionando como punto de acceso y que esté corriendo el sistema operativo GNU/Linux Debian etch. Tal punto de acceso debe tener dos interfaces de red, una por la cual se recibe internet y otra a través de la cual se proporciona conectividad e internet a los clientes asociados a la red. En este caso se tiene una interfaz ethernet y una inalámbrica.

- Mínimo 3 computadoras clientes, laptop o de escritorio, con tarjetas inalámbricas funcionando en modo infraestructura y configuradas previamente para que trabajen a una velocidad de 54 Mbps, es decir, bajo el estándar IEEE 802.11g. Corriendo bajo los sistemas Linux o Windows.

- Al menos dos clientes que tenga instalado algún cliente de voz IP, como puede ser x-lite, wengo, twinkle, etc., y que cuenten por obvias razones con cámara web y micrófono correctamente configurados.

- Un servidor PBX (Private Branch Exchange), instalado en la estación base y con una configuración de lo más simple y dos usuarios al menos dados de alta y configurados.

- Un servicio generador de caracteres (chargen) corriendo bajo la estación base. El servicio chargen es un protocolo de internet definido en el RFC 864, y tiene la finalidad de realizar pruebas y mediciones. Un host puede conectar a un servidor que soporta este protocolo en el puerto 19. Una vez abierta la conexión, el servidor comienza a enviar de forma arbitraria caracteres hasta que el mismo host cierra la conexión o suceda algún evento que corte la conexión. Para su configuración, el servicio es habilitado en el archivo */etc/inetd.conf* y después de agregarle las siguientes líneas, se recarga el demonio inetd:

  chargen stream tcp nowait root internal

  chargen dgram udp wait root internal

- Un servidor apache versión 2 configurado por defecto, el cual contendrá en su directorio raíz un archivo de 100MB aproximadamente, el cual será descargado por todos los clientes conectados a la red al momento de realizar la prueba.

- Por último, un servidor FTP (ftpd) corriendo en el punto de acceso y para efectos de estas pruebas, configurado con las opciones por defecto para que los usuarios o clientes conectados, hagan descargas de archivos de más de 50 MB.

Lamentablemente, no entra en los alcances de este trabajo, el implementar la seguridad mínima pertinente en los diferentes servicios disponibles en la estación base pero sería un elemento extra en caso de que se configurara. Es importante señalar que todas las pruebas se realizaron bajo el estándar IEEE 802.11g, que trabaja a una velocidad de 54 Mbps (Megabits por segundo). Realmente estas velocidades son teóricas porque se calculan matemáticamente y es muy poco probable que realmente trabajen a tal velocidad. Por ejemplo, el estánda 802.11b trabaja a una velocidad promedio en flujos TCP de 5.9 y 7.1 para UDP. Ahora, se procede con los experimentos.

El escenario para esta prueba es el siguiente, completando algunos detalles más: se cuenta con una estación base, trabajando obviamente como punto de acceso a una velocidad de transferencia (teórica) de 54 Mbps. Esta estación comparte internet a sus clientes mediante un proxy transparente explicado antes. La red bajo la cual la estación base recibe internet es la 172.30.131.0 con una máscara de subred de 23 bits, no hay mucho que explicar de esta red porque es la de la Escuela Superior Politécnica de Chimborazo que utiliza para conectar todos las facultades y proveerlos de servicios como internet.

La red bajo la cual los clientes se conectan, es un rango de direcciones privadas, pudiendo obtener desde la dirección 192.168.10.100 hasta la 192.168.10.200 (sin utilizar las direcciones que finalizan en 0 y 255, de red y broadcast respectivamente). La interfaz inalámbrica se le asignó la dirección 192.168.10.2 y los clientes se les asignaron las direcciones mediante el servicio de DHCP(Protocolo Configuración Dinámica de Servidor). También es necesario hacer la aclaración que el identificador o ESSID de la red bajo la cual se trabajará será conocido como "DRAP", ahora, en cuanto al tipo de tráfico que generará cada cliente y el punto de acceso será de la siguiente forma. Se tienen cuatro estaciones clientes conectadas al punto de acceso, tres de ellas se conectarán al servidor ftp previamente instalado, y realizarán dos descargas cada uno, de un archivo aproximadamente 100 MB de tamaño. Esto mediante la línea de comandos, siendo indistinta la plataforma desde donde se conectan. Por ejemplo, teclearán el comando *ftp* 192.168.10.2 y podrán ingresar con el usuario "jorge" y el password "jorge". Con esto generarán tráfico de tipo TCP pero específicamente bajo el protocolo FTP. También, todos los clientes realizarán una descargada cada uno del archivo de 100 MB colocado en el servidor apache versión 2. Esto se puede realizar también desde la línea de comandos en Linux con wget *192.168.10.2/traficoTCP.zip*, o mediante un navegador Web tecleando la dirección *192.168.10.2/traficoTCP.zip*.

Para terminar la generación de tráfico TCP, en tres de los clientes se iniciarán 10 sesiones telnet hacia el puerto 19, es decir, al puerto del protocolo chargen para que

el servidor arroje caracteres hacia los clientes y por tanto, tráfico TCP puro. Esto se realiza con el comando telnet 192.168.10.2 19, en instantes se puede observar cómo aparecen caracteres en la pantalla de forma muy rápida.

Finalmente, al menos dos de los clientes realizarán una llamada entre ellos con la finalidad de probar el rendimiento de la red que se explicará abajo acerca de tráfico UDP, el cual pasará por el punto de acceso y de ahí a su destino en el otro cliente. La conversación con voz y vídeo se logra mediante algún cliente de telefonía. Para ello hay que configurar cada uno con el usuario *jorge* y otro con *vero,* la dirección del servidor, las contraseñas y los números telefónicos de cada uno: 6000 y 6001. De esta forma se iniciará la transmisión de paquetes de tipo UDP tanto de voz como de vídeo (ver figura III.23).

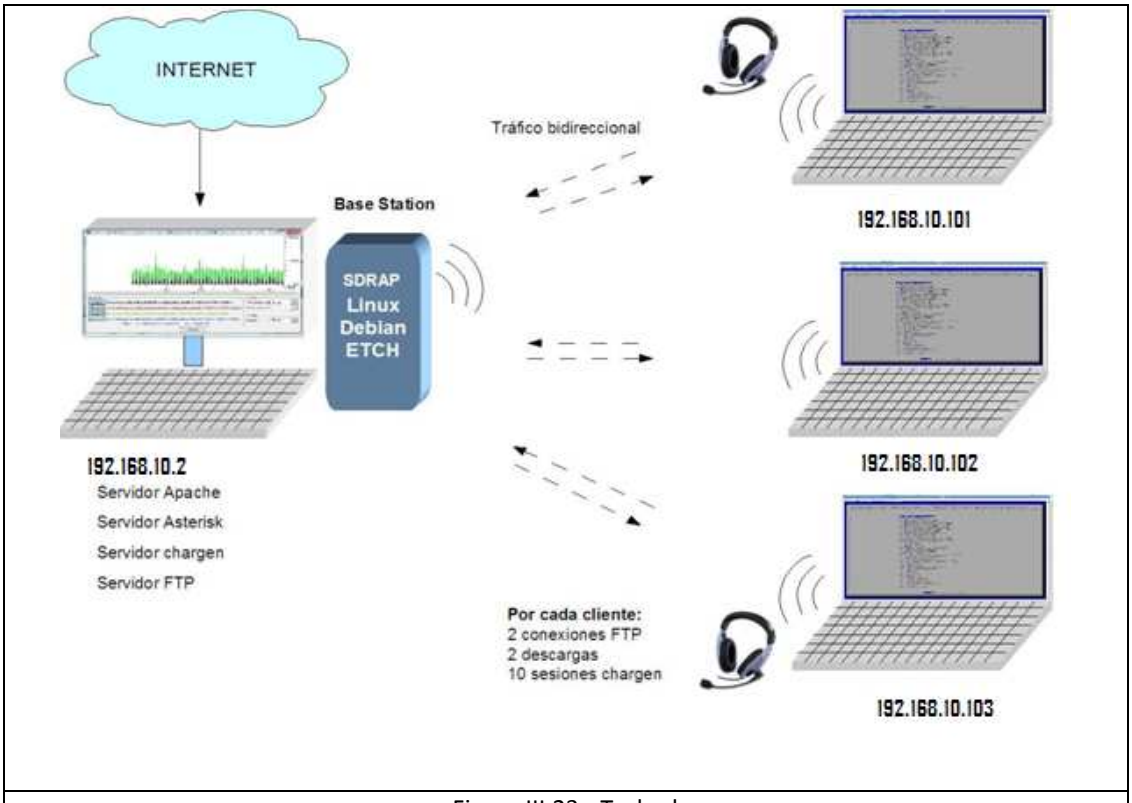


Figura III.23.- Tesbed

3.5.3. Metodología del Tesbed.

Los pasos o metodología que se siguen para analizar los datos generados en el testbed son explicados cada uno a continuación.

Considerando que el testbed ya ha sido configurado, es decir, se han configurado las PC con el hotspot, iniciado las sesiones al servidor Asterix y sincronizados los relojes, lo que prosigue es realizar la captura con wireshark tanto en los clientes entre lo que se establece la video llamada y el hotspot. La captura debe ser iniciada y finalizada al mismo tiempo tanto en los clientes como en el punto de acceso. La captura tiene una duración aproximada de 5 minutos, en donde en ambos clientes se está hablando sin parar durante este tiempo. Con lo anterior lo que se pretende es saturar el canal de paquetes de UDP.

Al finalizar la captura se procede a realizar el análisis mediante la conversión de todos los paquetes UDP a formato RTP, de un flujo en particular, (Real Time Protocol), con la conversión anterior se obtendrá la marcación correcta de los paquetes así como también el porcentaje de pérdidas y las estadísticas del jitter. Para la codificación sólo basta con seleccionar un paquete UDP, dar un click derecho sobre éste y seleccionar la opción “Decode As…”. Al realizar lo anterior se abrirá la ventana que se muestra en la figura 3.12. En esta ventana se selecciona el tipo de codificación a realizar, que en este caso es RTP y dar click en OK. Cabe mencionar que no todos los paquetes UDP se codifican, por ejemplo, si se escoge un paquete con la dirección de origen 192.168.10.101 y destino 192.168.10.116, se

codificarán todos los que venga de este origen más el flujo contrario pero no los de la dirección 192.168.10.115. Es una codificación bidireccional entre pares de direcciones.

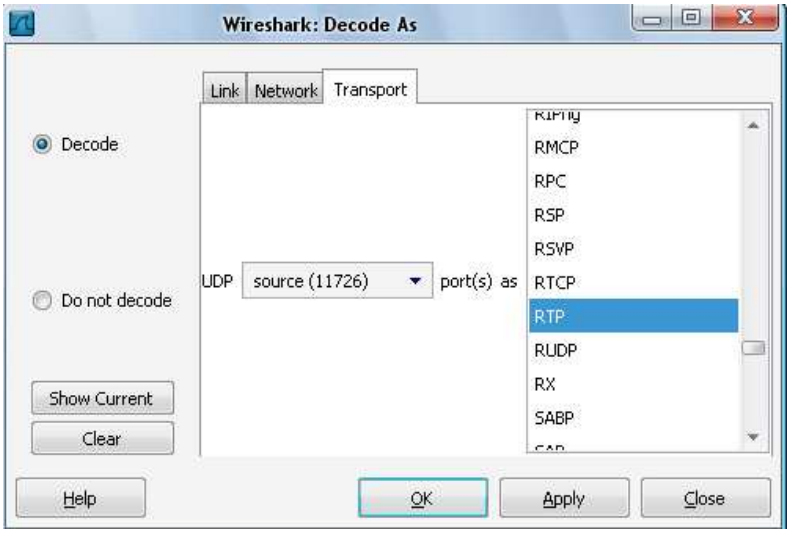


Figura III.24.- Ventana de codificación de Wireshark

Teniendo convertidos los paquetes UDP a RTP se puede realizar análisis sobre estos datos, con este análisis se puede obtener el porcentaje de paquetes perdidos y el jitter. Los datos anteriores se obtienen al seleccionar un paquete RTP, dar click en la opción de "Statistics"→ "RTP" →"Stream Analysis…" (Ver figura III.24).

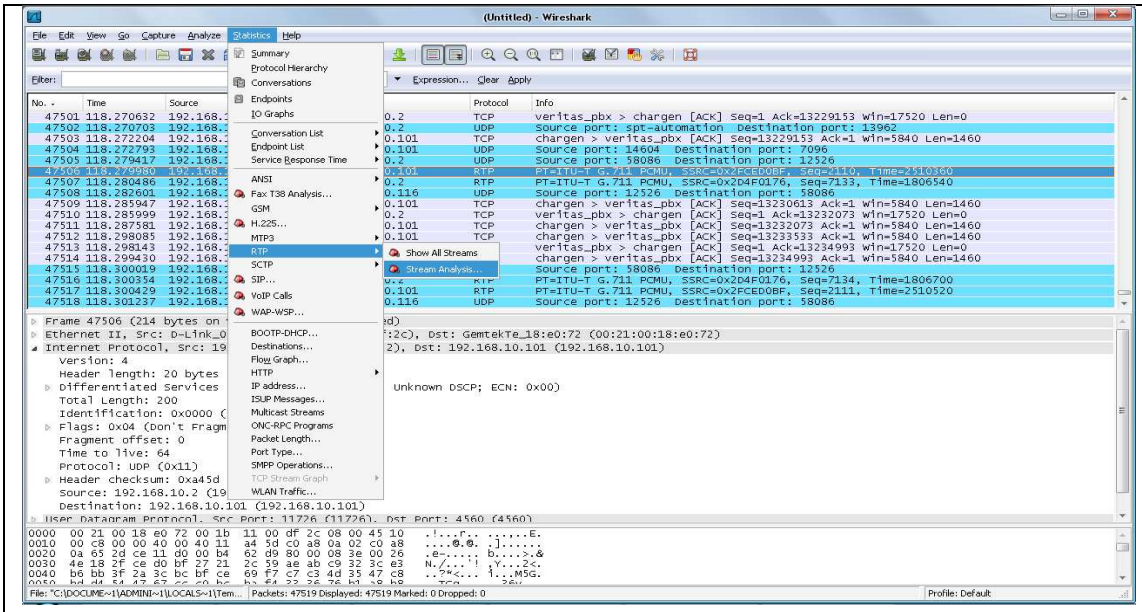


Figura III.25. Análisis de paquetes RTP

El cálculo del jitter es calculado en base al campo timestamp y el tiempo del paquete. El timestamp es obtenido en base a la frecuencia del codec[17] que se maneja, el cual es de 8000 en la mayoría de los codecs de audio y 90000 en los de vídeo. Después de aplicar la formula se obtiene la ventana mostrada en la figura III.26, donde se puede observar el jitter de los paquetes capturados en la comunicación establecida y el número de paquetes perdidos.

[17] Abreviatura de Codificador-Decodificador. Describe una especificación desarrollada en software, hardware o una combinación de ambos, capaz de transformar un archivo con un flujo de datos o una señal.

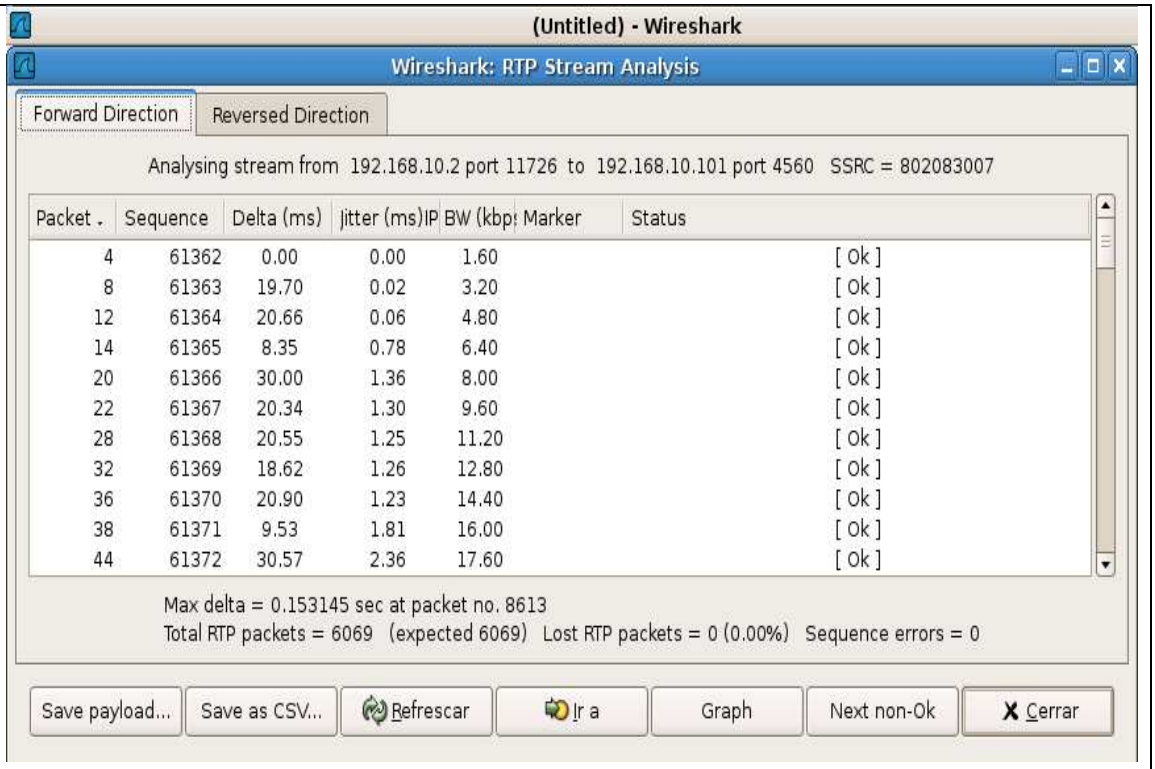


Figura III.26.- Ventana de jitters y pérdidas de paquetes

Al obtener la ventana anterior, lo que sigue es guardar el archivo con extensión txt.

Para lo anterior sólo hay que dar click en el botón “Save as CVS…”, el archivo generado contendrá todas las columnas que se observan en la figura III.26. El archivo txt generado debe ser importando a Excel para obtener le promedio, máximo y mínimo de los jitters.

CAPÍTULO IV

ANÁLISIS Y RESULTADOS

A continuación, se mostrarán todos los resultados de las pruebas realizadas de la implementación de este prototipo, basado en algunos aspectos del mecanismo DRAP, incluyendo gráficas y análisis de las mismas. Se han probado dos diferentes escenarios:

- Escenario con el estándar PCF (sin colas)
- Escenario con DRAP (cola HTB)

En estos dos escenarios, los pasos que se siguen y la metodología es exactamente la misma, al igual que todo el contexto de comunicación, excepto por las disciplinas de colas aplicadas a la interfaz inalámbrica del punto de acceso. Particularmente, los

dos datos más importantes que se requieren para la evaluación de resultados son el jitter o variación en el retardo y el packet loss o pérdida de paquetes, de ahí se pueden obtener gráficas y encontrar datos muy relevantes como puede ser el número de paquetes perdidos durante una conexión o comunicación con voz y vídeo en diferentes circunstancias. Los puntos de referencia para considerar si existe QoS en base al jitter y delay son los que se muestran en la tabla IV.3, particularmente el de 64 kb/s que es el de una videoconferencia. Mientras que el parámetro de paquetes perdidos necesario para que exista QoS siempre tiene que ser menor al 1%.

Tabla IV.3 Requerimientos de jitter y delay

APLICACIÓN	DELAY	JITTER (MS)
64kb/s Videoconferencia	300	130
1.5 Mb/s MPEG NTSC video	5	6.5
20 MB/s HDTV video	0.8	1
16Kb/s voz comprimida	30	130
256 Kb/s voz MPEG	7	9.1
Fuente: www.eslared.org		

4.1. Resultados en escenario con el estándar PCF.

Se comenzará a explicar un poco el escenario PCF. Este es el escenario más común y en bruto, si se le puede llamar de esa forma. Se requiere hacer pruebas con esta forma porque se necesitará después comparar y ver los resultados que arroja el

mecanismo implementado. En este contexto trabajan actualmente todas las redes inalámbricas de área local y específicamente las que son de tipo infraestructura. Es una situación de lo más simple porque no hay ningún elemento extra que trate de garantizar ancho de banda y calidad de servicio más que el mismo ancho de banda podría ser el límite para una eventual saturación de tráfico.

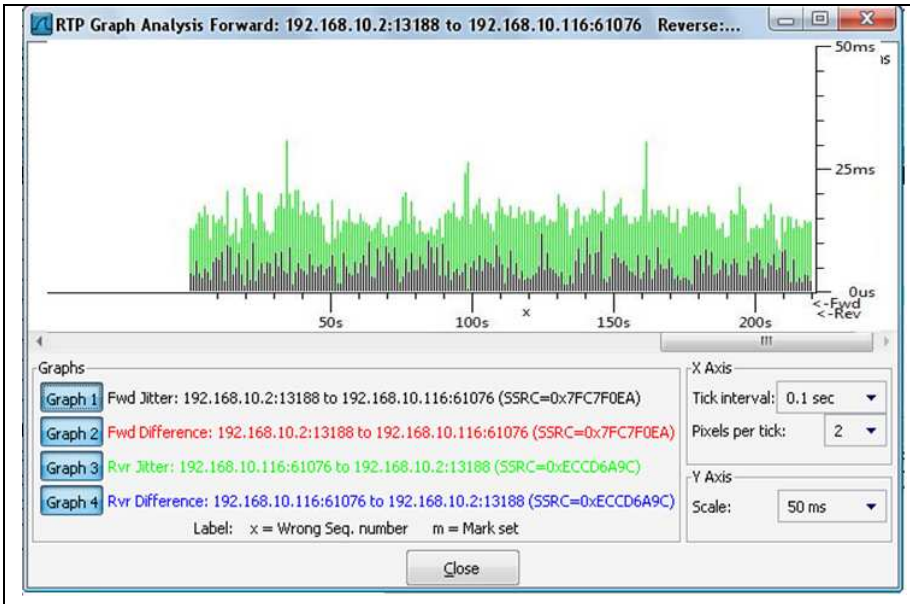


Figura IV.27.- Prueba 1 Estación Base

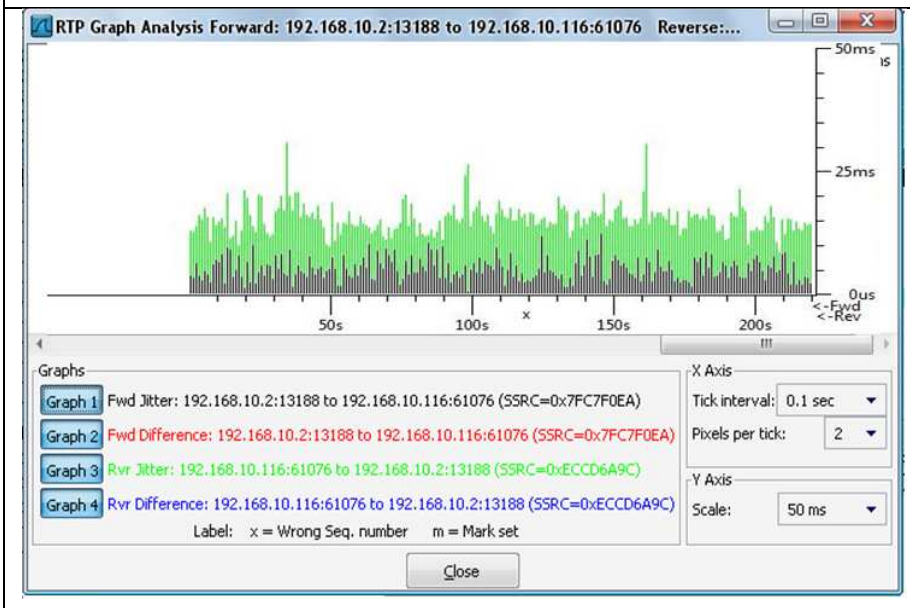


Figura IV.28.- Prueba 1 PC1

En las figuras anteriores se puede observar claramente que no se aplicó una disciplina de colas la variación de los jitter es muy clara, además se muestra que la mayoría se encuentra en los 15 ms. Respecto a la pérdida de paquetes, en un ambiente sin disciplinas de cola, se obtuvieron resultados de pérdida de paquetes de 87 y 85 % respectivamente para los flujos de tráfico UDP de la computadora 192.168.10.2 hacia la 192.168.10.116. Lo que quiere decir que en esos lapsos, se consiguió que la interfaz se saturara de tráfico de todo tipo y perdiera paquetes en demasía.

Los resultados obtenidos con wireshark de estas pruebas para obtener el jitter se presentan en la tabla IV.2

Tabla IV.2. Resultado de jitter con PCF

PCF	192.168.10.2-192.168.10.116	192.168.10.116 - 192.168.10.2	192.168.10.2 - 192.168.10.102	192.168.10.102-192.168.10.2
Promedio	11.9384	2.3267	13.7018	2.7640
Máximo	30.6	12.09	42.98	11.90
Mínimo	0	0	0	0

Fuente: Autores Tesis

4.2. Resultados en escenario con el mecanismo de Calidad de Servicio (SDRAP)

En la prueba 2 el tráfico de mayor prioridad fue el UDP con tres tipos de marcado, el tráfico de audio tenía un valor de 0x10, el vídeo 0x04 y el TCP 0x08, donde también el audio era el que tenia la mayor prioridad.

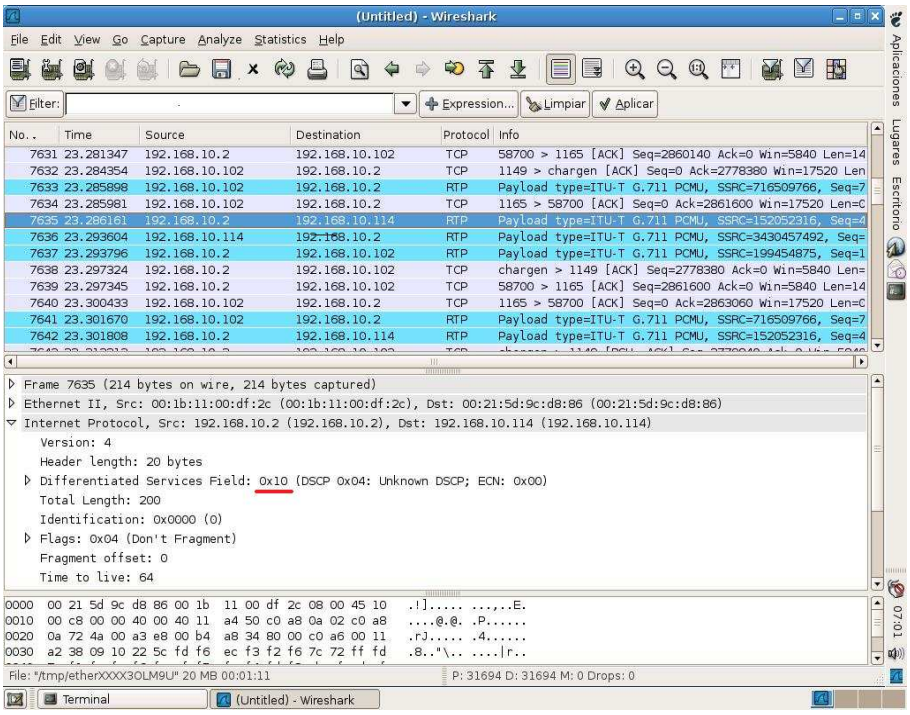


Figura IV.29.- Macado de paquetes de Audio

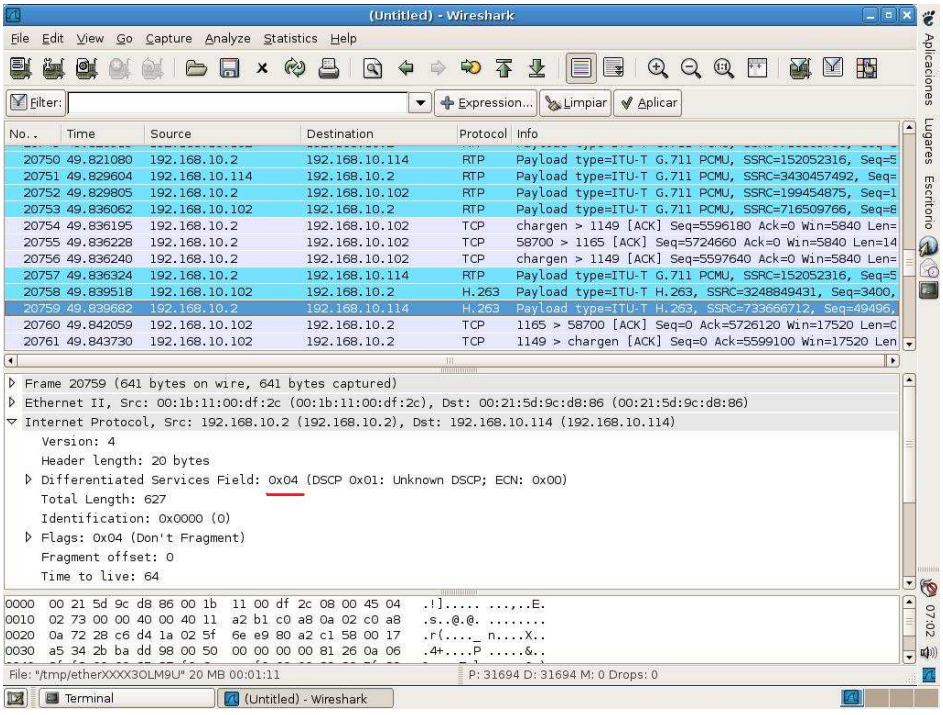


Figura IV.30.- Marcado de paquetes de Video

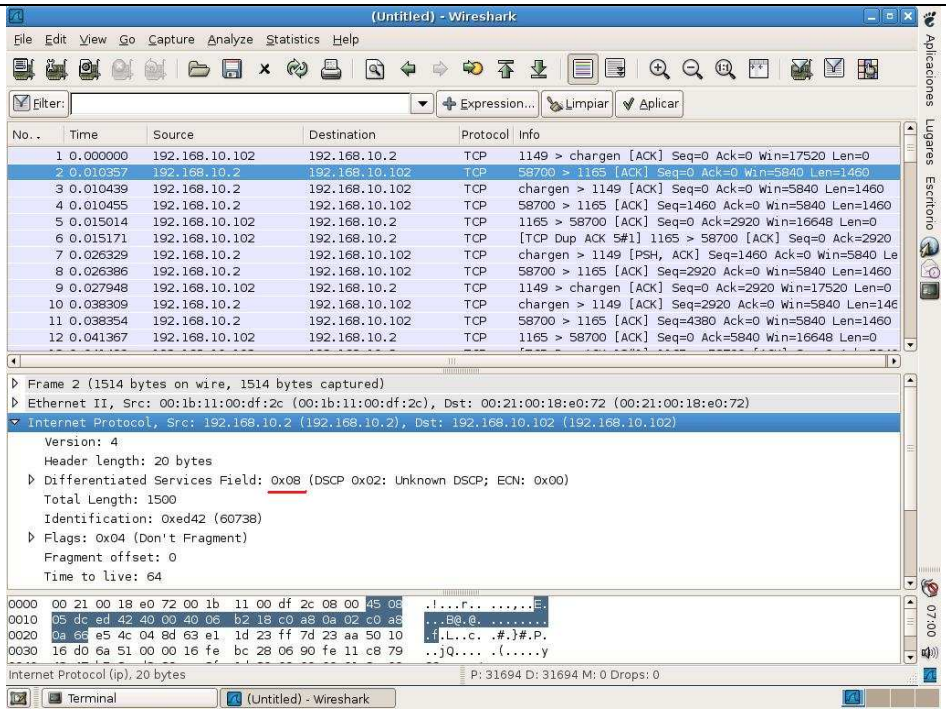


Figura IV.31.- Marcado de paquetes TCP

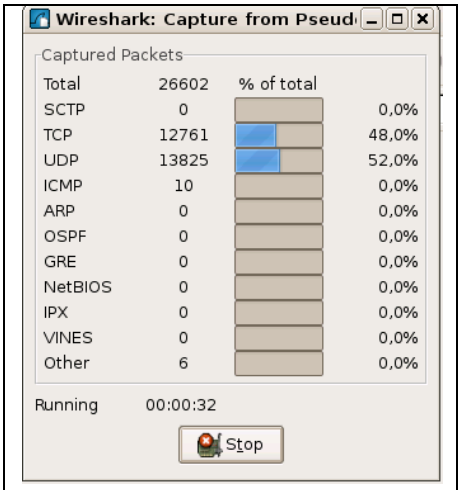


Figura IV.32.- Paquetes en la Red

HTB permite ocupar más ancho de banda del que se defina en la clase, siempre y cuando las otras clases no lo estén ocupando, con lo anterior se puede decir que el tráfico TCP puede usar mas del ancho de banda asignado, puede aumentar

cuando no existan paquetes UDP en la red, además existe una prioridad para los paquetes UDP con respecto a los TCP.

Se puede observar que la prueba 2 en donde se distinguió el audio del vídeo es la que cuenta con el promedio de jitters más bajo entre los clientes y el servidor que se estableció la videollamada. Además en esta prueba también se observa que fue la que tiene los valores máximos más pequeños que las otras dos pruebas. La prueba 1 en donde no se aplicó ninguna discliplina de colas se observa que es la que tienes los máximos más grandes, debido a que todo el tráfico que pasa por la dirección 192.168.10.2 que es el hotspot es tratado de igual manera.

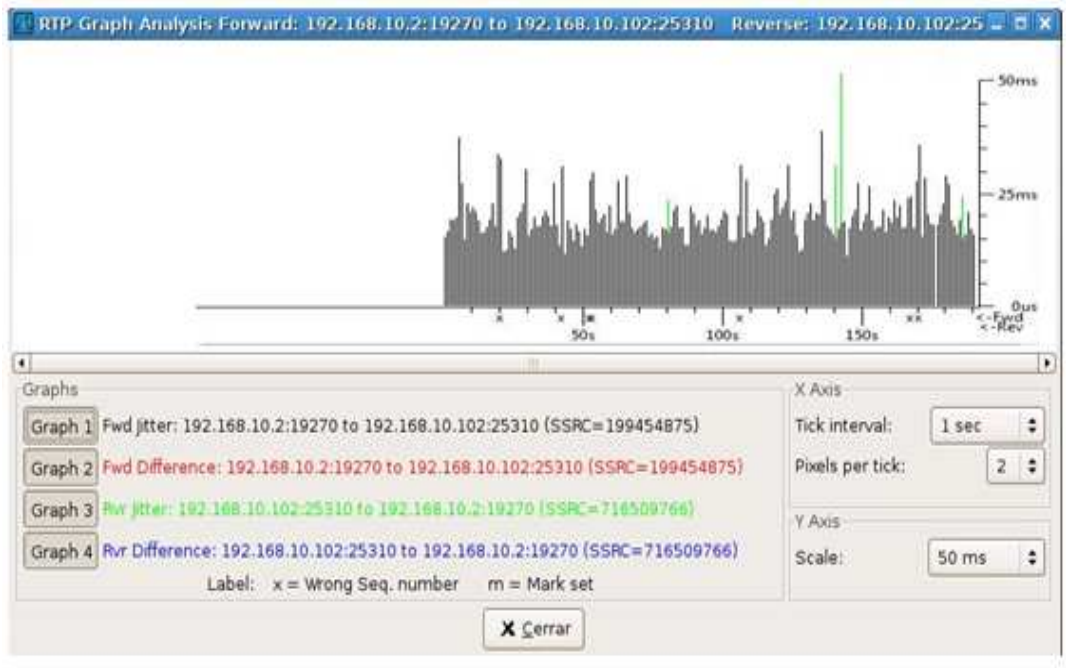


Figura IV.33.- Prueba 2 Estación Base

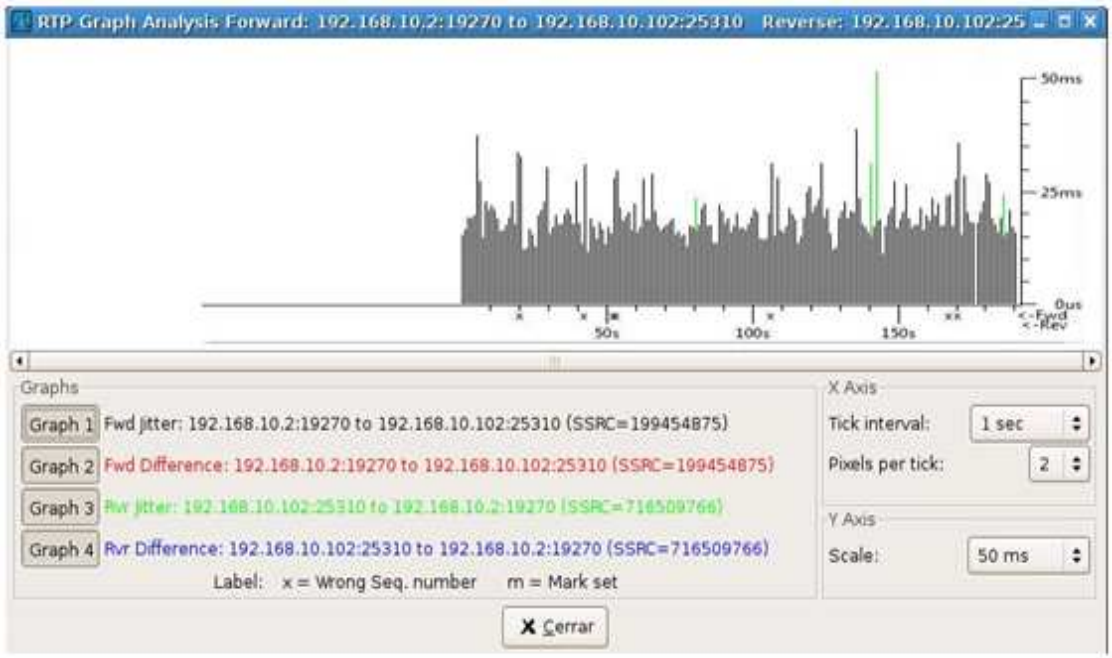


Figura IV.34.- Prueba 2 PC1

El resultado de las pruebas anteriores se puede apreciar más claramente en las figuras IV.33 y IV.34 en donde la variación de los jitter es la clave para determinar si hay una mejora cuando se aplica la disciplina de colas HTB (DRAP).

Tabla IV.5 Resultados de jitter SDRAP

SDRAP	192.168.10.2 - 192.168.10.102	192.168.10.102 - 192.168.10.2	192.168.10.2 - 192.168.10.116	192.168.10.116- 192.168.10.2
Promedio	2.8029	3.5513	4.6893	1.6654
Máximo	21.94	8.74	21.78	8.89
Mínimo	0	0	0	0

Fuente: Autores

Por lo tanto se puede decir que al aplicar el mecanismo de calidad de servicio basando en el algoritmo DRAP , sí se reduce considerablemente la variación del jitter, debido a que se gestiona un trato preferencial al tráfico de audio y vídeo y el ancho de banda se divide además en las diferentes prioridades asignadas, la calidad de servicio mejora considerablemente respecto a un ambiente sin disciplinas de cola. Como el trafico TCP es considerado el de menor prioridad y se le asigna un ancho de banda muy pequeño, se evita que compita por tener salida cuando existe trafico UDP, así el trafico UDP se favoreces por todos los filtros aplicados a él.

Por último, también puede observarse una pérdida de paquetes o packet loss casi nula cuando se aplica la disciplina de cola HTB, garantizando un ancho de banda determinado a cada flujo, en orden de importancia y prioridad.

Tabla IV.6 Determinación de Calidad de Servicio en Pruebas

			Prueba 1	*Prueba 2*
PC 1	*192.168.10.2*	*Jitter*	*30.6*	*21.94*
	192.168.10.116	*% Pérdida*	*0.09*	*0.02*
	192.168.10.116	*Jitter*	*12.9*	*8.74*
	192.168.10.2	*% Pérdida*	*0*	*0.09*
PC 2	*192.168.10.2*	*Jitter*	*42.98*	*21.78*
	192.168.10.102	*% Pérdida*	*87.21*	*0.02*
	192.168.10.102	*Jitter*	*11.90*	*8.89*
	192.168.10.2	*% Pérdida*	*0*	*0*
Calidad de Servicio			***NO***	***SI***
Fuente: Autores Tesis				

CONCLUSIONES

1) Hay mucho trabajo de investigación detrás del concepto calidad de servicio, quizá es un término que no es muy tratado en la rama de las redes y por tanto existe poca información al respecto.

2) El mecanismo o middleware desarrollado durante el transcurso del proyecto que está basado en algunos conceptos del mecanismo DRAP es un gran paso para cambiar los paradigmas de comunicación en lo que a redes inalámbricas se refiere.

3) El algoritmo DRAP desarrollado por el Maestro Rubén Álvaro González Benítez, fue una aportación fundamental para el desarrollo del presente trabajo, ya que se tomó como punto de partida para la priorización y marcado de paquetes UDP y su clasificación: audio y video.

4) Es perfectamente factible desarrollar un punto de acceso usando software libre para implementar redes inalámbricas, y éste puede implementar las mismas características que las comerciales, con dos ventajas: costo y facilidad de ampliar la capacidad del sistema.

5) La implementación de WLAN en software libre permite demostrar que se puede utilizar computadores de bajas características que usualmente se le puede llegar a considerar obsoletos.

6) Con los resultados arrojados se ha demostrado claramente que con un mecanismo como éste, se logra mejorar de manera considerable la satisfacción de un usuario que necesita una buena comunicación mediante voz y vídeo.

7) Para la implementación de este trabajo se requiere de un conocimiento a nivel medio-alto de las herramientas de software libre y su campo de aplicaciones.

8) Como la experiencia de trabajo obtenida, introducirse en las redes inalámbricas nos ha abierto un nuevo enfoque respecto a su funcionamiento interno y no sólo el externo. El investigar aspectos de los paquetes de red fuera de un ambiente escolar es de gran importancia para el conocimiento y superación personal y profesional.

RECOMENDACIONES

1) En este trabajo el shaper se implementó de otra forma con disciplinas de colas. Se puede también programar con algún script o adecuar el código fuente del kernel, específicamente de las disciplinas de cola para insertar el shaper deseado.

2) Es recomendable seguir incursionando en proyectos de software libre, para poder beneficiarnos de esta gran filosofía.

3) Se recomienda el estudio e implementación de sistemas para garantizar Calidad de Servicio en Redes Inalámbricas pues se tiene poca información del tema, y con ellos podríamos aportar a este tema de gran importancia pero todavía incógnito.

4) El middleware implementado fue lo más parecido a DRAP pero como proyecto de investigación habría que agregar otros mecanismos adicionales para que DRAP funcione y se adecue al modelo teórico.

RESUMEN

Se implementó un mecanismo de calidad de servicio para redes inalámbricas de área local de tipo infraestructura con software libre para garantizar el tráfico de aplicaciones en tiempo real (voz y video).

Fueron empleados métodos lógicos y sintéticos para la unión de los elementos. Empleando GNU/LINUX como sistema operativo en su distribución Debian etch 4.0 sobre un PC con plataforma x386 de 512 Mb de RAM, Pentium IV, tarjeta de red inalámbrica d-link dwl-g520 air plus, integrada con Chipset Atheros de la familia 5212, se procedió a su instalación compilando el kernel para dar soporte extensiones de administración de redes inalámbricas y de control de tráfico usando MADWIFI y Wireless Tools; para implementar el mecanismo con algoritmo DRAP se utilizó colas con clase (HTB y PRIO), colas sin clase (SFQ) y traffic shapping; en la clasificación de paquetes fue modificado el campo TOS mediante iptables.

La investigación demostró que el punto de acceso tiene la capacidad de brindar a los clientes accesibilidad a todos sus servicios en forma simultánea en un 90%, y con dicho mecanismo basado en DRAP se reservó un ancho de banda de 60% para audio, 30% video y 10% TCP; priorizando el tráfico multimedia sobre TCP, pero cuando no existen paquetes UDP en la red, el trafico en TCP puede aumentar y ocupar todo el ancho de banda existente. Las pruebas fueron realizadas en los servidores: VozIP, en el de caracteres y en el generador de archivos que saturaron la red y al realizar una video llamada entre dos clientes, la comunicación se mantuvo operativa.

El mecanismo de calidad de servicio basado en el algoritmo DRAP implementado, mejoró el tráfico multimedia en un 85% a la implementada en los sistemas PCF y DCF que vienen establecidos por defecto.

SUMARY

We implemented a mechanism for quality of service for local area wireless networks include infrastructure with free software to ensure application traffic in real time (voice and video).

Logical methods were used for binding and synthetic elements. Using GNU / Linux operating system on your Debian etch 4.0 on a PC platform x386 with 512 MB RAM, Pentium IV, wireless network card D-Link DWL-G520 Air Plus, integrated with Atheros chipset family 5212, installation proceeded to compile the kernel extensions to support wireless network management and traffic control using MADWIFI and Wireless Tools, to implement the mechanism used algorithm DRAP class queues (HTB and PRIO), classless queues (SFQ) and traffic shaping, packet classification in the TOS field was amended by iptables.

The investigation showed that the access point is able to provide customers with access to all services simultaneously by 90%, with RBAP-based mechanism that reserved a bandwidth of 60% for audio, 30% video and 10% TCP; prioritizing multimedia traffic over TCP, but when no UDP packets on the network, TCP traffic may increase and occupy all the existing bandwidth. The tests were performed on servers: VozIP, the character and the file generator that saturated the network and to place a video call between two clients, communication remained operational.

The QoS mechanism based on the algorithm implemented RBAP, improved multimedia traffic by 85% to the systems implemented in the PCF and DCF are set to default.

GLOSARIO

Beacon.- Señal no direccional transmitida de manera constante por un emisor en una determinada frecuencia cuya misión es informar de su presencia y permitir que los receptores puedan comunicarse con él.

Best-effort.- "el mejor esfuerzo", para definir la forma de prestar aquellos servicios para los que no existe una garantía de calidad de servicio (QoS). Esto implica que no existe una preasignación de recursos, ni plazos conocidos, ni garantía de recepción correcta de la información.

Celda.- Celda en Radiocomunicaciones, define el máximo espacio de cobertura de una estación transmisora dentro del cual todos los usuarios pueden conectarse hacer uso de los servicios que éste brinda.

Gestor de paquetes.- Es una colección de herramientas que sirven para automatizar el proceso de instalación, actualización, configuración y eliminación de paquetes de software frecuentemente encapsulado en un solo fichero.

Ipchains.- Es un cortafuegos libre para Linux. Es el código reescrito del código fuente del cortafuego IPv4 anterior de Linux.

Jitter.- Cambio o variación en cuanto a la cantidad de latencia entre paquetes de datos que se reciben. Por ejemplo, el jittering son los saltos que pueden dar los CDs al ser leídos.

Leaky-bucket.- Tiene varios usos, es mejor entendida en el contexto de tráfico de red de configuración o de limitación de velocidad. Normalmente, el algoritmo se utiliza para controlar la velocidad a la que los datos se inyectan en una red, suavizando "explosividad" en la tasa de datos.

Minix.- MINIX es un clon pequeño y gratuito de UNIX diseñado para una tener una fiabilidad muy alta. Es particularmente apropiado para PCs de bajo costo, sistemas con recursos limitados y aplicaciones embebidas.

Paquete.- Cada uno de los bloques en que se divide la información que se envía a través de una red en el nivel de red del modelo OSI. Por debajo de este nivel el paquete adquiere el nombre de trama de red.

Repositorio.- Sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos, están preparados para distribuirse habitualmente sirviéndose de una red informática como Internet o en un medio físico como un disco compacto.

Scripts.- Archivo de órdenes o archivo de procesamiento por lotes es un programa usualmente simple, que generalmente se almacena en un archivo de texto plano. Los script son casi siempre interpretados, pero no todo programa interpretado es considerado un script.

Sensibilidad de umbral.- El umbral es la cantidad mínima de señal que ha de estar presente para ser registrada por un sistema.

Thoughtput.- Es la efectiva velocidad de transferencia de datos entre dos computadoras, considerando la compresión de datos, la corrección de errores y eventualmente el tiempo pasado para la conexión.

Traffic shaper.- Se trata de una técnica que permite controlar el tráfico de una red de ordenadores para optimizar o garantizar su funcionamiento, las bajas latencias y/o el ancho de banda.

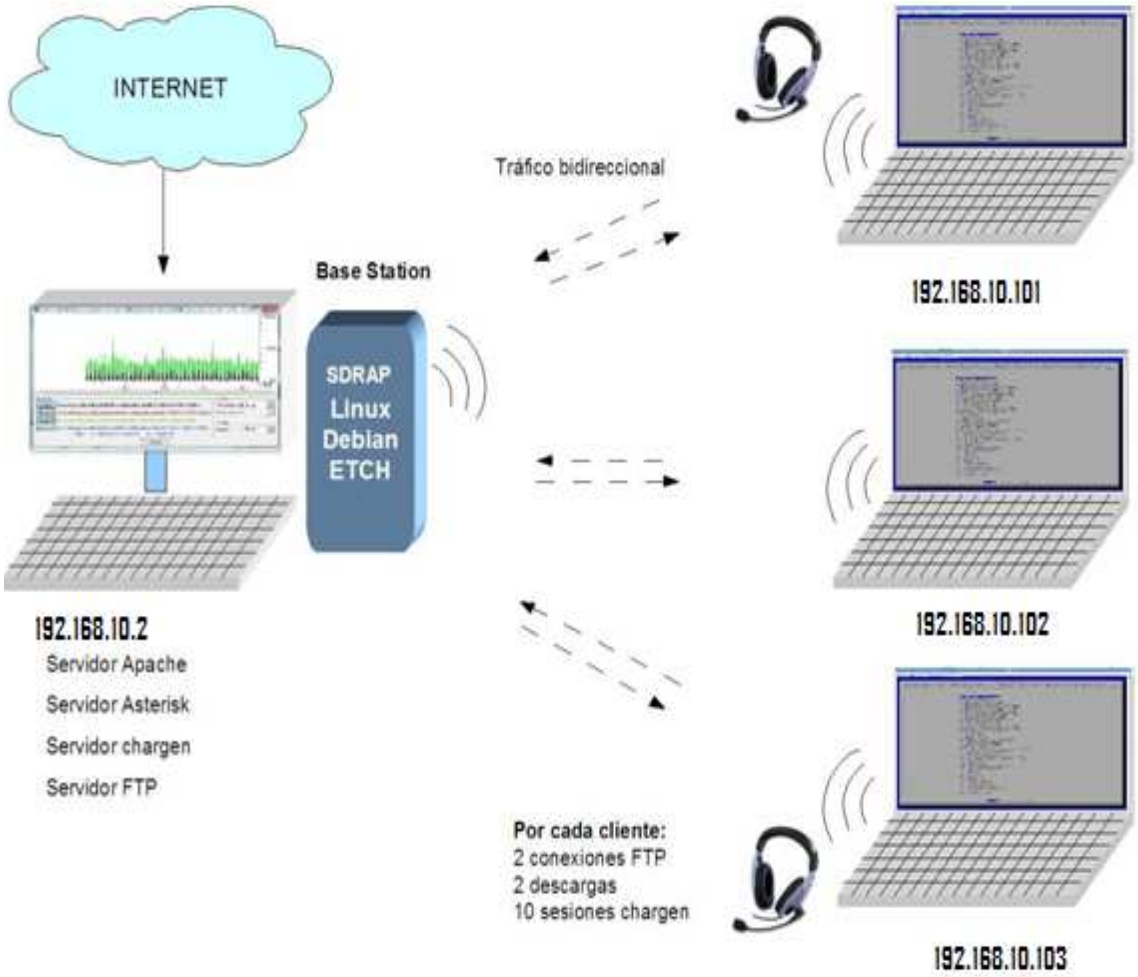
Wavelan.- es un sistema completo e integrado de hardware y software que puede añadir la conectividad inalámbrica a una LAN existente, ayudando a crear una red independiente al instante o incluso enlazar las redes locales de varios edificios.

Wifi.- Conjunto de estándares para redes inalámbricas basado en las especificaciones IEEE 802.11. No es acrónimo de “Wireless Fidelity”. Es una marca de WI-FI Alliance, la organización comercial que prueba y certifica que los equipos cumplan los estándares.

A N E X O S

ANEXO A

Escenario de Pruebas (Testbend)



ANEXO B

ARCHIVO DE CONFIGURACION DE INTERFACES

/etc/network/interfaces

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static

address 172.30.131.10
netmask 255.255.254.0
gateway 172.30.131.1


auto ath0
iface ath0 inet static
wireless-mode master
wireless-channel 11
wireless-essid DRAP

address 192.168.10.2
netmask 255.255.255.0
broadcast 192.168.10.255

pre-up ifconfig ath0 down
pre-up wlanconfig ath0 destroy
pre-up wlanconfig ath0 create wlandev wifi0 wlanmode ap
```

ANEXO C

ARCHIVO DE CONFIGURACION DHCP

/etc/dhcpd.conf

```
#Asignación de direcciones de red por DHCP
subnet 172.30.200.0 netmask 255.255.255.0 {
	range 172.30.200.100 172.30.200.110;
	option broadcast-address 172.30.200.255;
	option domain-name-servers 172.30.60.5, 172.30.60.32;
	option routers 172.30.200.1;
}
dhcpd.conf (END)
```

ANEXO D

ARCHIVO DE CONFIGURACION DE FTP

/etc/proftpd.conf

```
# Includes DSO modules
Include /etc/proftpd/modules.conf

# Set off to disable IPv6 support which is annoying on IPv4 only boxes.
UseIPv6                 off

ServerName              "Debian"
ServerType              standalone
DeferWelcome            on

MultilineRFC2228        on
DefaultServer           on
ShowSymlinks            off

TimeoutNoTransfer       600
TimeoutStalled          600
TimeoutIdle             1200

DisplayLogin            welcome.msg
DisplayFirstChdir       .message
ListOptions             "-l"

DenyFilter              \*.*/

# Port 21 is the standard FTP port.
Port                    21
```

ANEXO E

ARCHIVO DE CONFIGURACIÓN DEL PROXY

/etc/squid/squid.conf

```
http_port 192.168.10.2:3128 transparent

visible_hostname debian

hierarchy_stoplist cgi-bin ?

#We recommend you to use the following two lines.
acl QUERY urlpath_regex cgi-bin \?
cache deny QUERY

#acl apache rep_header Server ^Apache
#broken_vary_encoding allow apache


#Default:
cache_mem 16 MB

cache_dir ufs /var/spool/squid 100 16 256

hosts_file /etc/hosts

#auth_param basic program /usr/lib/squid/ncsa_auth /etc/squid/claves
auth_param basic children 5
auth_param basic realm Squid proxy-caching web server Servidor de cache
auth_param basic credentialsttl 2 hours
auth_param basic casesensitive on

#Suggested default:
refresh_pattern ^ftp:    1440  20%  10080
refresh_pattern ^gopher:  1440  0%  1440
refresh_pattern .   0  20%  4320

#Recommended minimum configuration:
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl to_localhost dst 127.0.0.0/8
acl SSL_ports port 443 563  # https, snews
acl SSL_ports port 873    # rsync
```

```
acl Safe_ports port 80    # http
acl Safe_ports port 21    # ftp
acl Safe_ports port 443 563  # https, snews
acl Safe_ports port 70    # gopher
acl Safe_ports port 210   # wais
acl Safe_ports port 1025-65535  # unregistered ports
acl Safe_ports port 280   # http-mgmt
acl Safe_ports port 488   # gss-http
acl Safe_ports port 591   # filemaker
acl Safe_ports port 777   # multiling http
acl Safe_ports port 631   # cups
acl Safe_ports port 873   # rsync
acl Safe_ports port 901   # SWAT
acl purge method PURGE
acl CONNECT method CONNECT

http_access allow manager localhost
http_access deny manager
# Only allow purge requests from localhost
http_access allow purge localhost
http_access deny purge
# Deny requests to unknown ports
http_access deny !Safe_ports
# Deny CONNECT to other than SSL ports
http_access deny CONNECT !SSL_ports

#ACL de nosotros
acl miredlocal src 192.168.10.0/24
http_access allow miredlocal
http_access allow localhost

# And finally deny all other access to this proxy
#http_access allow miredlocal
http_access deny all

http_reply_access allow all
icp_access deny all

#Default:
error_directory /usr/share/squid/errors/Spanish
```

ANEXO F

ARCHIVOS DE CONFIGURACIÓN DEL SERVIDOR ASTERISK

/etc/asterisk/sip.conf

```
[general]
context=default
allowoverlap=no
videosupport=yes
port=5060

bindaddr=0.0.0.0
srvlookup=yes

[6001]
type = friend
username = Verito
secret = 1234
context = phones
host = dynamic
canreinvite = no
allow = ulaw

[6002]
type = friend
username = Carmen
secret = 1234
context = phones
host = dynamic
canreinvite = no
allow = ulaw

[6003]
type = friend
username = Mariana
secret = 1234
context = phones
host = dynamic
canreinvite = no
allow = ulaw
```

/etc/asterisk/extensions.conf

```
 [globals]
FEATURES =
DIALOPTIONS =
RINGTIME = 20
FOLLOWMEOPTIONS =
PAGING_HEADER = Intercom

[general]
autofallthrough = yes
static = yes
writeprotect = no
clearglobalvars = yes

[default]
exten => s,1,Verbose(1|Unrouted call handler)
exten => s,n,Answer()
exten => s,n,Wait(1)
exten => s,n,Playback(tt-weasels)
exten => s,n,Hangup()

[incoming_calls]

[internal]
exten => 500,1,Verbose(1|Echo test application)
exten => 500,n,Echo()
exten => 500,n,Hangup()

[phones]
;include => internal
exten => 6000,1,Dial(SIP/6000)
exten => 6000,2,Hangup

exten => 6001,1,Dial(SIP/6001)
exten => 6000,2,Hangup

exten => 6001,1,Dial(SIP/6001)
exten => 6001,2,Hangup

exten => 6002,1,Dial(SIP/6002)
exten => 6002,2,Hangup

exten => 6003,1,Dial(SIP/6003)
exten => 6003,2,Hangup
```

```
 [macro-stdexten]
exten = s,1,Set(__DYNAMIC_FEATURES=${FEATURES})
exten = s,2,Set(ORIG_ARG1=${ARG1})
exten = s,3,GotoIf($["${FOLLOWME_${ARG1}}" = "1"]?6:4)
exten = s,4,Dial(${ARG2},${RINGTIME},${DIALOPTIONS})
exten = s,5,Goto(s-${DIALSTATUS},1)
exten = s,6,Macro(stdexten-followme,${ARG1},${ARG2})
exten = s-NOANSWER,1,Voicemail(${ORIG_ARG1},u)
exten = s-NOANSWER,2,Goto(default,s,1)
exten = s-BUSY,1,Voicemail(${ORIG_ARG1},b)
exten = s-BUSY,2,Goto(default,s,1)
exten = _s-.,1,Goto(s-NOANSWER,1)
exten = a,1,VoicemailMain(${ORIG_ARG1})

[macro-stdexten-followme]
exten = s,1,Answer
exten = s,2,Set(ORIG_ARG1=${ARG1})
exten = s,3,Dial(${ARG2},${RINGTIME},${DIALOPTIONS})
exten = s,4,Set(__FMCIDNUM=${CALLERID(num)})
exten = s,5,Set(__FMCIDNAME=${CALLERID(name)})
exten = s,6,Followme(${ORIG_ARG1},${FOLLOWMEOPTIONS})
exten = s,7,Voicemail(${ORIG_ARG1},u)
exten = s-NOANSWER,1,Voicemail(${ORIG_ARG1},u)
exten = s-BUSY,1,Voicemail(${ORIG_ARG1},b)
exten = s-BUSY,2,Goto(default,s,1)
exten = _s-.,1,Goto(s-NOANSWER,1)
exten = a,1,VoicemailMain(${ORIG_ARG1})
```

ANEXO G

Archivo de configuración para el servicio chargen

/etc/inetd.conf

```
# /etc/inetd.conf:  see inetd(8) for further informations.
#
# Internet superserver configuration database
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
#:INTERNAL: Internal services
#discard    stream  tcp  nowait  root  internal
#discard    dgram udp  wait  root  internal
#daytime    stream  tcp  nowait  root  internal
#time   stream  tcp  nowait  root  internal

#:STANDARD: These are standard services.
ftp    stream  tcp  nowait  root  /usr/sbin/tcpd
  /usr/sbin/in.ftpd

#:BSD: Shell, login, exec and talk are BSD protocols.
chargen    stream  tcp  nowait  root  internal
chargen    dgram   udp  wait  root  internal

#:MAIL: Mail, news and uucp services.

#:INFO: Info services
ident   stream  tcp  wait  identd  /usr/sbin/identd  identd

#:BOOT: TFTP service is provided primarily for booting.  Most sites
#       run this only on machines acting as "boot servers."
tftp    dgram udp  wait  nobody /usr/sbin/tcpd /usr/sbin/in.tftpd --
tftpd-timeout 300 --retry-timeout 5     --mcast-port 1758 --mcast-addr
239.239.239.0-255 --mcast-ttl 1 --maxthread 100 --verbose=5  /tftpboo
```

ANEXO H

Script de inicio para cargar proxy transparente, disciplinas de cola y marcado

/etc/init.d/colas.sh

```
#!/bin/bash

#########################################################################
##
#Script para cargar reglas de inicio del marcado de paquetes, disciplinas de cola y
proxy transparente
#
#########################################################################
##

case $1 in
  start)

iwconfig ath0 rate 11M

#####inicio de las reglas para el PROXY TRANSPARENTE
iptables -t nat -X
iptables -t nat -F
iptables -t nat -A PREROUTING -i ath0 -s 192.168.10.0/24 -p
tcp --dport 80 -j REDIRECT --to-port 3128
iptables -t nat -A PREROUTING -i eth0 -s 192.168.10.0/24 -j
ACCEPT
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -o eth0 -j
MASQUERADE
   echo 1 > /proc/sys/net/ipv4/ip_forward

#####inicio de las reglas del MARCADO DE PAQUETES,
distinguiendo audio, vídeo y best effort

#tráfico tcp en general
iptables -t mangle -A POSTROUTING -p tcp -j TOS --set-tos
Maximize-Throughput

#tráfico ftp
iptables -t mangle -A POSTROUTING -p tcp --sport 20:21 --
dport 20:21 -j TOS --set-tos Maximize-Throughput
```

```
   #tráfico icmp
iptables -t mangle -A POSTROUTING -p icmp -j TOS --set-tos
Maximize-Throughput

#tráfico de audio de asterisk
iptables -t mangle -A POSTROUTING -m length --length 50:300 -
p udp -j TOS --set-tos Minimize-Delay

#tráfico de vídeo de asterisk
iptables -t mangle -A POSTROUTING -m length --length 301:1500
-p udp -j TOS --set-tos Maximize-Reliability


   #####inicio de las DISCIPLINAS DE COLA HTB
#agregando la disciplina de cola raíz
tc qdisc add dev ath0 root handle 1:0 htb default 13

#agregando las clases
tc class add dev ath0 parent 1:0 classid 1:1 htb rate 256kbps
tc class add dev ath0 parent 1:1 classid 1:11 htb rate
160kbps ceil 256kbps prio 0
tc class add dev ath0 parent 1:1 classid 1:12 htb rate 80kbps
ceil 256kbps prio 1
tc class add dev ath0 parent 1:1 classid 1:13 htb rate 16kbps
ceil 256kbps prio 2
tc qdisc add dev ath0 parent 1:11 handle 11: sfq perturb 10
tc qdisc add dev ath0 parent 1:12 handle 12: sfq perturb 10
tc qdisc add dev ath0 parent 1:13 handle 13: sfq perturb 10


#agregando los filtros
tc filter add dev ath0 parent 1: prio 1 protocol ip u32 match
ip tos 0x10 0xff flowid 1:11
tc filter add dev ath0 parent 1: prio 2 protocol ip u32 match
ip tos 0x04 0xff flowid 1:12
tc filter add dev ath0 parent 1: prio 3 protocol ip u32 match
ip tos 0x08 0xff flowid 1:13

   ;;
  stop)
   #####Borrar disciplinas de cola y marcado
   iptables -t mangle -X
   iptables -t mangle -F
   tc qdisc del dev ath0 root
esac
```

BIBLIOGRAFÍA

1. ARIAS, S.J. Implementación Segura de una red Inalámbrica con Equipo de Recursos Limitados y Herramientas de Software Libre. 2da. ed. Xalapa. México: Depp Multimedia, 2007. 230 p.

2. ENGST, A. y Glenn, F. Introducción a las Redes Inalámbricas. Madrid: Anaya Multimedia, 2003. 180 p.

3. HERBERT, T.F. The Linux TCP/IP Stack: networking qos systems. Hingham: Charles River Media, 2004. 220 p.

4. RAPAAPORT, T.S. Wireless Communications: principles and practice. New York: Prentice Hall, 2002. 200 p.

5. ROBBINS, A. Programación en Linux: casos prácticos. 2da. ed. Madrid: Anaya Multimedia, 2005. pp. 120-145.

Recursos Web

- DEBIAN GNU/LINUX

6. http://hpl.hp.com/personal/Jean_Tourrilhes/
   Linux.Wireless.Extensions.html
   2000-07-10.

7. http://linux.com/base/ldp/howto/Adv-Routing-HOWTO/
   index.html
   2010-04-02

8. http://www.mailxmail.com/curso/informatica/redesbasicas/capitulo4.htm,
   2009-06-13.

9. http://www.debian.org/
   2009-04-18.

- MODELOS PARA CALIDAD DE SERVICIO

10. http://es.wikipedia.org/wiki/Calidad_de_Servicio
    2009-10-30.

11. http://comet.columbia.edu/swan/index.html
2009-03-22.

12. http://www.knowplace.org/pages/howtos/traffic_shaping_with_linux
2009-05-24.

13. http://www.tldp.org/HOWTO/Traffic-Control-HOWTO/
2010-01-05.

14. http://www.informatik.unitrier.de/~ley/db/conf/iastedCSN
/iastedCSN2006.html
2009-06-24

15. http://commons.wikimedia.org/wiki/
Image:Diagrama_linux_netfilter_iptables.gif
2009-11-08.

16. http://www.opalsoft.net/qos/DS.htm,
2010-01-15

17. http://luxik.cdi.cz/%7Edevik/qos/htb/manual/userg.htm#burst,
2010-01-22.

18. http://www.rhyshaden.com/ipdgram.htm,

2009-11-28.

- REDES INALAMBRICAS DE AREA LOCAL

19. http://www.monografias.com/trabajos35/redes-inalambricas/redes inalambricas.shtml.

2009-06-14.

20. http://nmg.upc.es/intranet/qos/9/9.3/9.3.19.pdf

2009-07-04.

21. http://www.unincca.edu.co/boletin/indice.htm

2009-08-12