CONTROL DE TRÁFICO MEDIANTE GESTIÓN DINÁMICA DE ANCHO DE BANDA EN 802.11

LUIS EDUARDO GUERRERO MELO
MANUEL JOSÉ HURTADO POLANCO

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
POPAYÁN
2007

CONTROL DE TRÁFICO MEDIANTE GESTIÓN DINÁMICA DE ANCHO DE BANDA EN 802.11

LUIS EDUARDO GUERRERO MELO
MANUEL JOSÉ HURTADO POLANCO

Trabajo de Grado para optar al título de
Ingeniero en Electrónica y Telecomunicaciones

DIRECTOR*: Ing. Guefry Agredo Méndez*

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
Departamento de Telecomunicaciones
Grupo I+D Nuevas Tecnologías en Telecomunicaciones – GNTT
Línea de investigación: Gestión Integrada de Redes y Servicios y Arquitecturas de Telecomunicaciones
Popayán
2007

A quienes con tanto amor han sido mi apoyo constante e incansable, mi ejemplo, mi inspiración, mis ganas de salir adelante… de triunfar. Quienes me han dado los consejos más sabios, me han dado las mejores clases y me han enseñado las mejores jugadas.
Quienes me enseñaron a nunca rendirme, quienes me han llenado la vida de ilusiones y de sueños, de felicidad y que ahora extraño más que nunca, mis papás Manuel y Margarita.

A mis mejores amigos, que extraño día y noche. Mi razón de ser cada día una persona mejor, para darles el mejor ejemplo. A quienes nunca me cansaré de ayudar y proteger, mis hermanos Tata y Alejo.

A quien ha sido mi punto de equilibrio, mi apoyo determinante en esta nueva etapa de mi vida. Quien fue el mejor testigo de mi trabajo en este proyecto y con quien afortunadamente, hoy que está terminado, puedo compartir, mi novia Valen.

A mi familia…. A J

MANOLO

Ha pasado mucho tiempo desde que perdí la oportunidad de abrazarla y decirle que la quiero, sin embargo, en mi dejó la huella de recuerdos inolvidables que el transcurrir del tiempo no ha borrado. Por ser la persona que siempre estuvo a mi lado y se preocupó aunque no debía.
Por que aun tengo presente cuando me decía que tenía que sacar un "cinco sobresaliente".
Por muchas cosas más, y por que no pude decirle gracias…
Quiero dedicar este trabajo y mi carrera
a la gran mujer que para mi fue mi segunda madre.
A la memoria de mi abuela Elina.

LUIS EDUARDO

AGRADECIMIENTOS

Agradecemos a todas las personas que con su colaboración hicieron posible, la elaboración de este documento.

A Dios, por que sin él nada seria realidad, por cuidar de nosotros y nuestras familias.

A nuestros padres por brindarnos todo el apoyo necesario para este proyecto, todas las facilidades, la oportunidad de educarnos y salir adelante.

A nuestras familias que han estado a nuestro lado durante todo este proceso, nos han apoyado y han creído en nosotros.

A Guefry Agredo Méndez, nuestro director, por darnos la oportunidad de trabajar con él, por poner todo su esfuerzo para que el trabajo saliera adelante de la mejor manera, por sus consejos y su invaluable ayuda.

A todos los profesores de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca, por su participación en nuestra formación como profesionales.

A todas aquellas personas que han contribuido en nuestra formación como personas y como ingenieros.

A todos nuestros amigos y compañeros, que creyeron en nosotros y siempre han estado ahí.

A todos y cada uno,

De corazón,
Gracias.

Luis Eduardo y Manuel José

CONTENIDO

RESUMEN xii
INTRODUCCIÓN 13
1. PRINCIPOS DE TEORÍA DE COLAS, PLANIFICACIÓN Y CONFORMACIÓN DE TRÁFICO 15
1.1 EL CONCEPTO DE TEORÍA DE COLAS 15
1.1.1 Situaciones reales de encolamiento 15
1.1.2 Estación básica de servicio: clientes y servidores 17
1.1.3 Proceso de llegada 17
1.1.4 Proceso de servicio 18
1.1.5 Modelos clásicos de encolamiento 18
1.1.6 Colas con prioridades 19
1.2 PLANIFICACIÓN DE PAQUETES 21
1.2.1 First In First Out (FIFO) 22
1.2.2 Encolamiento de Prioridad (*Priority Queuing*, PQ) 23
1.2.3 Encolamiento Equitativo (*Fair Queuing*, FQ) 24
1.2.4 Encolamiento Equitativo por Pesos (*Weighted Fair Queuing*, WFQ)26
1.2.5 Encolamiento Equitativo por Pesos Basado en Clases (*Weighted Fair Queuing*, CB WFQ) 26
1.2.6 Deficit Round Robin (DRR) 27
1.2.7 División jerárquica de enlaces 28
1.2.7.1 Encolamiento basado en clases (CBQ) 28
1.2.7.1.1 Encolamiento Basado en Clases Mejorado con Clasificador de paquetes dependiente del estado del canal 29
1.2.7.2 Curva Jerárquica de Servicio Justo (*Hierarchical Fair Service Curve*, H-FSC) 30
1.2.8 Planificación de paquetes en redes inalámbricas 30
1.3 CONFORMACIÓN DE TRÁFICO 31
1.3.1 Implementación *Leaky Bucket* 32
1.3.2 Implementación *Token Bucket* 33
1.3.3 Múltiples *Token Bucket* 34
RESUMEN 36
2. CALIDAD DE SERVICIO 37
2.1 INTRODUCCIÓN A LA CALIDAD DE SERVICIO EN REDES IP 38
2.1.1 Reserva de recursos 39
2.1.1.1 Protocolo para Reserva de Recursos (RSVP) 39
2.1.2 Control de admisión 39
2.1.2.1 “Negociadores” de Ancho de Banda 39
2.1.3 Marcado de paquetes 40
2.1.4 Clasificación de paquetes 41
2.1.5 Policing 42
2.1.6 Gestión Activa de Colas 42
2.2 ARQUITECTURAS DE QoS EN REDES IP 44

2.2.1 Marcador de tres colores de tasa única (*Single Rate Three Color Marke*r, srTCM)....48
2.2.2 Marcador de tres colores de dos tasas (*Two Rate Three Color Marker*, trTCM)....49
2.3 CALIDAD DE SERVICIO EN REDES INÁLAMBRICAS 50
2.3.1 Red de Área Local Inalámbrica (WLAN) 802.11....50
2.3.2 Calidad de Servicio en 802.11....51
2.3.3 Impacto de la movilidad en la calidad de servicio....53
RESUMEN 54
3. CONTROL DE TRÁFICO EN LINUX Y DESCRIPCIÓN DEL TRÁFICO IP 56
3.1 TRÁFICO DE TIEMPO REAL 56
3.1.1 Tráfico Interactivo (intolerante) 57
3.1.2 Tráfico no Interactivo (tolerante)....57
3.1.3 Soporte de Tráfico de Tiempo Real 57
3.2 TRÁFICO DE NO TIEMPO REAL....59
3.3 MÉTRICAS DE CALIDAD DE SERVICIO....60
3.3.1 Qos en Servicios de Tiempo Real 60
3.3.1.1 Ruido de Cuantificación....60
3.3.1.2 Retardos....61
o Retardo de Trama....61
o Retardo de Intercalamiento....61
o Probabilidad de Pérdida de Paquete 61
3.3.2 Jitter o Variación de Retardo 62
3.4 DESCRIPCIÓN DE LOS DIFERENTES TIPOS DE TRÁFICO....63
3.4.1 Tráfico agregado de Internet 63
3.4.2 Componentes del tráfico de Internet....65
3.5 CLASIFICACIÓN DEL TRÁFICO, CON BASE EN LAS EXIGENCIAS DE CALIDAD DE SERVICIO....66
3.6 CONTROL DE TRÁFICO EN LINUX 69
3.6.1 NetFilter....71
3.6.2 IPtables....71
3.6.3 Tabla *Mangle*....72
3.6.4 Targets 72
3.6.5 Extensiones de comparación....74
3.6.6 Componentes del Control de Tráfico en Linux....74
3.6.6.1 Disciplinas de Colas 74
3.6.6.2 Clases....76
3.6.7 IPROUTE2....77
3.6.8 Administración y configuración de Linux TC....78
3.6.9 *Hierarchical Token Bucket* (HTB) 78
3.6.10 *Intermediate Queueing Device* (IMQ) 79
3.6.11 Control de Tráfico de Nueva Generación (TCNG)....80
RESUMEN 81
4. IMPLEMENTACIÓN Y PRUEBAS DEL PILOTO DE GESTIÓN DINÁMICA DE ANCHO DE BANDA....82

4.1 CONTROL DE TRÁFICO DE LINUX EN REDES INALÁMBRICAS 82
4.2 IMPLEMENTACIÓN DEL PLANIFICADOR DE TRÁFICO HTB 83
4.2.1 El *script* de funcionamiento.....84
4.2.2 Implementación en puntos de acceso inalámbricos 90
4.2.2.1 OpenWRT.....91
4.2.2.2 Enrutador de Banda Ancha Linksys WRT54GL.....91
4.2.2.3 Instalación del Planificador de Tráfico 92
4.2.3 Configuración del Enlace Inalámbrico 94
4.3 PRUEBAS Y RESULTADOS 100
RESUMEN 110
CONCLUSIONES Y RECOMENDACIONES 111
REFERENCIAS BIBLIOGRÁFICAS CAPÍTULO 1 114
REFERENCIAS BIBLIOGRÁFICAS CAPÍTULO 2 116
REFERENCIAS BIBLIOGRÁFICAS CAPÍTULO 3 117
REFERENCIAS BIBLIOGRÁFICAS CAPÍTULO 4 119
ANEXO 1: *SCRIPT* DE FUNCIONAMIENTO 120

LISTA DE FIGURAS

Figura 1.1 Ejemplos de conmutadores de red 16
Figura 1.2 Estación básica de servicio 17
Figura 1.3 Paquetes llegando a un conmutador de paquetes 17
Figura 1.4 Mecanismo de encolamiento FIFO 22
Figura 1.5 Mecanismo de encolamiento PQ 23
Figura 1.6 Mecanismo de encolamiento FQ 25
Figura 1.7 Mecanismo de encolamiento WRR 25
Figura 1.8 Mecanismo de encolamiento CB WFQ 27
Figura 1.9 Errores en el canal inalámbrico dependientes de la ubicación 30
Figura 1.10 Implementación del control de tráfico en los extremos LAN-WLAN, WLAN-LAN 32
Figura 1.11 Implementación *Leaky-Bucket* 33
Figura 1.12 Implementación *Token Bucket* 34
Figura 1.13 Múltiples *Token Bucket* 35
Figura 2.1 Procesamiento típico en enrutadores sin conformación de tráfico ni marcado 44
Figura 2.2 Procesamiento típico en enrutadores 44
Figura 2.3 Procesamiento típico en enrutadores con conformación de tráfico 45
Figura 2.4 Componentes básicos de la arquitectura IntServ/RSVP 46
Figura 2.5 Arquitectura DiffServ 47
Figura 2.6 srTCM 48
Figura 2.7 802.11 entre otros miembros de la familia IEEE 802 50
Figura 2.8 Modo adhoc e infraestructura en 802.11. 51
Figura 3.1 Estrategias para alcanzar QoS 58
Figura 3.2 Análisis del tráfico de Internet basado en protocolos [1] 65
Figura 3.3 Análisis del tráfico TCP [1] 66
Figura 3.4 Interconexión en Linux y el Modelo de referencia OSI 70
Figura 3.5 Niveles donde se ve involucrado el control de tráfico 70
Figura 3.6 Interacción entre las herramientas de control de tráfico Linux 80
Figura 4.1 Representación esquemática de la arquitectura de funcionamiento de HTBi .. 83
Figura 4.2 Diagrama de flujo del funcionamiento del planificador en las colas de entrada. 84
Figura 4.3 Escenario de funcionamiento HTBi 90
Figura 4.4 Enrutador de banda ancha Linksys WRT54GL 92
Figura 4.5 Topología del enlace punto a punto 95
Figura 4.6 Configuración del enlace punto a punto 97
Figura 4.7 Topología de enlace punto - multipunto 98
Figura 4.8 Montaje de prueba. 100
Figura 4.9 Comportamiento del tráfico sin ningún mecanismo de control de tráfico. 101
Figura 4.10 Efectos del planificador de tráfico sin asignación de prioridades. 102
Figura 4.11 Funcionamiento de HTBi. 104
Figura 4.12 Tratamiento para el tráfico masivo 105
Figura 4.13 Identificación y control de tráfico frente a una llamada VoIP 106
Figura 4.14 Distribución del enlace en los porcentajes asignados 107
Figura 4.15 Comportamiento del tráfico con burstiness configurado a 300 ms 108
Figura 4.16 Comportamiento del tráfico con burstiness configurado a 1000 ms . 109

Figura 4.17 Comportamiento del tráfico con burstiness configurado a 2000 ms .109

LISTA DE TABLAS

Tabla 2.1 Semántica para Tipo de Servicio en IPv4 40
Tabla 2.2 Semántica para Precedencia IP IPv4 41
Tabla 2.3 Configuración del Campo DSCP 41
Tabla 2.4 Mapeado de prioridad a categoría de acceso 53
Tabla 3.1 Características del tráfico de aplicaciones en Internet 63
Tabla 3.2 Clasificación de Servicios basada en tipo, requerimientos de tiempo real y demanda de Calidad de servicio 67
Tabla 3.3 Reglas por defecto en NetFilter 71
Tabla 3.4 Targets de la tabla Mangle 73
Tabla 3.5 Targets de IPtables 73

LISTA DE ACRÓNIMOS

AP	Punto de Acceso (*Access Point)*
AQM	Gestión Activa de Colas (*Active Queue Management)*
CBQ	Encolamiento Basado en Clases (*Class Based Queuing*)
CAC	Control de Admisión de Conexión (*Connection Admission Control*)
CIR	Tasa de Información Comprometida (*Committed Information Rate*)
CSMA/CA	Acceso Múltiple por Detección de Portadora con Evasión de Colisión (*Carrier Sense Multiple Access with Collision Avoidance*)
DCF	Función de Coordinación Distribuida (*Distributed Coordination Function*)
DCS	Sistema de Comunicaciones Digital (*Digital Communications System*)
DSCP	Punto de Código de Servicios Diferenciados (*DiffServ Code Point*)
DSN	Servidor de Nombres de Dominio (*Domain Server Name*)
EDCA	Acceso al Canal Distribuido Mejorado (*Enhanced Distributed Channel Access)*
EDCF	Función de Coordinación Distribuida Mejorada *(Enhanced Distribution Coordination Function)*
ETSI	Instituto Europeo de Estándares de Telecomunicaciones (European Telecommunications Standards Institute)
FIFO	Primero en llegar Primero en Salir (*First In First Out)*
FQ	Encolamiento Equitativo (*Fair Queuing*)
FTP	Protocolo de Transferencia de Archivos (*File Transfer Protocol*)
GPL	Licencia General Pública *(General Public License*)
HCCA	Acceso al Canal Controlado (*Controlled Channel Access)*
HCF	Función de Coordinación Híbrida (*Hybrid Coordination Function)*
HFSC	Algoritmo Jerárquico de Curva de Servicio Justo (*Hierarchical Fair Service Curve Algorithm*)
HoQ	Cabeza de la cola (*Head of Queue*)
HTB	*Token Bucket* Jerárquico (*Hierarchical Token Bucket*)
IEEE	Instituto de Ingenieros Eléctricos y Electrónicos (*Institute of Electrical and Electronics Engineers*)

IP	Protocolo de Internet *(Internet Protocol*)
ITU	Unión Internacional de Telecomunicaciones (*International Telecommunications Union)*
LAN	Red de Área Local (*Local Area Network*)
MAC	Control de Acceso al Medio (*Medium Access Control*)
MF	Clasificacion Multi-Campo (*Multi-Field*)
NAT	Traducción de Direcciones de Red (*Network Address Translation*)
OFDM	Multiplexación por División de Frecuencia Ortogonal (*Orthogonal Frequency Division Multiplexing*)
OSI	Interconexión de Sistemas Abiertos (*Open Systems Interconnection*)
P2P	Servicio *Peer to Peer*
PIR	Tasa Pico de Información (*Peak Information Rate)*
PCF	Función de Punto de Coordinación (*Point Coordination Function)*
PDA	Asistente Digital Personal (*Personal Digital Assistant)*
PQ	Encolamiento de Prioridad (*Priority Queuing)*
QoS	Calidad de Servicio (*Quality of Service*)
RED	Descarte Prematuro Aleatorio (*Random Early Discarding*)
RTP	Protocolo de Tiempo Real (*Real-Time Protocol*)
SIP	Protocolo de Inicio de Sesión (*Session Initiation Protocol*)
STA	Estación Móvil de Destino (*Destination Mobile Station*)
TCP	Protocolo de Control de Transmisión (*Transmission Control Protocol*)
ToS	Campo Tipo de Servicio (*Type of Service*)
UDP	Protocolo de Datagramas de Usuario (*User Datagram Protocol*)
VoATM	Voz sobre ATM (*Voice over ATM*)
VoIP	Voz sobre el Protocolo de Internet (*Voice over Internet Protocol*)
WAN	Red de Área Amplia (*Wide Area Network)*
WFQ	Encolamiento Equitativo por Pesos (*Weighted Fair Queuing)*
Wi-Fi	*Wireless Fidelity*

WiMAX Interoperabilidad Mundial para Acceso por Microondas (*Worldwide Interoperability for Microwave Access*)

WISP Proveedor de Servicios de Internet Inalámbrico (*Wireless Internet Service Provider)*

WLAN Red de Área Local Inalámbrica (*Wireless Local Area Network*)

WPS Planeación de Paquetes Inalámbricos (*Wireless Packet Scheduling*)

WWW World Wide Web

RESUMEN

Para los dispositivos que trabajan con la tecnología 802.11 se tiene la necesidad de compartir el ancho de banda inalámbrico de una manera controlada y eficiente, asignando adecuadamente los recursos de una red para responder a los requerimientos de cada tipo de servicio, los cuales no tienen un funcionamiento apropiado dado que reciben un tratamiento *Best Effort*. Lo que representa un problema al ocasionar una percepción de mala calidad de los servicios y aplicaciones, que se soportan sobre la red en el usuario final.

Mediante Gestión de Ancho de Banda se puede diferenciar el tráfico entrante para dar mayor prioridad a aplicaciones que lo requieren (Ej: paquetes de video o voz), los dispositivos de 802.11 actuales, se limitan a cumplir con los requerimientos de funcionamiento de las dos capas más bajas de la arquitectura OSI (capas física y de enlace de datos), definidos en el estándar, y no implementan ninguna de estas técnicas, imposibilitando brindar un tratamiento específico según la aplicación y olvidando por completo los requerimientos especiales que se deben tener para determinados tipos de tráfico, si se quiere controlar adecuadamente el mismo.

Este proyecto involucra el desarrollo de un piloto, el cual resulta económico, dado que no requiere hacer cambios significativos en la red; el único cambio necesario, es la adecuación del *firware* de los Puntos de Acceso inalámbricos en los extremos de transición LAN – WLAN y WLAN – LAN.

INTRODUCCIÓN

Los accesos inalámbricos y las infraestructuras de red sin cables se posicionan como una alternativa llamativa en el competido mercado de los servicios de datos, esto no implica que las tecnologías cableadas sean obsoletas, ni desmerita su eficiencia, por el contrario, las tecnologías cableadas son soportadas por muchos años de investigación, proveedores de dispositivos con reconocimiento global, una basta documentación y se encuentran en evolución constante, lo que obliga a los promotores de las tecnologías inalámbricas para comunicación de datos a implementar proyectos de I+D[1] cuyos resultados ubiquen estas tecnologías, como mínimo, al nivel de las redes cableadas y a la altura de los servicios actuales de información.

El concepto actual de las redes, converge hacia una única infraestructura de red que se espera tenga la capacidad de transportar voz en forma de datos, video y aplicaciones críticas de tiempo real. Este tipo de aplicaciones tiene ciertos requisitos para brindar al usuario final un producto de calidad, es en este momento cuando la calidad de servicio QoS (*Quality of Service*, QoS) juega un papel importante.

El grupo de trabajo IEEE 802.11 asignó la responsabilidad al grupo 802.11e de adecuar la capa de control de acceso al medio (*Medium Access Control, MAC*) de 802.11 con el fin de incluir QoS bidireccional para soportar aplicaciones sensibles ante la latencia como lo son la voz y el video. Con el estándar 802.11e, la tecnología IEEE 802.11 soporta tráfico de este tipo de aplicaciones cumpliendo con requisitos de tiempo real. La opción de instalar equipos 802.11e para soportar calidad de servicio en una red inalámbrica puede ser viable cuando se plantea desde un principio, pero resulta poco conveniente para quienes ya tienen redes inalámbricas (no 802.11e) funcionando y necesitan darle más robustez a la red, controlando el flujo de la información; esta opción representa un gasto que puede ser infructuoso teniendo en cuenta que los equipos 802.11, a, b o g no son obsoletos.

La implementación del control tráfico mediante gestión dinámica de ancho de banda, en los Puntos de Acceso de los extremos LAN – WLAN y WLAN – LAN de este proyecto, busca planificar el tráfico de los enlaces principales para hacer un mejor aprovechamiento de los recursos y obtener un mejor desempeño mediante la implementación de mecanismos de calidad de servicio como son la conformación de tráfico, filtrado y estableciendo diferentes políticas de funcionamiento de acuerdo a los recursos de la red y a las necesidades de los usuarios, tarea que hasta el momento no se hace dadas las limitaciones de la tecnología.

Este documento contiene cuatro capítulos que recopilan el proceso de investigación, desarrollo e innovación. El primer capítulo **PRINCIPIOS DE TEORÍA DE COLAS, PLANIFICACIÓN Y CONFORMACIÓN DE TRÁFICO,** está enfocado en estos mecanismos de calidad de servicio y su papel determinante en la asignación de prioridades para los paquetes y la diferenciación de tráfico y servicios en redes IP.

El capítulo 2 **CALIDAD DE SERVICIO,** contiene las bases para obtener niveles de calidad de servicio en redes IP en diferentes escenarios, dentro de los que se encuentra el inalámbrico y los avances que en ésta área se tienen hasta el momento.

[1] Término utilizado para referirse a proyectos de Investigación y Desarrollo

Las consideraciones específicas para cada tipo de tráfico en particular junto a los principales inconvenientes que encuentra en su paso por la red, son estudiados en el capítulo 3 **CONTROL DE TRÁFICO EN LINUX Y DESCRIPCIÓN DEL TRÁFICO IP**, así como la introducción al control de tráfico en Linux, principal herramienta para el desarrollo de este proyecto.

El proceso de implementación del proyecto, se explica y justifica en el capítulo 4 **IMPLEMENTACIÓN Y PRUEBAS DEL PILOTO DE GESTIÓN DINÁMICA DE ANCHO DE BANDA**, que también incluye las pruebas realizadas y sus escenarios.

1. PRINCIPOS DE TEORÍA DE COLAS, PLANIFICACIÓN Y CONFORMACIÓN DE TRÁFICO

En este capítulo, inicialmente, se presentan las principales temáticas de la teoría de colas (o la teoría de encolamiento en redes), que es aplicada al campo de las telecomunicaciones para resolver problemas de tráfico, originando la disciplina del teletráfico, la cual tiene como propósito determinar reglas, para optimizar la utilización de equipos de red que manejan el tráfico solicitado por los usuarios [1].

La función básica de una red de telecomunicaciones es interconectar usuarios a través de terminales, y conectar usuarios a servidores, o terminales a otros terminales. La red configura una conexión entre dos o más terminales haciendo uso de las direcciones fuente y destino. Detrás de este concepto general existe un número de realidades e inconvenientes muy diferentes. La teoría de colas principalmente se aplica para los fenómenos de espera y congestión que están relacionados con la naturaleza aleatoria e impredecible de algunos eventos, en el caso de las telecomunicaciones eventos como: pedidos de configuración de conexión, paquetes IP a través de un enrutador, tareas para ser procesadas en un servidor, entre otros.

En las redes actuales de conmutación de paquetes, el ancho de banda del enlace se comparte por múltiples fuentes de tráfico, implementando básicamente dos mecanismos: el encolamiento y la planificación. Los enlaces transmiten un paquete al tiempo desde los *buffers*[2], como se tienen múltiples paquetes en el *buffe*r, los algoritmos de planificación, presentados en éste capítulo, definen el plan de transmisión de los paquetes en el enlace. Varios parámetros de calidad de servicio de un determinado flujo de tráfico, pueden afectarse significativamente por la elección del mecanismo de encolamiento y de planificación.

La conformación de tráfico, presentada en este capítulo, permite controlar la “forma” del tráfico. A pesar que puede parecer que éste mecanismo y el *policing*, explicado posteriormente en el capítulo 2, realicen la misma función, existen diferencias conceptuales y técnicas. Conformar el tráfico puede verse como una forma de suavizar ráfagas de tráfico y no como una forma de descartar el que no es conforme, incluso puede almacenarlo en *buffers*, mientras que el *policing* no tiene ninguna función como suavizante y se preocupa por detectar violaciones.

1.1 EL CONCEPTO DE TEORÍA DE COLAS

1.1.1 Situaciones reales de encolamiento

Muchas situaciones que se experimentan en la vida diaria, utilizan el concepto de encolamiento. Por ejemplo, un cliente en un restaurante hace la fila (cola) para ordenar, espera en la cola hasta que llega su turno, ordena, paga y sale del lugar; el cliente es atendido por una cajera, pero además de la cajera hay todo un grupo de gente trabajando para preparar su pedido, la velocidad con la que su pedido es preparado puede variar según el número de trabajadores en el restaurante. Si el servicio es muy lento, el restaurante puede contratar más trabajadores y ser más

[2]Es una ubicación de memoria en un computador o en un dispositivo digital reservada para el almacenamiento temporal de información digital, mientras que está esperando ser procesada.

eficiente, pero si en cambio el mantenimiento del restaurante es muy costoso, se debe de recortar el número de trabajadores y el servicio se puede deteriorar.

Otro ejemplo es el utilizado en los bancos donde existe una fila de clientes y varios cajeros, cada vez que un cajero está disponible el cliente en la cabeza de la cola (*Head of Queue, HoQ*) es atendido, recibe su servicio y abandona el banco. Estos dos ejemplos son similares y se pueden modelar con una cola de fila y servidor simple [2].

Cuando un carro llega a un peaje con múltiples casetas de pago, el conductor escoge al azar la caseta donde desea pagar, espera en la fila, paga el peaje y sigue; si se asume que en todas las casetas se paga el mismo precio, que todas tienen las mismas características y que el conductor escoge la caseta aleatoriamente, se puede modelar este caso como una cola de fila y servidor simple también.

La figura 1.1 muestra dos ejemplos de conmutadores de red; llamadas telefónicas atendidas por troncales en un conmutador de circuitos y peticiones para conexiones virtuales atendidas por un conmutador (*switch*) ATM. En el caso del conmutador de circuitos, cuando un usuario intenta realizar una llamada que va fuera del servidor central donde esta conectada la línea, el conmutador primero trata de encontrar un circuito troncal que no esté siendo utilizado, si encuentra un circuito disponible la llamada se coloca en este circuito, si no hay ninguna troncal disponible el usuario recibe un tono de indisponibilidad del servicio. En el conmutador ATM llegan una petición de conexión virtual, el algoritmo de Control de Admisión de Conexiones (*Connection Admission Control, CAC*) implementado en el conmutador determina si hay suficiente ancho de banda disponible en el puerto de salida para aceptar o no la petición.

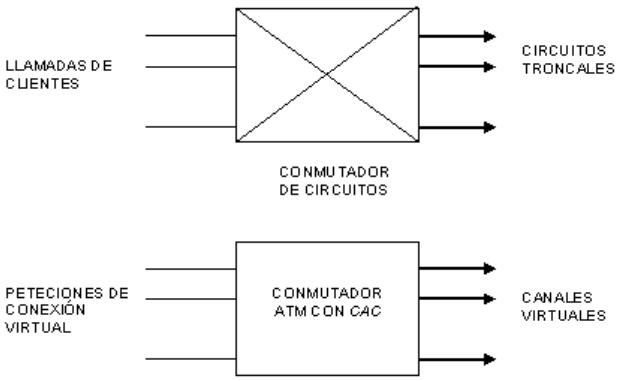

Figura 1.1 Ejemplos de conmutadores de red

En la figura 1.2 se muestra otro ejemplo de tipo de encolamiento; los paquetes de entrada dentro de un enrutador IP son puestos en un *buffer*, el planificador de paquetes[3] determina qué paquetes se envían por el puerto de salida. Existen varios planificadores de paquetes los cuales se utilizan en unos u otros tipos de enrutadores y serán mencionados posteriormente en este capítulo en la sección 1.2.

[3] Se refiere al término *Scheduling* utilizado en las redes de datos

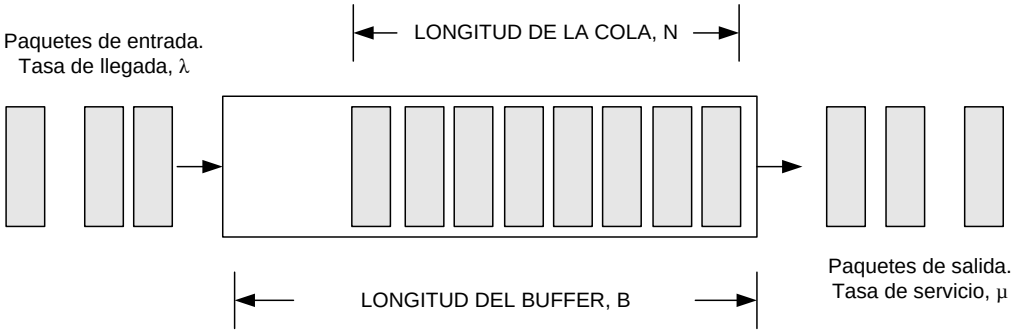

Figura 1.2 Estación básica de servicio

1.1.2 Estación básica de servicio: clientes y servidores

En la teoría de colas, los clientes son entidades que se mueven en una red de servidores, donde reciben procesamiento [1]. Cuando varios clientes tratan de obtener simultáneamente un servicio, algunos de ellos deben esperar en colas o *buffers*. En el caso específico de redes IP de paquetes, pueden representar paquetes que son enviados a una línea de transmisión y esperan su disponibilidad. El servidor es entonces, el enlace de transmisión y el tiempo de servicio es el tiempo de transmisión.

La estación de servicio más general consiste de una cola, de capacidad finita o infinita, vaciada por uno o más servidores y surtida por un flujo de clientes como se observa en la figura 1.3.

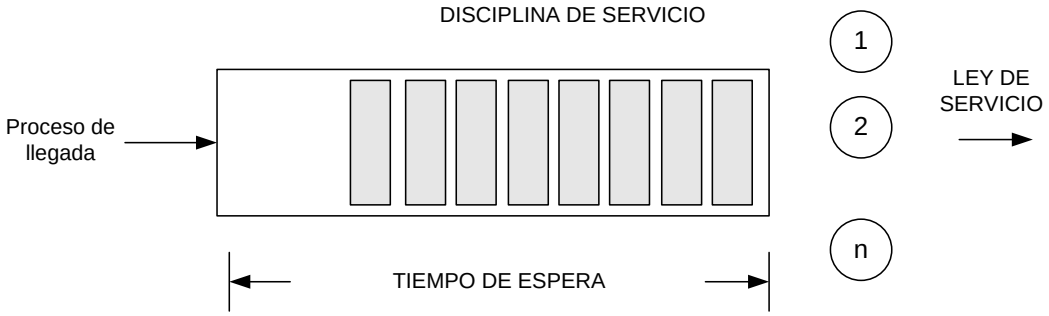

Figura 1.3 Paquetes llegando a un conmutador de paquetes

Para describir el sistema con mayor precisión, debe ser posible especificar:

- El mecanismo acorde al cual los clientes llegan al *buffer*
- El tiempo de servicio
- La disciplina de servicio

1.1.3 Proceso de llegada

Se observa la llegada de clientes a la entrada del sistema. Para describir el fenómeno (ley de llegada) la primera idea es utilizar el intervalo de tiempo entre llegadas exitosas, o el número de llegadas en determinado intervalo. Durante el tiempo T,

ocurren n(T) llegadas. La intensidad del flujo se expresa como un número, la tasa de llegada, cuya definición es [1]:

$$\lambda = \lim_{T \to \infty} \frac{n(T)}{T}$$

Es también posible estimar el intervalo promedio entre llegadas, siendo éste el inverso de la cantidad previa.

En el caso más favorable, la descripción estadística del tiempo entre llegadas es muy simple. La situación puede describirse de la siguiente manera: cada vez que ocurre una llegada, el intervalo hasta la siguiente llegada se modela de acuerdo a una ley dada, de una forma tal que intervalos sucesivos son independientes el uno del otro. Este es el caso especial del proceso de renovación (*renewal process*).

Cuando el proceso de llegada cumple con las siguientes características:

- La probabilidad de llegada en un intervalo $[t,t+\Delta t]$, no depende de lo que paso antes del instante t. Ésta es la propiedad de la probabilidad conocida como *memoryless* (sin memoria).
- La probabilidad de llegada de un cliente es proporcional a Δt y la probabilidad de más de un evento es insignificante.

Se tienen las consideraciones clásicas que llevan a un proceso de distribución de *Poisson.*

1.1.4 Proceso de servicio

El proceso de servicio puede ser extremadamente complejo, pero usualmente se puede limitar a asumir que cada tiempo de servicio es independiente de otros y que cada uno de ellos cumple con la misma función de distribución: se habla de variables independientes e idénticamente distribuidas (*iid*). Esta ley, la ley de servicio, se describe por su distribución de probabilidad:

$$B(x) = P\{\text{tiempo de servicio} \leq x\}$$

Asumir la independencia e idéntica distribución entre servicios sucesivos, así sea una simplificación, de hecho representa una muy buena aproximación en la mayoría de los casos, tales como llamadas o tiempos independientes entre sesiones, entre otros. Pero existe la posibilidad de encontrarse con sistemas manejando clientes cuyas leyes de servicio utilicen diferentes distribuciones. Algunas de las leyes de servicio más comunes son: la Ley Exponencial, las Leyes de Erlang y la Ley Hiperexponencial.

1.1.5 Modelos clásicos de encolamiento

Para identificar un sistema de encolamiento, la siguiente expresión formal se ha propuesto y adoptado unánimemente (notación Kendall):

A/B/n/K/N

La primera letra identifica la distribución del proceso de llegada, la segunda el proceso de servicio; en los dos casos se tienen las siguientes convenciones:

- M: Ley sin memoria (Llegadas de Poisson o servicio exponencial)
- D: Ley deterministica
- Ek: Ley Erlang K
- GI: Ley general, con variables sucesivas independientes
- G: Ley general, sin asumir ninguna independencia

La siguiente representación en la expresión es para identificar el número de servidores, la siguiente (que es opcional) identifica el tamaño de la cola y el tamaño de la fuente (si estos valores se omiten, se suponen como infinitos).

Por ejemplo, M/M/1 se refiere al modelo básico: llegadas tipo *Poisson*, servicio exponencial, un servidor simple, longitud ilimitada de la cola. En una cola, un parámetro de vital importancia es el factor de utilización: ρ. Este es el producto de la tasa de llegada del cliente por el tiempo de servicio: $\rho = \lambda E(s)$. Se puede demostrar que $1-\rho$ es la probabilidad de encontrar el servidor inactivo [1]. Es necesario (para una cola de capacidad infinita) que $\rho<1$.

M/M/1 es uno de los modelos de encolamiento mas conocidos, éste permite la ilustración de los conceptos fundamentales relacionados con el encolamiento en un servidor, de la forma como se encuentran en la mayoría de equipos de red, ya sean tareas de procesamiento de llamadas o paquetes en un enrutador. Cuando $n>0$ clientes están en el sistema (uno en servicio, n-1 encolados) en un instante t, la finalización del servicio del cliente en progreso ocurre en el intervalo $[t, t+\Delta t]$ con probabilidad de $\mu\Delta t+o(\Delta t)$: propiedad fundamental de la ley exponencial de servicio. De forma similar, una llegada ocurrirá en el intervalo con probabilidad $\lambda\Delta t+o(\Delta t)$.

El modelo M/M/R/R (modelo Erlang) es uno de los más populares y el primero en ser estudiado. Se consideran un grupo de R servidores, operados en modo colectivo (por ejemplo, cada uno de los servidores puede atender cualquiera de los clientes que se presentan). Los clientes llegan frente al grupo de servidores de acuerdo a un proceso de *Poisson*, con tasa λ. A su llegada, los clientes son servidos inmediatamente mientras al menos un servidor este libre; si todos los servidores están ocupados, el cliente que llega es rechazado y se asume que desaparece definitivamente. Se habla de un modelo de pérdida.

1.1.6 Colas con prioridades

Si una clase de clientes tiene niveles particulares de urgencia, es posible en la mayoría de los casos, configurar mecanismos para dar prioridad. Por ejemplo, un procesador puede tener la capacidad de procesar rápidamente tareas relacionadas con funciones vitales de seguridad, aunque esto implique retardar tareas menos criticas. Alternativamente, un enrutador dará preferencia a paquetes cuyos requerimientos de calidad de servicio sean más demandantes.

Se consideran dos puntos de vista en el análisis de sistemas con prioridades [1]: el del cliente y el del servidor. En algunos casos la opción del cliente es indiferente para el servidor, en el sentido que ésta decisión no cambia el trabajo total que debe manejar. La consecuencia es que en cada instante, el servidor es capaz de considerar la

cantidad de trabajo que se le ofrece (trabajo no finalizado): el tiempo de servicio aun debe ser brindado al cliente actual, al cual se le agrega la suma de los servicios de clientes esperando en el instante en cuestión. Esta cantidad puede ser evaluada en un instante fijo, pero solo es representativa si el trabajo restante no depende de la siguiente decisión. Se denota *T* el trabajo pendiente (no finalizado) en un instante t; si se para el proceso de llegada en este instante y se observa el comportamiento aleatorio del sistema el servidor trabajara continuamente hasta t+*T*, cualquiera que sea la decisión de planificación tomada después de t.

Un sistema conservador de trabajo, es un sistema de este tipo en el cual el trabajo no finalizado no depende de la disciplina de decisión aplicada por el servidor; aunque esta situación es común, existen contra ejemplos como los sistemas impacientes, los sistemas cuyo tiempo de servicio depende de esperas, entre otros. Para la mayoría de estos sistemas, no es ni siquiera posible calcular éste trabajo en un instante dado.

En el caso de una cola que recibe varios flujos diferentes, se denota λ_j la intensidad de flujo de clientes del tipo j (se habla de clases de clientes) y como W_j el promedio de tiempo de espera que experimentan. Se denota λ el flujo total enviado al servidor. Es posible, por ejemplo experimentalmente, medir el tiempo de espera mientras se ésta inconciente de la existencia de clases de clientes, ésta cantidad se denota $\overline{W}$, que es una suma de porcentajes, donde los porcentajes representan la proporción de cada flujo:

$$\overline{W} = \sum_j \frac{\lambda_j}{\lambda}\left(W_j\right)$$

Si se tiene una cola recibiendo clientes de diferentes clases y servidos de acuerdo a cualquier mecanismo de prioridad que no conserva y que no es preventivo (puede ser o no basado en clases), la ley de conservación es:

$$\sum \rho_i W_i = \frac{\rho}{1-\rho} W_0 = Cste$$

Donde $W_0 = \sum \lambda_i \frac{E(s_i^2)}{2} = \sum_i \rho_i \frac{E(s_i^2)}{2E(s_i)}$.

El índice i corre en las clases de clientes; $\rho_i = \lambda_i W_i$ representa la carga en la clase i (W_i es el tiempo promedio de espera que experimenta); $\sum \rho_i = \rho$ es la carga total ofrecida al servidor. La formula implica que la expresión no depende de la disciplina de servicio [1].

1.2 PLANIFICACIÓN DE PAQUETES

La planificación de paquetes (***packet scheduling***), la clasificación de paquetes y el encolamiento son esenciales para proveer diferenciación de servicios o garantizar calidad de servicio extremo a extremo para servicios multimedia y en aplicaciones en redes inalámbricas de paquetes como el caso de este proyecto.

La forma como se tratan los paquetes de entrada y como se planifica la transmisión a cada cola afecta directamente al *throughput*[4], al retardo, al *jitter*[5] e incluso la tasa de pérdida de paquetes del tráfico correspondiente [3] [4].

La planificación y el encolamiento básicamente se encargan del comportamiento temporal de los paquetes desde el momento que ingresan a la cola (s) de entrada del nodo correspondiente hasta que salen hacia el próximo salto o hacia aplicaciones de niveles superiores [4]. Con estos algoritmos se busca:

- Mejorar el *Throughput*: El algoritmo debe tener la capacidad de garantizar *throughput* de corta duración para sesiones libres de error y *throughput* de larga duración para todas las sesiones.

- Ecuanimidad y eficiencia: El algoritmo no debe castigar estaciones que temporalmente excedan el ancho de banda reservado para el canal, asignándole ancho de banda que no este en uso. Se debe asegurar ecuanimidad entre sesiones.

- Escalabilidad y complejidad: El algoritmo debe de ser de baja complejidad, desde el punto de vista de encolamiento y des-encolamiento[6] de un paquete, factor que es muy importante en el caso de redes inalámbricas que presentan limitaciones de recursos de procesamiento y de energía. Además, el algoritmo debe operar eficientemente a medida que aumenta el número de sesiones compartiendo el canal.

- Consumo de energía: Para el caso de redes inalámbricas y redes móviles, el algoritmo debe considerar la necesidad de prolongar la batería del terminal.

- Robustez: El algoritmo debe ser robusto para tolerar errores de ráfagas en canales inalámbricos e inteligentemente proteger sesiones con buenas condiciones de canal cuando reconecte sesiones que experimenten malas condiciones.

- Retardos/Desacople de ancho de banda: Para muchos algoritmos el retardo está fuertemente acoplado con la tasa reservada, pero es recomendable desacoplarlos porque algunos servicios de gran ancho de banda como el FTP pueden tolerar retardos relativamente largos [4].

Cuando se trabaja con redes inalámbricas es más difícil lograr una perfecta planificación del tráfico, principalmente por que el *troughput* y los tiempos de retardo varían según la calidad de la señal, sin embargo, en conexiones punto a punto o punto

[4] Determinada cantidad de datos que se transmiten en un tiempo dado
[5] Posteriormente en el capítulo 3 se explica este inconveniente de red
[6] Proceso inverso al encolamiento

multipunto la variación de dichos parámetros no es tan crítica ya que no hay movilidad, y las características de la comunicación se mantienen regularmente estables por lo que puede trabajarse asumiendo valores estáticos. El detalle del alcance del proyecto con respecto a estos objetivos se puede ver en la sección 4.2.1

Para el desarrollo de este proyecto se estudiaron varios métodos de planificación de paquetes:

1.2.1 *First In First Out*[7] (FIFO)

Este mecanismo de encolamiento es utilizado por defecto cuando no existen otros algoritmos específicos de planificación de paquete [5]; los paquetes de entrada son encolados en una sola cola en el orden en que llegan y son enviados por el enlace de salida en este mismo orden.

Dado que el primer paquete en llegar es el primero en ser atendido, la cola FIFO se le denomina también como cola FCFS (*First-Come-First-Served* o primero en llegar primero en ser atendido). Esta cola tiene como gran ventaja su facilidad de implementación y funcionamiento, no utiliza ningún algoritmo especial, solo requiere de un solo *buffer*, el cual puede almacenar los paquetes entrantes a medida que estos llegan y sacarlos en el mismo orden.

En FIFO todos los paquetes se tratan de la misma manera [5][6], entonces, es un mecanismo apto para redes *best effort*; el principal inconveniente que presenta, es no distinguir (o tener una mínima capacidad de distinguir) entre clases de tráfico, por lo que todos los flujos de tráfico se afectan de igual forma durante la congestión. El esquema de funcionamiento de este mecanismo se muestra en la figura 1.4.

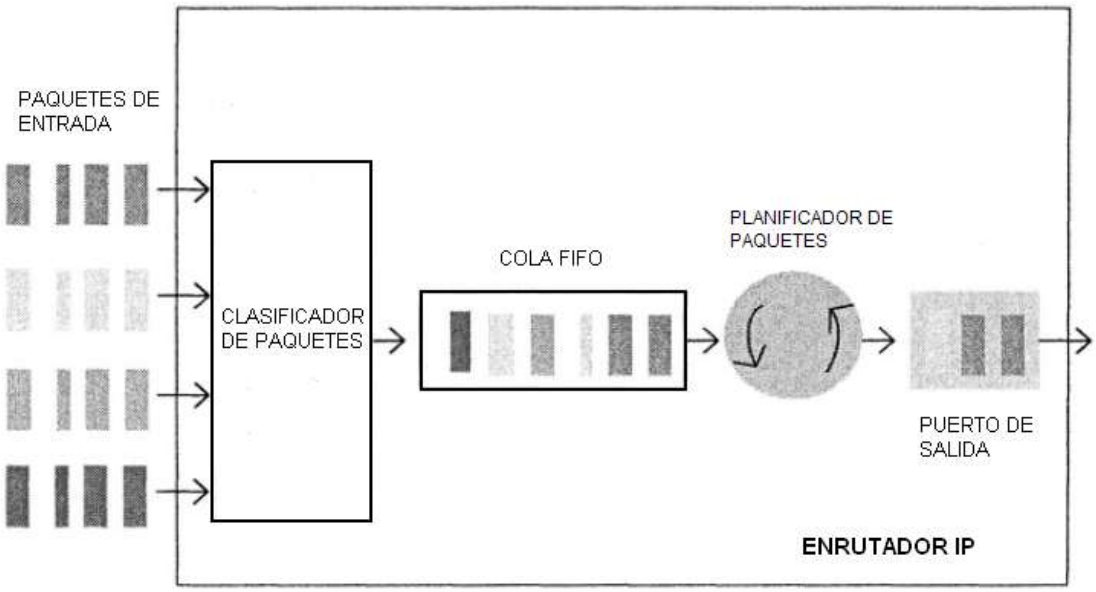

Figura 1.4 Mecanismo de encolamiento FIFO

[7] Primero en entrar, Primero en salir

1.2.2 Encolamiento de Prioridad (*Priority Queuing*, PQ)

Este tipo de encolamiento es una manera sencilla de implementar distinción de clases; en PQ, como se muestra en la figura 1.5, se crean N colas con un ranking de prioridad de 1 a N. El orden de planifcación de los paquetes se define según el orden de prioridad y según la existencia de paquetes en las colas de mayor prioridad, los paquetes en la j^{aba} cola no son procesados si existen paquetes en las colas de prioridad mayor o sea en las colas de prioridad 1 hasta j-1. Esta es una manera simple de implementar diferenciación de tráfico.

El principal inconveniente de este mecanismo es un fenómeno conocido como ***starvation*** [5] [6], el cual ocurre cuando las colas de mayor prioridad siempre tienen paquetes para ser procesados y entonces los paquetes que se encuentran en colas de menor prioridad corren el riesgo de no tener oportunidad para enviar sus paquetes y pueden privarse por completo de acceder al ancho de banda del puerto de salida. Este problema tiene muy alta probabilidad de presentarse, lo cual debe tenerse en cuenta al utilizar PQ.

Este mecanismo es una buena alternativa cuando el tráfico de alta prioridad representa una pequeña fracción del tráfico total de las colas. PQ es conveniente para aplicaciones de tiempo real como vídeo y voz, las cuales típicamente utilizan UDP [7]; utilizar PQ para TCP puede ser un riesgo ya que el comportamiento de TCP durante congestión podría agravar el fenómeno de *starvation* para el resto de tráfico de las otras colas.

Para mejorar estas falencias de PQ y asegurar un mínimo de ancho de banda para las colas de baja prioridad se utiliza el mecanismo de PQ de tasa controlada, con el cual los paquetes de las colas de alta prioridad son tratados antes que los paquetes de baja prioridad solo si la cantidad de tráfico en las colas de alta prioridad permanece por debajo de un nivel umbral.

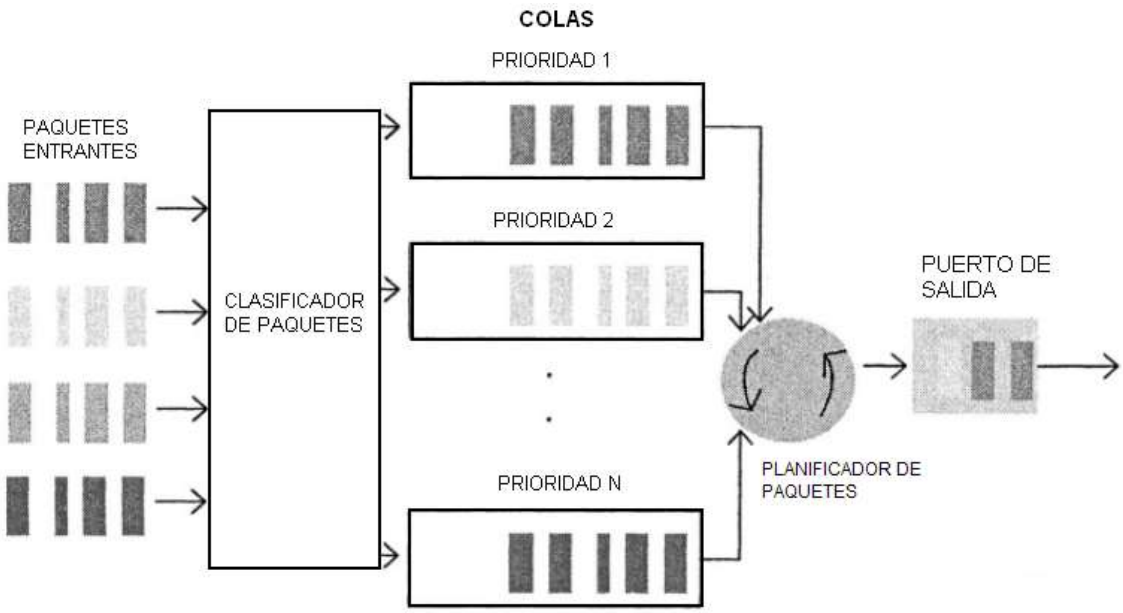

Figura 1.5 Mecanismo de encolamiento PQ

1.2.3 Encolamiento Equitativo (*Fair Queuing*, FQ)

Este tipo de encolamiento es otra forma muy general de crear distinción entre clases, también es conocido como encolamiento por flujos o encolamiento basado en flujos[8].

Como se muestra en la figura 1.6, los paquetes entrantes se clasifican en N colas, a cada cola se le asigna 1/N del ancho de banda total de salida, el planificador de paquetes asigna turnos a las colas acorde a un orden ***Round Robin***[9] saltándose las colas vacías. Para cada turno asignado por el planificador a una cola, se transmite a la salida un paquete de esta cola. Este es un mecanismo simple, el cual es de fácil escalabilidad ya que si se agrega una nueva cola para crear una nueva clase de tráfico el planificador automáticamente ajusta el ancho de banda asignado a cada cola a 1/(N+1) del ancho de banda total de salida; el principal problema de este mecanismo está precisamente relacionado con la asignación equitativa del ancho de banda [6] [7]: si las clases del tráfico entrante tienen diferentes requisitos en cuanto al mismo, FQ no tendrá la posibilidad de distribuirlo entre las clases de acuerdo con los requerimientos que éstas tengan.

Dado que un paquete es transmitido desde cada cola en su respectivo turno sin importar el tamaño del paquete, este tamaño influye en la distribución del ancho de banda entre las colas aun cuando cada cola tenga su turno equitativamente; si por ejemplo, una cola en particular tiene paquetes de un tamaño más grande que las otras, esta cola utilizaría más del 1/N ancho de banda asignado.

La imposibilidad de este mecanismo para asignar ancho de banda a las clases de tráfico entrante acorde a sus requisitos, es corregido por el mecanismo ***Weighted Round Robin*** *(WRR)* [5][6], el cual divide el ancho de banda del puerto de salida acorde a las características de ancho de banda de las clases del tráfico entrante.

Inicialmente en el mecanismo WRR los flujos de tráfico entrante se agrupan en m clases y el ancho de banda del puerto de salida es distribuido entre las m clases de acuerdo con el peso[10] determinado para cada clase según sus requerimientos de ancho de banda; la suma de todos los pesos de cada una de las clases debe ser igual a 100%. Para cada una de las clases, los flujos individuales son planificados con FQ.

Como se muestra en la figura 1.7, WRR requiere de dos capas de planificación *Round Robin* [7]: inicialmente las clases tienen turnos en un orden *Round Robin*, esta es la primera capa. La segunda capa es implementada cuando el planificador se encuentra en una clase en particular, las FQ dentro de esa clase reciben turnos en orden *Round Robin*.

---

[8] *Per-flow queuing o flow-based queuing*

[9] Es uno de los algoritmos de planificación más sencillos, asigna turnos en intervalos iguales y en orden, sin implementar ningún tipo de prioridad

[10] Se refiere a la prioridad asignada a la clase.

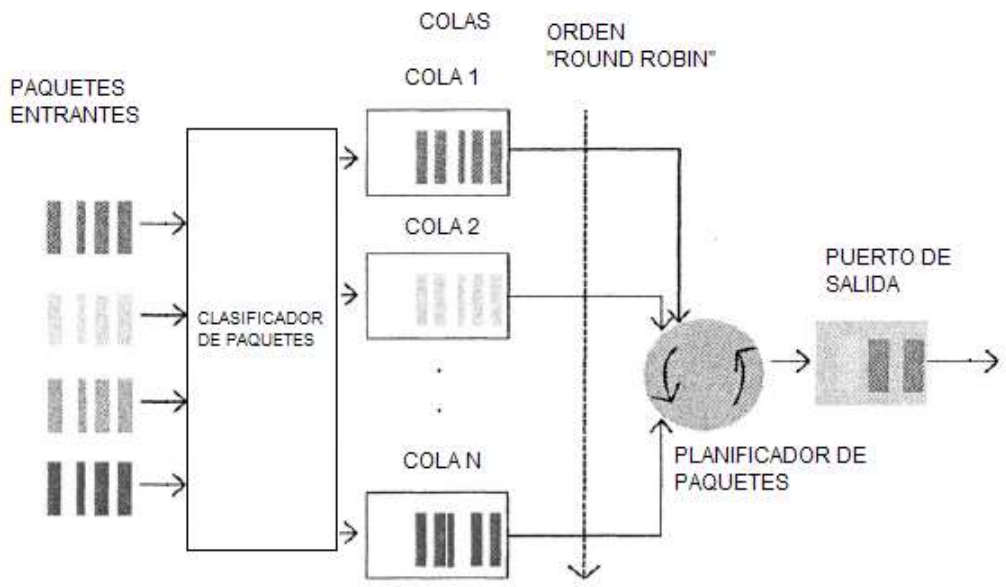

Figura 1.6 Mecanismo de encolamiento FQ

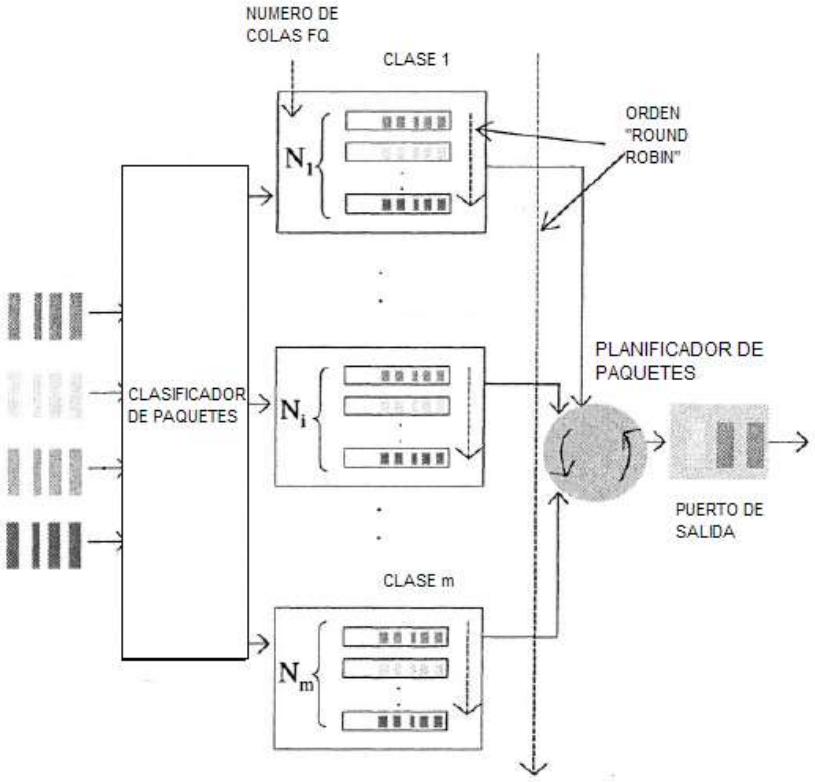

Figura 1.7 Mecanismo de encolamiento WRR

1.2.4 Encolamiento Equitativo por Pesos (*Weighted Fair Queuing*, WFQ)

El problema de FQ para distribuir el ancho de banda total del puerto de salida, de una forma acorde a los requisitos de ancho de banda de las clases de tráfico entrante se soluciona con el mecanismo WRR [2][5]. Se tiene todavía el inconveniente mencionado previamente del impacto del tamaño del paquete para compartir ancho de banda, esto dado que WRR utiliza FQ dentro de sus clases.

Una alternativa para este problema es el encolamiento equitativo por pesos o WFQ, en donde al igual que FQ los flujos de tráfico entrantes se agrupan en m colas, pero el ancho de banda es distribuido entre las m colas de acuerdo con los pesos apropiados determinados por los requisitos de ancho de banda de las clases, en vez de ser dividido equitativamente. En FQ cada cola envía un paquete por cada turno asignado por el planificador, mientras que en WFQ el planificador envía paquetes desde las colas basándose en el orden calculado de terminación del paquete; WFQ intenta aproximarse a un modelo teórico denominado **Planificador Bit a Bit *Round Robin* por Pesos** ("*Weighted bit-by-bit round robin scheduler*") [6], el cual asigna turnos a las colas en orden *Round Robin*, pero en cada turno el planificador toma un bit de la cola al tiempo, un ensamblador de paquetes almacena todos los bits del paquete y cuando este está reensamblado se envía. Este es un modelo teórico y no es práctico [2].

1.2.5 Encolamiento Equitativo por Pesos Basado en Clases (*Weighted Fair Queuing*, CB WFQ)

Este es un mecanismo similar a WRR, pues, los flujos de tráfico entrante se agrupan en m clases y el ancho de banda del puerto de salida se disribuye a estas m clases de acuerdo con los pesos apropiados, pesos que son predefinidos por los requisitos de ancho de banda de cada una de las clases, la diferencia está en que WRR utiliza FQ en cada una de sus clases, mientras que CB WFQ utiliza para sus clases WFQ; la figura 1.8 ilustra el proceso y es útil para aclarar las diferencias:

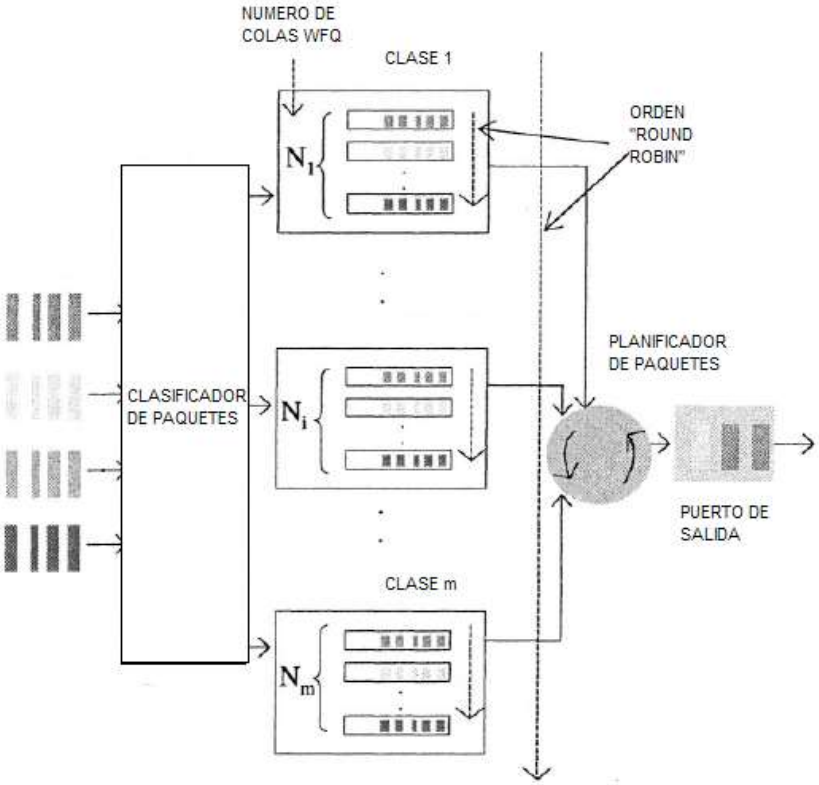

Figura 1.8 Mecanismo de encolamiento CB WFQ

1.2.6 *Deficit Round Robin* (DRR)

DRR mejora WRR al tener la capacidad de servir paquetes de longitud variable sin conocer previamente el tamaño promedio de paquete de las conexiones [8]. El algoritmo trabaja de la siguiente manera: se inicializa la variable *quantum* para representar el número de bits a ser servidos de cada cola. El planificador empieza sirviendo cada cola que tiene un paquete para ser atendido, si el tamaño del paquete es igual o menor que el *quantum*, el paquete es servido. Pero si el paquete es más grande que el tamaño del *quantum*, el paquete tiene que esperar otra ronda (*round*), en éste caso otro contador, llamado contador déficit, se inicializa para esta cola. Si un paquete no puede ser servido en una ronda, el contador déficit se incrementa en el tamaño del *quantum*. La herramienta HTB en el control de tráfico en Linux, que será explicada posteriormente en el capítulo 3 en la sección 3.6.9, utiliza DRR [9] como se define en [8].

1.2.7 División jerárquica de enlaces

Estos algoritmos permiten la planificación de paquetes basada en una estructura jerárquica. Una de sus principales aplicaciones es compartir el exceso de ancho de banda, el cual en vez de ser simplemente distribuido proporcionalmente entre las clases, puede ser configurable [3]. Por ejemplo en el caso de un enlace que es compartido entre dos usuarios, el ancho de banda no utilizado por una de las clases de tráfico de uno de los usuarios, está primero disponible para otras clases de tráfico de este mismo usuario. Solo si estas clases no tienen suficiente demanda, el exceso de ancho de banda estará disponible al otro usuario.

Los dos algoritmos principales de división de enlaces son el Encolamiento Basado en Clases (*Class Based Queuing, CBQ*) y el algoritmo de Curva Jerárquica de Servicio Justo (*Hierarchical Fair Service Curve Algorithm, H-FSC*).

1.2.7.1 Encolamiento basado en clases (CBQ)

El encolamiento basado en clases es una solución apta para compartir enlaces y aplicaciones con consideraciones de tiempo real. CBQ aísla el tráfico de tiempo real y el *best effort* permitiendo la separación de diferentes clases de tráfico, las cuales se tratan posteriormente de acuerdo con sus requerimientos. *Sally Floyd* y *Van Jacobson* [10] [11] desarrollaron un modelo para la división jerárquica de enlaces y muestran como el algoritmo de encolamiento basado en clases puede ser utilizado para configurar estructuras jerárquicas que permiten controlar la distribución del exceso de ancho de banda.

Las clases dentro de este mecanismo pueden tener las siguientes características [10] [11]:

- *Regulada:* Paquetes de esta clase se planifican mediante el planificador utilizado para compartir enlaces.
- *No regulada*: Los paquetes son planificados por un planificador genérico de la red.
- *Sobre el límite (overlimit)*: La clase ha utilizado recientemente más de su ancho de banda asignado.
- *Bajo el límite (underlimit):* La clase ha recibido menos de su asignación de ancho de banda.
- *Al límite*: La clase no está sobre o bajo él limite.
- *Insatisfecha:* Una subclase que está bajo el límite y tiene constantemente paquetes atrasados.
- *Satisfecha*: Una clase que no está insatisfecha.
- *Exenta:* La clase nunca estará regulada.

- *Limitada:* La clase no tiene permitido tomar prestado ancho de banda de sus clases ancestras.

- *Aislada:* Una clase que no permite a clases que no sean sus descendientes utilizar el ancho de banda sobrante y que además, no utiliza ancho de banda de otras clases.

Cada clase interior o subclase debe recibir como mínimo su ancho de banda designado en intervalos de tiempo apropiados cuando tiene demanda, y si todas las clases interiores con demanda han recibido al menos el ancho de banda designado; la distribución del ancho de banda sobrante no debe ser arbitraria sino que debe de seguir un procedimiento lógico [3] [10], de acuerdo con estos propósitos se derivan los siguientes parámetros de funcionamiento:

1. Una clase puede continuar no regulada si se mantiene una de las siguientes condiciones:

    a. La clase no está sobre el limite, o
    b. La clase tiene un ancestro no sobre el límite a un nivel i y la estructura de división de enlace no tiene clases insatisfechas a niveles menores que i.

2. Una clase continuará siendo regulada mientras una de las siguientes condiciones se mantenga:

    a. La clase está por debajo del límite, o
    b. La clase tiene un ancestro bajo el límite a un nivel i y la estructura de división de enlace no tiene clases insatisfechas a niveles menores que i.

1.2.7.1.1 Encolamiento Basado en Clases Mejorado con Clasificador de paquetes dependiente del estado del canal

En [12] el esquema CBQ se extiende para compartir canales inalámbricos, combinándolo con clasificación de paquetes según el estado del canal. Esta mejora propone utilizar el RTS-CTS (*Request to Send / Clear To Send*) para censar la calidad del enlace a un destino específico. Si el enlace está en mal estado, entonces, se selecciona un paquete con un destino diferente, evitando problemas de bloqueo, además, el planificador almacena la calidad del enlace utilizando un parámetro denominado “g” el cual se incrementa inversamente proporcional al número de intentos de RTS-CTS efectuados antes de detectar un buen enlace. La idea es que en un enlace que tuvo buena calidad en el pasado, es más factible encontrar buenas condiciones en el futuro. El planificador prueba cada estación hasta g veces, si se detecta un buen canal antes de g pruebas, el parámetro se incrementa, sino se recibe un CTS después de g intentos éste se decrementa.

Otro cambio realizado al algoritmo CBQ, es que a una clase regulada se le permite transmitir, aunque existan clases insatisfechas, solo cuando todas estas clases insatisfechas experimentan malas condiciones del canal. Simulaciones realizadas en [12] demuestran que al implementar estos cambios, el algoritmo tiene la capacidad de alcanzar *throughputs* más altos.

1.2.7.2 Curva Jerárquica de Servicio Justo (*Hierarchical Fair Service Curve*, H-FSC)

Este algoritmo, para compartir enlaces desacopla los retardos y la asignación de ancho de banda [3]. Se basa en el concepto de curva[11] de servicio, el cual define los requisitos de calidad de servicio de una clase de tráfico en términos del ancho de banda y la prioridad.

Cada clase en la jerarquía tiene asociada una curva de servicio. El exceso de ancho de banda se distribuye de acuerdo con estas curvas y un servicio que reciba este exceso no es penalizado posteriormente. Esto es cuestionable, ya que en [13] los autores prueban que con curvas de servicio no lineales (las cuales son necesarias si el ancho de banda y los retardos se van a desacoplar) es imposible evitar periodos en los cuales el planificador no tiene la capacidad de garantizar las curvas de servicio de todas las clases o no tiene la posibilidad de garantizar tanto las curvas de servicio como las propiedades de ecuanimidad.

Para este algoritmo en [14] también se ha demostrado, como la planificación del tráfico en esquemas *best effort* puede mejorarse utilizando H-FSC; impidiendo el trato preferencial del tráfico de larga duración, continuamente retrasado, como la transmisión de archivos, comparado con el tráfico *best effort* de ráfagas, como el WWW.

1.2.8 Planificación de paquetes en redes inalámbricas

La planificación de paquetes en entornos inalámbricos es más complicada que en redes cableadas [3] [10]; principalmente por qué el canal inalámbrico es afectado por factores adicionales, que los mecanismos de planificación tienen que considerar. Estos factores son:

- *Errores dependientes de la ubicación* (como por ejemplo: errores de desvanecimiento multi-trayecto). En el caso de la figura 1.9, mientras que las estaciones A y B tienen toda la capacidad del enlace disponible, el usuario C experimenta problemas de ubicación que le impiden utilizar la capacidad del enlace.

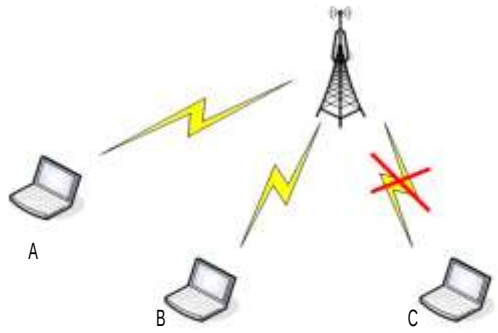

Figura 1.9 Errores en el canal inalámbrico dependientes de la ubicación

[11]Con el propósito de desacoplar retardos y asignación de ancho de banda, HFSC se diseña con base en el Modelo de Curvas de Servicio. HFSC, por simplicidad, solamente utiliza curvas de servicio lineales de dos componentes. En el Modelo de Curvas de Servicio, a cada clase se le asigna una curva de servicio, la cual especifica la mínima cantidad de servicio que esta debe recibir.

- *Mayor probabilidad de transmisión de ráfagas con errores*
- *Interferencia con otras fuentes de radio*
- *Incremento/decremento dinámico del número de estaciones*

La mayoría de algoritmos de planificación asumen que se planifica el tráfico de bajada en un punto de acceso (AP) y que tienen control total sobre el medio inalámbrico [15]. La mayoría asumen también un conocimiento exacto del estado de las colas del enlace de subida de las estaciones móviles y la condición del canal [3]. En entornos de simulación se utiliza comúnmente un modelo de cadena de Markov para los estados del canal inalámbrico, conocido como el *Modelo Gilbert-Elliot*, el cual consiste de dos estados "buen canal" y "mal canal" y sus probabilidades de transición [16].

En el caso de las redes cableadas, usualmente todas las estaciones experimentan las mismas condiciones del canal; en el caso inalámbrico es muy normal que algunos usuarios tengan buenas condiciones de canal y otros no, pero fracciones de segundo después, las condiciones pueden cambiar completamente [17]. Por esto puede ser necesario brindar a los usuarios que han experimentado problemas, capacidad adicional del canal una vez presenten buenas condiciones de transmisión; esto se conoce como el mecanismo de compensación inalámbrica.

Los mecanismos de planificación de paquetes en redes inalámbricas más utilizados son el encolamiento equitativo inalámbrico (*Wireless Fair Queuing, WFQ*) y una adaptación de este mecanismo para mejorar los errores dependientes de la ubicación de la estación que es el encolamiento equitativo inalámbrico idealizado (*Idealized Wireles Fair Queuing, IWFQ*) [3], estos mecanismos utilizan implícitamente la compensación inalámbrica. Otro mecanismo muy utilizado es la planificación de paquetes inalámbricos (*Wíreless Packet Scheduling, WPS*) que es una aproximación al mecanismo IWFQ que utiliza WRR y asume que todos los paquetes tienen el mismo tamaño.

1.3 CONFORMACIÓN DE TRÁFICO

Como una alternativa o como un complemento, considerado en aplicaciones como HTB que puede utilizar disciplinas de colas diferentes de FIFO, para controlar la prioridad de ciertos tipos de tráfico, existen otros métodos como la conformación y el control de admisión, los cuales se utilizan para controlar qué tráfico es transmitido por la red o la tasa a la cual es admitido.

Existen varios esquemas para la conformación de tráfico y para el control de admisión, algunos de los cuales son utilizados como tecnologías únicas y otros que se integran a otras tecnologías como en la arquitectura de servicios integrados (*IntServ, Integrated Services Architecture*) [18]. La conformación de tráfico cambia el desempeño del flujo de tráfico. El control de admisión, como su nombre lo dice se encarga de discriminar que tráfico es "admitido" en la red; existe también el método de *policing*, el cual se encarga de examinar en cada salto dentro de la red, si en la entrada se siguen las políticas de conformación de tráfico preestablecidas u otros mecanismos reconocibles. El control de admisión y el *policing* son explicados posteriormente en el capítulo 2, en las secciones 2.1.2 y 2.1.5 respectivamente.

Tanto los esquemas de conformación como los de control de tráfico deben implementarse en los extremos de la red para así controlar el tráfico entrante [3]; esta observación es importante para el trabajo desarrollado en este proyecto, pues el control se implementa en los extremos LAN-WLAN, WLAN-LAN, como se indica en la figura 1.10.

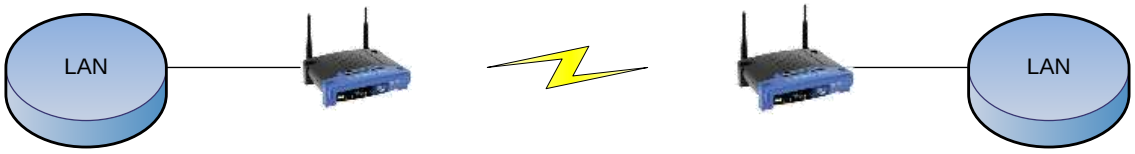

Figura 1.10 Implementación del control de tráfico en los extremos LAN-WLAN, WLAN-LAN

La conformación de tráfico es un mecanismo que se utiliza para controlar la cantidad y el volumen de tráfico enviado a la red, además de la tasa a la cual éste se envía. También puede ser necesario identificar flujos de tráfico en el punto de ingreso a la red, con una granularidad tal que permita al mecanismo de control de conformación de tráfico separarlos en flujos individuales y conformarlos diferente.

Entre los métodos de conformación de tráfico, predominan dos [2][18]: la implementación *Leaky-Bucket* y la implementación *Token-Bucket*.

1.3.1 Implementación *Leaky Bucket*

Su nombre se refiere a un contenedor virtual (balde o *bucket*) el cual tiene agujeros y está goteando (*leaky*). Esta implementación se utiliza para controlar la tasa a la cual el tráfico se envía a la red, brinda un mecanismo con el cual el tráfico con alteraciones y desordenado puede “conformarse” en flujos constantes y ordenados dentro de la red (ver figura 1.11).

Esta implementación se puede entender realizando una analogía con una autopista de varios carriles que posteriormente convergen a uno solo, se debe implementar un intervalo de admisión regulado para ayudar el movimiento del tráfico. El beneficio de esta implementación es que el flujo de tráfico entrante en la red es predecible y controlado. El principal inconveniente es cuando el volumen de tráfico es ampliamente mayor que el tamaño del contenedor virtual (*bucket*) [18].

Figura 1.11 Implementación *Leaky-Bucket*

Esta implementación fue inicialmente diseñada para controlar la tasa a la cual el tráfico se transmite en una red ATM, pero actualmente se utiliza en aplicaciones de nivel 3 [18]. El tamaño del contenedor y la tasa de transmisión son configurados generalmente por el usuario y medidos en bytes. Este mecanismo utiliza una medida de tiempo para indicar cuando el tráfico en una cola FIFO puede transmitirse, para controlar la tasa a la cual éste se inyecta a la red; es posible que el contenedor se llene y el tráfico subsiguiente sea descartado. Este es un mecanismo muy simple para controlar la tasa a la cual el tráfico es transmitido a la red.

Leaky Bucket presenta una deficiencia, la cual en proyectos como éste, donde se busca optimizar la utilización de los recursos de red se hace bastante notoria: la tasa a la cual se inyecta la información a la red es un parámetro fijo, se presentarán muchas ocasiones en las cuales el volumen de tráfico es muy bajo y porciones de los recursos de red (ancho de banda) no se están utilizando. No existe un mecanismo en la implementación para permitir que flujos independientes utilicen recursos de red que están siendo subutilizados.

1.3.2 Implementación *Token Bucket*

Este es otro mecanismo de conformación de tráfico y de control de ingreso, bastante diferente que la implementación *leaky bucket*. En la implementación anterior, el contenedor se llena con tráfico y continuamente transmite este tráfico a una tasa fija, en el *token bucket* el tráfico no transita el contenedor virtual, como se ve en la figura 1.12.

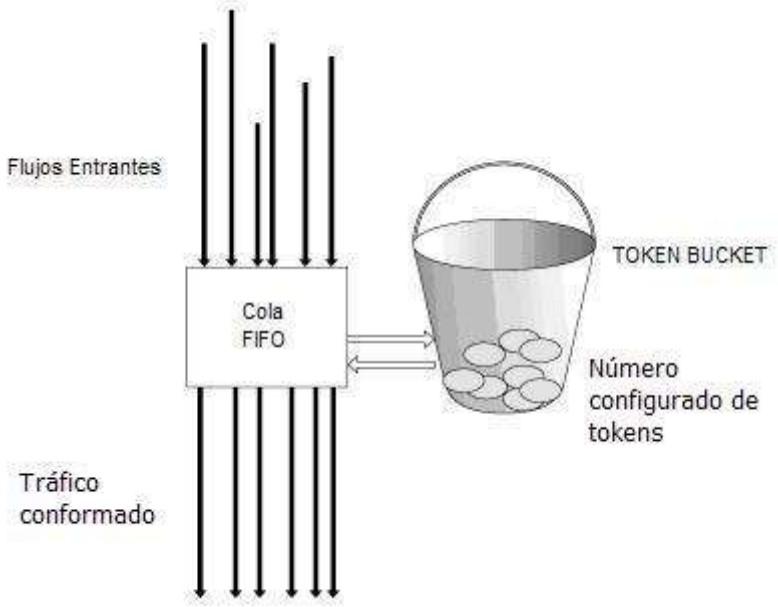

Figura 1.12 Implementación *Token Bucket*

Token Bucket dictamina cuándo el tráfico puede ser transmitido basándose en la presencia de *tokens*[12] en el contenedor virtual. Este es más eficiente para utilizar los recursos de red, ya que permite que los flujos de tráfico tengan niveles de transmisión configurables, lo cual es una ventaja aprovechada para la implementación de éste proyecto. El contenedor tiene los *tokens*, cada uno de estos puede representar una unidad de bytes, el administrador de red determina cuantos *tokens* se requieren para transmitir un número de bytes.

Cuando hay *tokens*, el flujo tiene permitido transmitir tráfico, si no los hay no se pueden transmitir paquetes. Así cada flujo puede transmitir tráfico hasta su tasa pico si existen *tokens* adecuados en el contenedor y si los niveles están configurados adecuadamente. El mecanismo HTB, base de este proyecto, se generó como una modificación del *Token Bucket* original y se explica posteriormente en el capítulo 3, en la sección 3.6.9.

1.3.3 Múltiples *Token Bucket*

Una variación de la implementación *Token Bucket* con un solo contenedor es incluir múltiples contenedores y múltiples umbrales, como se muestra en la figura 1.13. En este caso, un clasificador de tráfico puede interactuar con *token buckets* separados, cada uno con diferentes umbrales pico, permitiendo que diferentes clases de tráfico sean conformadas de manera independiente.

[12] En Telecomunicaciones este término también se conoce como paso de testigo. Se utiliza para asignar turnos.

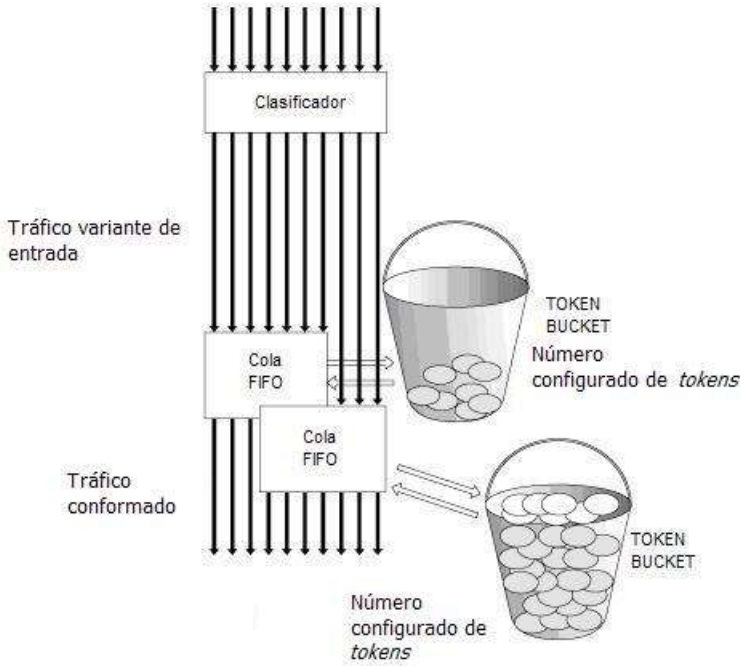

Figura 1.13 Múltiples *Token Bucket*

Otra variación de estos dos esquemas, es una combinación del *leaky bucket* y el *token bucket*. La razón para este tipo de implementación es que mientras el *token bucket* permite que flujos individuales alcancen su tasa pico, este también permite que estos flujos dominen los recursos de red en el periodo de tiempo durante el cual los *tokens* estén presentes en el contenedor o hasta el umbral definido. Esta situación, si no está configurada correctamente, permite a algún tráfico utilizar más ancho de banda que otros y puede interferir con la capacidad de otros tráficos para transmitir adecuadamente, dada la disputa por los recursos en la red [18]. Un *leaky bucket* se puede utilizar para acomodar las tasas de tráfico a niveles umbrales configurables, después que el tráfico ha sido sujeto a un *token bucket*.

Una aplicación muy importante de los *token buckets* es en el medidor utilizado para implementar el marcado de tres colores de tasa única (*Single Rate Three Color Marker*) y de dos tasas (*Two Rate Three Color Marker*), que puede usarse en el acondicionamiento de tráfico en la arquitectura DiffServ, explicados posteriormente en el capítulo 2 en la sección 2.2.1 y 2.2.2 respectivamente. Este concepto de colores también se aplica a HTB, y su implementación específica, en éste caso, será explicada en el capítulo 3 en la sección 3.6.9.

RESUMEN

En este capítulo se describen tres piezas fundamentales para brindar QoS, como se explica posteriormente en éste documento, y para la implementación del control de tráfico de este proyecto: el encolamiento, la planificación de paquetes y la conformación de tráfico.

La teoría de colas modela el proceso de las filas de espera, que en el caso de redes de paquetes es una pieza esencial para la diferenciación y la asignación de prioridades. En este proyecto se utilizan colas propias del control de tráfico en Linux (HTB por defecto utiliza colas tipo FIFO), cada una correspondiente a una prioridad de tráfico. Esta teoría puede ser estudiada más a fondo y brinda la posibilidad de realizar extensos análisis matemáticos, pero en el caso de este documento se incluye la información más importante para el proyecto y que en general se debe conocer para entender el concepto.

El planificador DRR que se describió en este capítulo, es clave para el funcionamiento de HTB, pues éste lo implementa dentro de su funcionamiento global. Así mismo el concepto de *Token Bucket*, ya que, como su nombre lo indica (*Hierarchical Token Bucket* o *Token Bucket Jerárquico*), HTB utiliza múltiples *token buckets*, como se describe posteriormente en el capítulo 3, sección 3.6.9

2. CALIDAD DE SERVICIO

Las redes de computadores dependen de la forma de compartir los recursos limitados de la red para las comunicaciones que por ellas se puedan establecer. Si estos recursos fueran ilimitados, un enlace directo punto a punto, de ancho de banda ilimitado, sería posible entre dos usuarios finales que quisieran comunicarse. Obviamente, esto no es posible en el contexto del mundo real. Razón por la cual, en aplicaciones reales, dos usuarios finales que quieran comunicarse, a menos que estén conectados a la misma LAN, necesitan establecer una conexión mediante un conjunto de enlaces que tienen características como:

- Son compartidos entre tráfico de diferentes fuentes hacia múltiples destinos.
- Tienen recursos limitados.
- Tienen retardos de propagación.
- Los nodos conectados directamente a cada enlace tienen niveles limitados de procesamiento y de memoria.

¿Cómo se deben compartir estos recursos limitados? La forma más sencilla es tratar todo el tráfico igual, o sea, darle el mismo tratamiento a todos los paquetes. Así, todos son procesados en el orden en el cual llegan, de forma FIFO como se mencionó en el capítulo 1 de este documento. Si un número de paquetes llega a un nodo cuyos *buffers* están llenos, los paquetes subsecuentes muy probablemente serán descartados. Internet, al igual que muchas otras redes, se comporta de éste modo.

Este tratamiento simple para los paquetes, no es suficiente en las redes multi-servicio actuales, las cuales transportan una gran variedad de servicios y tipos de tráfico. Diferentes aplicaciones tienen distintos requerimientos de los servicios de comunicación brindados por la red, por ejemplo, la voz y el video son exigentes ante los retardos y la latencia. Esta clase de demandas, sumada a otras como el ancho de banda garantizado, se conocen como *Requerimientos de Calidad del Servicio* y el conjunto completo de exigencias y la forma como son cumplidas en las redes, se conoce como *Calidad de Servicio* (QoS) [1].

Algunos usuarios de redes están dispuestos a pagar una tarifa más alta (*Premium)*, para lo cual la red debe tener la capacidad de brindar prioridad al tráfico de estos usuarios, o garantizar cierto ancho de banda. Este concepto puede generalizarse a requerimientos para soportar múltiples clases de servicio. En el escenario empresarial, donde se ubica este proyecto, es una buena alternativa priorizar varios tipos de tráfico basándose en las necesidades y requerimientos del negocio. Por ejemplo, tráfico urgente de videoconferencias puede ser más importante que la navegación Web, en la red de una misma empresa. En un contexto militar, las comunicaciones de un general pueden tener mayor prioridad sobre las comunicaciones de oficiales de menor rango; con este ejemplo es importante aclarar que: las consideraciones de seguridad en redes inalámbricas, no son objeto de este proyecto y no se mencionan en este documento.

Este capítulo se enfoca en las bases para brindar QoS, en particular en las redes IP y también se consideran las redes IP inalámbricas, las cuales son el escenario específico del desarrollo de este proyecto. En este capítulo no se consideran las métricas de QoS, las cuales son abordadas en el siguiente, ni tampoco modelos de precios y tarificación.

Se describen algunos mecanismos de QoS utilizados actualmente, que de una forma u otra, se involucran en el desarrollo total del proyecto. Estos pueden encontrarse en enrutadores y otros equipos de red, pero su inclusión no es obligatoria y esta ligada a la tecnología de red donde se desempeñe el dispositivo. Es una alternativa viable que los mecanismos de QoS sean coordinados en una misma arquitectura como: la Arquitectura de Servicios Integrados (IntServ) y la Arquitectura de Servicios Diferenciados (DiffServ).

2.1 INTRODUCCIÓN A LA CALIDAD DE SERVICIO EN REDES IP

El Internet actual es una red *Best Effort*. No se brindan garantías como: que los paquetes no se perderán, los tiempos de propagación de estos pueden ser controlados, que los paquetes viajen en un determinado camino o que lleguen en secuencia, entre otros. Por esto, se dice que Internet brinda un servicio *Best Effort*, o que hace lo que puede por llevar la información del origen al destino. Internet ha evolucionado de ser una red de investigación a ser una utilizada por millones de usuarios, corporaciones, entidades, etc. los cuales demandan un servicio de mayor calidad e incluso están dispuestos a pagar por ello. También, con el incremento en el rango de aplicaciones utilizándolo, se presenta un incremento en los requerimientos de red de estas aplicaciones.

La industria de las telecomunicaciones está en un contínuo desarrollo de mecanismos para soportar diferenciación de servicios, asignando diferentes niveles y tipos de garantías de servicio a diferentes flujos de tráfico, incluyendo garantizar tiempos más rápidos ó mayor probabilidad de encontrar disponible el *throughput* deseado. Los requerimientos de diferenciación de servicios son fundamentales para brindar QoS en redes IP [1]. Dada la alta utilización de la red, niveles altos de calidad del servicio no pueden ser brindados a cada uno de los usuarios. Para una solución de QoS en redes IP se han considerado importantes, entre otros, tres requerimientos fundamentales:

- Brindar mecanismos para ofrecer diferentes niveles de servicio a diferentes usuarios, aun así algunos reciban mejor servicio que otros.
- El servicio no debe ser, usualmente, negado por completo al tráfico de baja prioridad.
- Los mecanismos deben ser eficientes, sin desperdiciar recursos importantes de red.

Algunos de los principales propósitos que se consideran necesarios para implementar diversos mecanismos de QoS son:

1. Cómo reservar recursos de red para ciertos flujos.
2. Cómo controlar la admisión de tráfico a determinada red.
3. Cómo controlar la secuencia de paquetes procesados o reenvidados por un nodo; en otras palabras encolamiento de entrada o de salida.
4. Cómo controlar la tasa del flujo de tráfico.
5. Cómo monitorear la conformidad de los flujos con respecto a perfiles.
6. Cómo controlar el enrutamiento de diferentes flujos en la red.

2.1.1 Reserva de recursos

Este es uno de los pasos principales para abandonar el esquema *best-effort*; se busca reservar recursos para algunos tipos de tráfico, mientras se sigue implementando el enrutamiento IP tradicional; este mecanismo consiste en arreglos dinámicos para brindar calidad de servicio extremo a extremo sobre una o más redes intermedias. Usualmente la reserva se especifica con un perfil de tráfico el cual describe la tasa y otras características del mismo [2].

2.1.1.1 Protocolo para Reserva de Recursos (RSVP)

Es un protocolo de señalización que permite coordinar la reserva de recursos en una red [3]. La señalización con RSVP generalmente reserva los recursos en enrutadores que soporten el protocolo, en el tramo transitado por el flujo de tráfico. RSVP es unidireccional [1], los recursos se reservan en una vía desde el origen hacia el destino solamente. Tradicionalmente, este protocolo ha sido asociado con la arquitectura de de servicios integrados IntServ, como se explicará posteriormente en la sección 2.2.

Es importante mencionar la aplicación de este protocolo en la reserva de recursos, pero dado que no es utilizado en el proyecto, no se describe detalladamente. Se puede encontrar información ampliada en [3].

2.1.2 Control de admisión

El control de admisión se refiere a la decisión de admitir cierto flujo o agregados de flujos en la red, en respuesta a un pedido de reserva de recursos [1]. Es coherente que las decisiones sobre control de admisión sean hechas por una entidad con una visibilidad total de los recursos de red, esta es la idea detrás de los “negociadores” ó agentes de ancho de banda (*Bandwidth Brokers*), explicados posteriormente en la sección 2.1.2.1 de este documento.

Los mecanismos de conformación de tráfico explicados en el capítulo 1, sección 1.3 como el *Token Bucket*, colocados en los extremos de la red pueden verse como una forma de control de admisión leve, ya que estos pueden disminuir la tasa del tráfico entrante a la red, en vez de negar el acceso completamente.

2.1.2.1 “Negociadores” de Ancho de Banda

Para que la asignación de ancho de banda en los enrutadores de una red sea dinámica existen dos posibilidades: que cada enrutador tome sus propias decisiones, ya sea configurado manualmente, como en este proyecto, donde se modifican las características de funcionamiento del enrutador inalámbrico o por su funcionamiento por defecto, o la otra opción es que una sola entidad tome las decisiones para la red, tomando en consideración las condiciones dinámicas; a estas entidades se les conoce como “Negociadores” ó Agentes de Ancho de Banda.

Por la naturaleza de sus funciones, es típico que cada dominio tenga su propio “negociador” para controlar sus recursos de ancho de banda. Este debe ser el encargado de negociar, con el “negociador” de ancho de banda de otro dominio, los

recursos para el tráfico ínterdominios, tomando decisiones como por ejemplo, admitir ó no, pedidos para reservar recursos.

2.1.3 Marcado de paquetes

Con este proceso se configuran los bits binarios de los campos apropiados en la cabecera IP para especificar valores que permitan distinguir un tipo de paquete IP de otro. Por ejemplo, un paquete puede distinguirse por su dirección fuente (o dirección de origen), su dirección de destino o una combinación de ambas; otro ejemplo es configurar en un valor específico el DSCP (*DiffServ Code Point*) del campo DiffServ de un paquete [4].

Para distinguir entre tipos de paquetes IPv4, se utiliza el campo **Tipo de Servicio** (*Type of Service, ToS*), de los 8 bits de este campo, 3 bits se utilizan para identificar la precedencia IP, 4 bits para el tipo de servicio y un bit no es utilizado. Los 4 bits del tipo de servicio, pueden interpretarse de diferentes formas por varios enrutadores, incluso, algunos no lo soportan y simplemente lo ignoran.

De manera ilustrativa se muestran las marcas colocadas en los campos respectivos en IPv4 y en la arquitectura DiffServ; en este proyecto los paquetes procesados en los enrutadores inalámbricos no modifican la información en la trama y las marcas utilizadas son solo reconocibles por el control de tráfico en Linux implementado en los dispositivos, esto con el fin de que no se afecte el tratamiento que otras redes deberían dar al paquete, cuando ingrese a otra red o sea procesado por otro dispositivo. También, es importante aclarar que las marcas de IPv4 y de DiffServ, son reconocidas en el control de tráfico de este proyecto y se asigna una prioridad acorde a éstas, utilizando las marcas propias de Linux. En el RFC 1349 [5] se define la semántica de los 4 bits de tipo de servicio como se muestra en la tabla 2.1.

Tabla 2.1 Semántica para Tipo de Servicio en IPv4

ToS	**Semántica**
1000	Minimizar retardo
0100	Maximizar *throughput*
0010	Maximizar fiabilidad
0001	Minimizar costo monetario
0000	Servicio normal

En el RFC 791 [6] se discuten los valores del campo de precedencia IP, estos valores se muestran en la tabla 2.2.

Tabla 2.2 Semántica para Precedencia IP IPv4

ToS	**Semántica**
111	Control de red
110	Control “Internetwork”
101	Crítico/ECP
100	“Flash override”
011	“Flash”
010	Inmediato
001	Prioridad
000	Rutina

En DiffServ los paquetes que se marcan con DSCP’s específicos, reciben tratamiento especial en cada salto. El *byte* DS puede utilizarse para marcar los paquetes con diferentes códigos, de esta forma, se pueden configurar los enrutadores intermedios, para que paquetes con códigos especiales se reenvíen de forma diferente. El campo ToS de IPv4 y el campo de clase de tráfico en IPv6, se han renombrado como *byte DS* en el RFC 2474 [7]. DSCP utiliza 6 bits, por lo cual existen 64 diferentes posibles combinaciones. La tabla 2.3 muestra la configuración del DSCP.

Tabla 2.3 Configuración del Campo DSCP

Pool	**Codificación**	**Propósito**
1	xxxxx0	Acción estándar
2	xxxx11	Acción experimental/local
3	xxxx01	Acción experimental/local (sujeto a estandarización)
Por defecto	000000	Reenvío *Best Effort*
	xxx000	Para compatibilidad de precedencia IP

En general, la idea de DiffServ es brindar QoS implementando en los enrutadores de los extremos de red clasificación, control y marcado en las cabeceras de los paquetes IP, acorde a diferentes criterios como: dirección de origen, tipo de aplicación, contrato de suscripción, protocolo de transporte, etc. [8]. Posteriormente en la sección 2.2 se explica el funcionamiento y la arquitectura de DiffServ, así como también la relación de la configuración del DSCP con las diferentes prioridades de servicio.

2.1.4 Clasificación de paquetes

La clasificación se ha mencionado previamente, especialmente en la sección anterior. El marcado y la clasificación de paquetes son conceptos relacionados y a pesar de que parecen muy similares, tienen diferencias importantes. Estas diferencias se pueden aclarar fácilmente mediante una analogía: un grupo de 20 estudiantes puede ser “marcados” con números del 1 al 20 y los estudiantes pueden “clasificarse” en dos grupos, los marcados con números pares y los marcados con números impares.

Las reglas ó criterios de clasificación separan los paquetes en diferentes clases, por ejemplo, pueden basarse en el puerto (el de origen ó de destino de la comunicación), el protocolo de transporte, en la aplicación, en la dirección (ya sea dirección origen ó destino del paquete), entre otros. También es posible que paquetes que no encajen en un perfil sean reclasificados, típicamente en una clase de menor prioridad [8]. Existen dos tipos de métodos para la clasificación de tráfico: la clasificación Multi-campo (*Multi-Field,MF*) y la clasificación de comportamiento agregado (*Behavior Agregate, BA*) [9]. En la clasificación MF los paquetes se clasifican según la combinación de los valores de uno o más campos de la cabecera, además, se tienen en cuenta otros parámetros, como los previamente mencionados. La clasificación BA se basa únicamente en los valores del DSCP [4].

La clasificación de paquetes utilizada en este proyecto, se define específicamente en el capítulo 4 en la sección 4.2.1, en ésta se utilizan diferentes filtros para la identificación del tráfico cuya implementación es posible en las versiones más recientes del control de tráfico en Linux, estos (filtros) se describen en el capítulo 3 en la sección 3.6.6.3.

2.1.5 Policing

Existen muchas definiciones para el término *policing*, que en general se refiere al procesamiento de paquetes que no cumplen con un perfil definido. Este es un proceso de vigilancia el cual básicamente se encarga de determinar si el tráfico es ó no conforme. Los paquetes de un flujo que no son conformes pueden ser descartados ó remarcados como no conformes y se tratan diferente que los paquetes que son conformes, generalmente se les cambia la prioridad ó se les brinda un tratamiento *best-effort* [2], previniendo que este flujo afecte otro tráfico a través del nodo.

La implementación *Token Bucket* mencionada previamente en el capítulo 1, es referida en algunas ocasiones como T*oken Bucket* de Conformación, existe también un T*oken Bucket* de P*olicing* [2], la diferencia se presenta en lo que sucede cuando hay o no hay suficientes *tokens*. En la implementación de conformación, el paquete espera hasta que haya suficientes *tokens,* para poder pasar, en la implementación de *policing* el paquete es descartado inmediatamente o reclasificado y remarcado [2].

2.1.6 Gestión Activa de Colas

AQM (*Active Queue Management, AQM*) es un mecanismo de control de congestión y su principal objetivo es prevenir la sincronización TCP[13]. La idea de AQM es anticiparse al comienzo de una congestión y reaccionar para prevenir o mitigar los efectos de la congestión. Existen tres métodos principales de AQM:

- Descarte Prematuro Aleatorio (*Random Early Discarding, RED*)
- Descarte Prematuro Aleatorio Por Pesos (*Weighted Random Early Discarding, WRED*)
- Notificación Explicita de Congestión (*Explicit Congestion Notification, ECN*)

[13] Transmission Control Protocol. El Protocolo de Control de Transmisión es uno de los protocolos fundamentales en Internet

Cuando un enrutador no implementa el mecanismo de AQM, el mecanismo por defecto que este utiliza es el método de "*tail-drop"* [9]. Este es una técnica de gestión pasiva de colas, mediante la cual los paquetes rebosados se descartan automáticamente cuando la cola está llena; la principal ventaja de esta técnica es su simplicidad, pero tiene un problema que es la causa del fenómeno conocido como sincronización global TCP [9] [10].

Este fenómeno ocurre cuando el *host* emisor TCP recibe una negativa *NAK*[14] indicando que el paquete TCP se extravió durante su transmisión en la red, el emisor asume que el paquete se perdió por congestión en la red y para ayudar con este problema automáticamente disminuye la tasa de transmisión de paquetes. Si el *buffer* está lleno, los paquetes entrantes se descartan indiscriminadamente, si estos paquetes son TCP, todas las sesiones TCP asociadas con los paquetes descartados disminuirían su velocidad. Dado que las sesiones TCP afectadas reaccionan de una manera sincronizada, el estado de congestión oscilará entre: estado de congestiones máximas y de no congestión; este fenómeno causa ineficiente utilización de los recursos de red como son espacio en el *buffer*, ancho de banda en el puerto de salida, etc. [9].

RED trabaja descartando de forma aleatoria paquetes de diferentes conexiones. Éste utiliza el principio de ventana deslizante (*sliding window*) y los mecanismos de recuperación rápida de TCP/IP, para forzar la comunicación a reducir la velocidad a la cual está transmitiendo los paquetes, reduciendo así la utilización de ancho de banda de esa conexión en particular [11]. Al aplicar este principio aleatoriamente, RED puede disminuir las velocidades de transmisión a medida que detecta que un enlace se acerca a un estado de congestión. Éste no es recomendable para situaciones donde el tráfico UDP es predominante [11].

Para enlaces donde se espera congestión de tráfico TCP/IP, es recomendable utilizar RED; éste no utiliza ninguna clasificación de tráfico o priorización relacionada con los descartes aleatorios, ya que descarta los paquetes indiscriminadamente a medida que detecta congestión inminente en el enlace. En cambio, WRED es una implementación de RED con diferentes perfiles de descarte, no se utiliza un mismo perfil para todas las colas, sino que, se pueden definir diferentes para colas individuales, además, múltiples perfiles se pueden definir dentro de una cola.

El método ECN se aplica también al tráfico TCP, éste se propuso en el RFC 2481[12] como una adición experimental a la arquitectura IP. Cuando se presenta un comienzo de congestión, se le comunica a los sistemas finales marcando el campo apropiado en las cabeceras TCP e IP, con indicadores de congestión en lugar de descartar el paquete.

[14] Negative Acknowledgement, se refiere a un ACK negativo.

2.2 ARQUITECTURAS DE QoS EN REDES IP

Los mecanismos mencionados en este capítulo como marcado y clasificación, junto a la planificación de paquetes, el encolamiento y la conformación de tráfico estudiados en el capítulo 1, usualmente se combinan en las redes, con el fin de proporcionar niveles de QoS extremo a extremo. En el caso de los enrutadores existen diferentes combinaciones de estos mecanismos, por ejemplo, los que se muestran en las Figuras 2.1, 2.2 y 2.3.

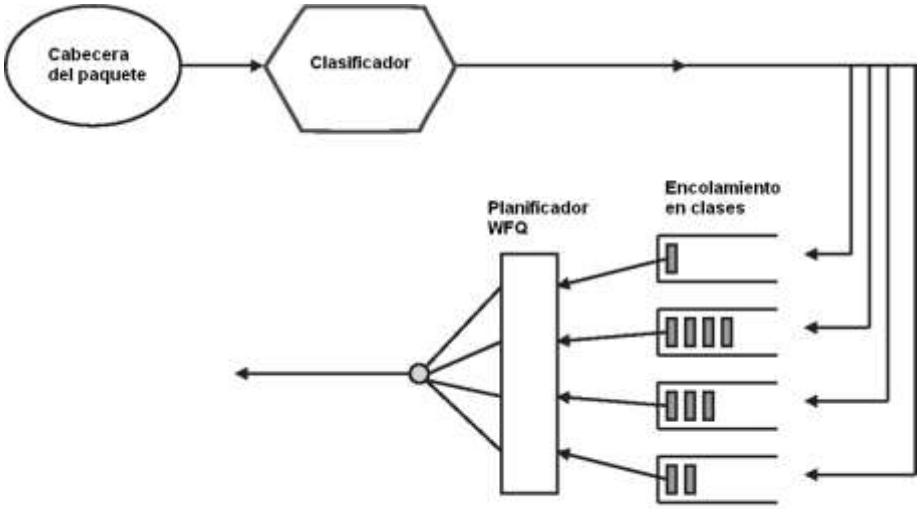

Figura 2.1 Procesamiento típico en enrutadores sin conformación de tráfico ni marcado

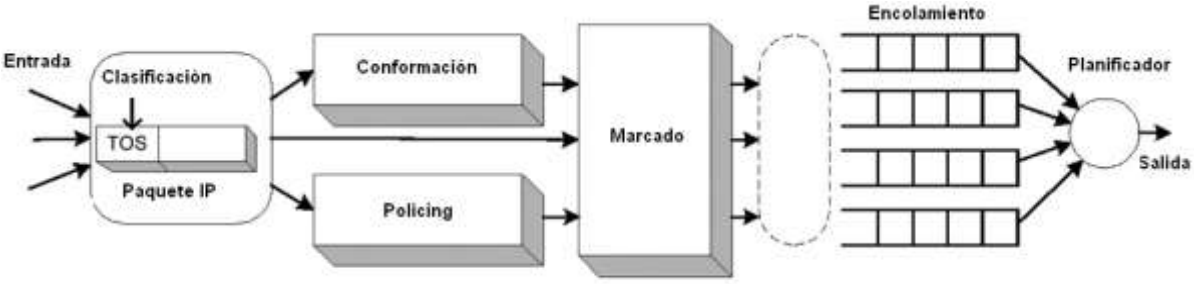

Figura 2.2 Procesamiento típico en enrutadores

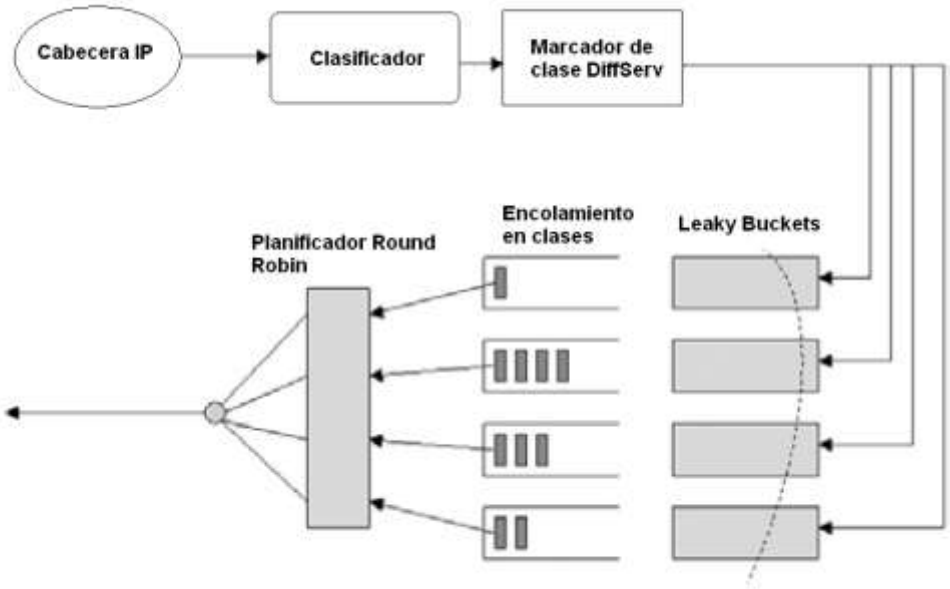

Figura 2.3 Procesamiento típico en enrutadores con conformación de tráfico

Estos arreglos de diferentes mecanismos en los enrutadores, no son ejemplos concretos de dispositivos de una tecnología u otra, se utilizan de manera ilustrativa para explicar configuraciones típicas. Como se observa en las figuras los paquetes son reconocidos y clasificados, no solamente por los campos de la cabecera IP, sino también por técnicas como las mencionadas en la sección 2.1.4. Los paquetes pueden ser marcados o remarcados con marcas especiales como las de DiffServ o las utilizadas en este proyecto que son independientes de la trama y solo se manejan en el control de tráfico en Linux. Se puede implementar un mecanismo de conformación para suavizar el tráfico entrante a cada cola, la cual puede o no utilizar clases (existen numerosos mecanismos de encolamiento, los estudiados en el capítulo 1 se consideran los más comunes y utilizados) y posteriormente un planificador asigna turnos a cada cola para enviar los paquetes por el enlace ó canal. La configuración específica y los mecanismos a utilizar, es una determinación de los proveedores de dispositivos, con base en los requerimientos de la tecnología, la información que ésta transporta y la necesidad de calidad de servicio, entre otros. Incluso, el administrador de red puede configurar manualmente el dispositivo para que se desempeñe de la manera más eficiente.

Existen formas de coordinar sistemáticamente los mecanismos de calidad de servicio utilizados en una red de acuerdo a una arquitectura que permita brindar QoS extremo a extremo para una variedad de tráfico. Estas arquitecturas organizadas son la de **Servicios Integrados** (IntServ) y la de **Servicios Diferenciados** (DiffServ).

La arquitectura IntServ utiliza un mecanismo explicito para señalizar requerimientos de QoS por flujo a las entidades de red como enrutadores y *hosts*. El protocolo RSVP, explicado en la sección 2.1.1.1 fue diseñado como un mecanismo dinámico para la reserva explicita de recursos en esta arquitectura (a pesar que, IntServ puede utilizar también otros mecanismos) [13].

En el contexto de IntServ, un flujo define una cadena de paquetes con la misma dirección fuente y de destino e igual número de puertos. IntServ requiere de un

protocolo que primero revise los recursos disponibles y posteriormente reserve una cantidad, dependiendo del flujo, en la red. RSVP se utiliza para comunicar la información disponible sobre la cantidad de recursos de red requeridos por una aplicación [13]. La figura 2.4 muestra los componentes utilizados en la arquitectura IntServ.

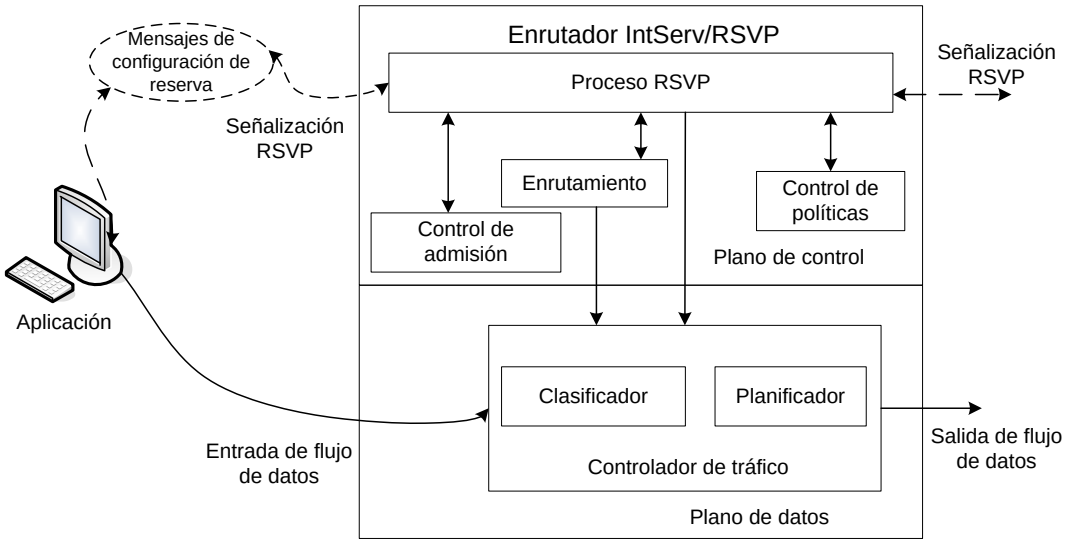

Figura 2.4 Componentes básicos de la arquitectura IntServ/RSVP

Los enrutadores IntServ/RSVP soportan conjuntamente los procesos de RSVP y de enrutamiento. El proceso de RSVP también necesita interactuar con el bloque controlador de tráfico (el cual consiste de los mecanismos de clasificación y planificación), el control de admisión (es aplicado a cada nodo y toma decisiones locales de aceptar o rechazar pedidos de reservas, basándose en una comparación de los recursos disponibles en el enrutador especifico y los recursos pedidos) y el controlador de políticas (el cual se encarga de hacer cumplir las políticas del administrador de red, sobre los recursos de ésta. Éste bloque determina quien es elegible para hacer reservas de QoS).

En la arquitectura de servicios diferenciados DiffServ, los enrutadores del núcleo de red procesan los paquetes basándose en el código de clasificación en las cabeceras de acuerdo a un comportamiento apropiado, denominado PHB (*Per Hop Behavior* o Comportamiento Por Salto) [8]. Todos los paquetes que llegan a una de éstas redes se marcan en el campo DSCP en la cabecera del paquete IP, como se mencionó en la sección 2.1.3, para indicar cual PHB debe ser aplicado al paquete. Los paquetes marcados iguales se consideran agregados de comportamiento (*behavior aggregate*) y todos reciben el mismo tratamiento PHB.

Para que un usuario pueda acceder a los servicios diferenciados de su proveedor de servicios de red, debe de tener un SLA (Service Level Agreement ó Acuerdo de Nivel de Servicios), el cual es un contrato que especifica la clasificación de los servicios soportados y el volumen de tráfico asignado para cada clase.

En esta arquitectura se definen tres PHB's:

1. BE (*Best Effort*): no hay garantías, la red hace solo lo que puede.
2. AF (*Assured Forwarding*): garantías más flexibles y fáciles de la red. Los paquetes AF tienen mayor prioridad
3. EF (*Expedited Forwarding*): los paquetes son procesado más rápido en la red

Los PHB`s se implementan en los enrutadores con la ayuda de mecanismos estudiados en el capítulo 1 de este documento como CBQ, WFQ, etc. de acuerdo al valor del campo DS en el paquete IP. Ésta arquitectura es escalable, pues la complejidad de implementación se desplaza del núcleo de la red hacia los enrutadores de frontera [13].

La arquitectura DiffServ se basa en un modelo simple donde el tráfico entrante a la red se clasifica y posiblemente se acondiciona en los extremos de la red y se asigna a un agregado de comportamiento. Los componentes de acondicionamiento de tráfico pueden clasificar, marcar, conformar y descartar paquetes, tan pronto entran o dejan un dominio DiffServ [13]. Un enrutador de ingreso realiza marcado y *policing* por flujo, lo que se muestra en la figura 2.5. A medida que los paquetes marcados avanzan, se combinan con paquetes marcados igual por DiffServ, en agregados de comportamiento. Para asegurarse que las características de los datos agregados sean conformes con las especificaciones de tráfico para el servicio requerido, cada dominio DiffServ debe realizar conformación (*shaping*), en cada agregado de tráfico a medida que deja el dominio.

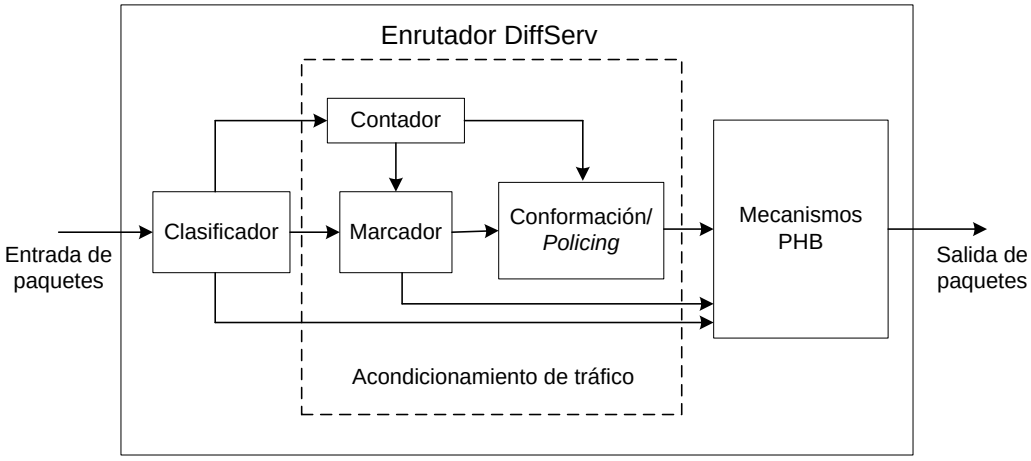

Figura 2.5 Arquitectura DiffServ

Los perfiles temporales (por ejemplo, tasa promedio y otras características) son escogidos por el proveedor de red DiffServ, con base en especificaciones de nivel de servicio (*Service-Level Specification, SLS*), las cuales se definen en un acuerdo de acondicionamiento de tráfico (*Traffic Conditioning Agreement, TCA*), dentro del cual los clientes reciben un servicio para su tráfico. Basándose en estos perfiles el contador determina si el tráfico de los clientes es conforme o no (por ejemplo, si esta dentro o fuera de perfil).

2.2.1 Marcador de tres colores de tasa única (*Single Rate Three Color Marke*r, srTCM)

En [14] se define srTCM, el cual puede ser utilizado como un componente del acondicionamiento de tráfico en DiffServ. srTCM mide un flujo de tráfico y marca los paquetes verde, amarillo o rojo. El marcado se basa en un **Velocidad de Información Acordada** (*Committed Information Rate, CIR*) y en dos tamaños de ráfaga (*burst size*): el Tamaño Acordado de Ráfaga (CBS) y el Tamaño de Exceso de Ráfaga (EBS). Un paquete se marca como verde si no excede la CBS, amarillo si excede la CBS, pero no la EBS y rojo para otros casos. srTCM puede ser útil, por ejemplo, para servicios de *policing* de ingreso de un servicio, donde solo la longitud, no la tasa pico, de la ráfaga determinen la elegibilidad del servicio [14]. El medidor analiza cada paquete y pasa el paquete y el resultado de la medida al marcador, como se observa en la figura 2.6

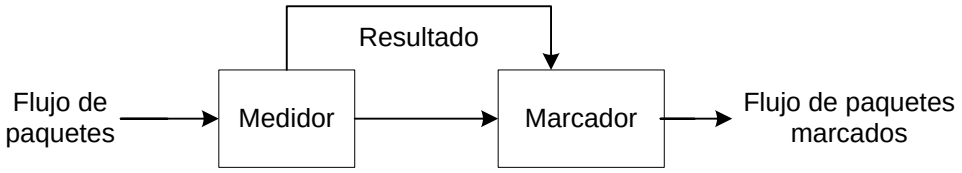

Figura 2.6 srTCM

El medidor opera en dos modos: invisibilidad de color, donde asume que el flujo no tiene colores y en identificación de colores, asumiendo que una entidad precedente ha asignado previamente colores al flujo de paquetes entrantes, de forma que cada paquete sea verde, amarillo o rojo. El marcado re-asigna color a un paquete IP de acuerdo al resultado del medidor. El color se codifica en el campo DS [14] de una forma específica PHP.

La parámetros CIR, CBS y EBS deben configurarse; para esto se mide la CIR en bytes de paquetes IP por segundo, la CBS y la EBS se miden en bytes (como mínimo una de las dos tiene que ser mayor que cero).

El comportamiento del contador se especifica en términos de su modo y de dos *token buckets*, C y E, los cuales comparten la tasa común CIR. El tamaño máximo del *token bucket* C es CBS y el máximo del *token bucket* E es EBS. Inicialmente tanto C como E están llenos, de ahí en adelante los contadores de *token* (*token count, Tc y Te*) se actualizan CIR veces por segundo de la siguiente forma:

- Si Tc es menor que CBS, Tc se incrementa en uno, o de otra forma
- si Te es menor que EBS, Te se incrementa en uno, de otra forma
- ni Tc ni Te se incrementan.

Cuando un paquete de tamaño B bytes llega en un tiempo t pasa lo siguiente, si el srTCM está configurado para operar en el modo de invisibilidad de color:

- Si $Tc(t)-B \geq 0$, el paquete es verde y el Tc se decrementa en B hasta el valor mínimo de 0, de otra forma
- si $Te(t)-B \geq 0$, el paquete es amarillo y Te se decrementa en B hasta el valor mínimo de 0, de otra forma
- el paquete es rojo y ni Te ni Tc se decrementan.

Cuando el srTCM está configurado para operar en el modo de identificación de colores:

- Si al paquete se le ha asignado previamente el color verde y Tc(t)-B≥0, el paquete es verde y Tc se decrementa en B hasta el valor mínimo de 0, de otra forma
- si al paquete se le ha asignado previamente el color verde o amarillo y si Te(t)-B≥0, el paquete es amarillo y Te se decrementa en B hasta el valor mínimo de 0, de otra forma
- el paquete es rojo y ni Tc ni Te se decrementan

De acuerdo a estas reglas, el marcado de un paquete con determinado color requiere que existan suficientes *tokens* de ese color para acomodar el paquete entero [14].

2.2.2 Marcador de tres colores de dos tasas (*Two Rate Three Color Marker*, trTCM)

En [15] se define trTCM, el cual también puede ser utilizado como un componente del acondicionamiento de tráfico en DiffServ. trTCM mide un flujo de tráfico y marca los paquetes verde, amarillo o rojo. El marcado se basa en dos tasas: la **Velocidad de Información Acordada** (*Committed Information Rate, CIR*) y la **Tasa Pico de información** (*Peak Information Rate, PIR*). Un paquete es marcado rojo si excede la PIR y amarillo o verde dependiendo si excede o no la CIR.

Al igual que en srTCM, el medidor analiza cada paquete y pasa el paquete y el resultado de la medida al marcador y puede operar en dos modos: invisibilidad de color e identificación de colores. El marcador re-asigna color a un paquete IP de acuerdo al resultado del medidor. El color se codifica en el campo DS [15] de una forma específica PHP.

Los parámetros CIR y PIR se miden en bytes de paquetes IP por segundo. Existen también dos tamaños de ráfaga asociados a cada uno de estos parámetros: PBS (*Peak Burst Size* o tamaño pico de ragafa) y CBS (*Commited Burst Size* o tamaño comprometido de ráfaga), medidos en bytes; por lo menos alguno de los dos debe ser mayor que cero.

Igualmente el comportamiento del contador se especifica en términos de su modo y de dos *token buckets*, P y C, con tasas PIR y CIR respectivamente. El tamaño máximo del *token bucket* P es PBS y el máximo del *token bucket* C es CBS. Inicialmente tanto P como C están llenos, de ahí en adelante el contador de *token* Tp se incrementa en un PIR veces por segundo hasta PBS y el contador Tc se incrementa en un CIR veces por segundo hasta CBS.

Cuando un paquete de tamaño B bytes llega en un tiempo t, pasa lo siguiente si el trTCM es configurado para operar en el modo de invisibilidad de color:

- Si Tp(t)-B<0, el paquete es rojo, de otra forma
- si Tc(t)-B<0, el paquete es amarillo y Tp se decrementa en B, de otra forma
- el paquete es verde y tanto Tp como Tc se decrementan en B.

Cuando el trTCM está configurado para operar en el modo de identificación de colores

- Si al paquete se le ha asignado previamente el color rojo o si Tp(t)-B<0, el paquete es rojo, de otra forma
- si al paquete se le ha asignado previamente el color amarillo o si Tc(t)-B<0, el paquete es amarillo y Tp se decrementa en B, de otra forma
- el paquete es verde y tanto Tp como Tc se decrementan en B.

2.3 CALIDAD DE SERVICIO EN REDES INÁLAMBRICAS

2.3.1 Red de Área Local Inalámbrica (WLAN) 802.11

El estándar IEEE 802.11 es parte de la familia de estándares 802, el cual maneja las redes de área local o LAN´s e incluye dentro de sus miembros uno de los estándares más utilizados actualmente, el 802.3, que define el control de acceso al medio (*Medium Access Control, MAC*) de Ethernet.

802.11 especifica la capa física inalámbrica y una capa MAC que trabaja con cualquiera de las físicas [1]. Como en 802.3, o en 802.5 (*Token Ring*), 802.11 esta diseñado para trabajar con un Control Lógico de Enlace (*Logical Link Control, LLC*) 802.2, como se muestra en la figura 2.7. El LLC y la MAC, son subcapas de la capa de enlace.

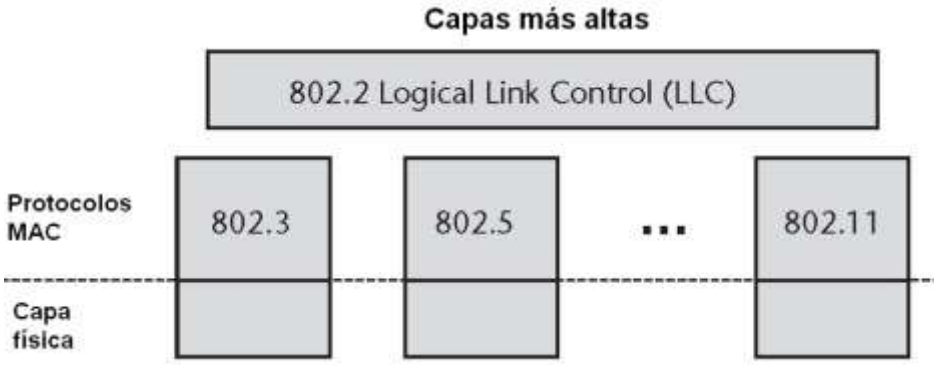

Figura 2.7 802.11 entre otros miembros de la familia IEEE 802

Los equipos en una WLAN 802.11 se conocen como estaciones. Un grupo de estaciones que se puede comunicar directamente entre ellas (por que están en una proximidad lo suficientemente cercana) se conoce como: **Conjunto de Servicio Básico** (*Basic Service Set, BSS*). El BSS es el bloque básico de las WLAN 802.11. Como se puede observar en la figura 2.8, éste tipo de redes pueden implementarse en dos configuraciones básicas: modo ad hoc y modo infraestructura. En el modo ad hoc, el BSS es un BSS independiente (IBSS), mientras que en modo infraestructura, se combinan dos o más BSS´s en un **Conjunto de Servicio Extendido** (*Extended Service Set, ESS*) que incluye un sistema de distribución (DS), el cual distribuye los paquetes entre todos los BSS de un ESS. Cada uno de los BSS´s en el ESS accede al DS a través de un punto de acceso (AP), cada AP es una estación con capacidades mejoradas. En muchos casos, un elemento funcional conocido como portal actúa como *gateway* hacia redes externas.

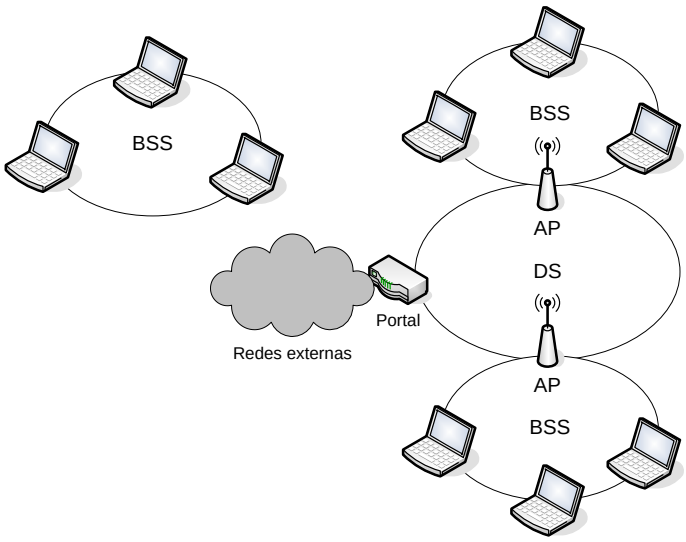

Figura 2.8 Modo adhoc e infraestructura en 802.11.

Originalmente el estándar definía tres posibilidades para la capa física de 802.11 [1]: infrarrojo (IR), DSSS (*Direct Sequence Spread Spectrum*) y una basada en FHSS (*Frequency Hopping Spread Spectrum*). La extensión b del estándar soporta DSSS en la banda de 2.4Ghz con tasas de 1, 2, 5.5 y 11 Mbps, utilizando CCK (Complementary Code Keying) para alcanzar las dos tasas de transmisión más rápidas. Las extensiones 802.11a y g en las bandas de 5Ghz y 2.4Ghz, respectivamente, utilizan OFDM y pueden alcanzar velocidades en el rango de los 6 a los 54Mbps.

La ventaja de las redes 802.11 es el amplio ancho de banda disponible, el cual es cada vez más comparable con el de las LAN´s cableadas; pero éstas aun presentan algunos inconvenientes, entre los que se encuentra una limitante importante: no tienen señalización o un plan de control para asegurar los niveles de servicio requeridos por los usuarios [8], mientras que las redes móviles celulares, por ejemplo, tienen un sistema de señalización y de control confiable y eficiente, para brindar y mantener un nivel de servicio requerido por los usuarios (el problema de esta redes es el bajo *throughput q*ue pueden ofrecer). Mientras se espera una integración eficiente de estas tecnologías, se requiere de mejoras en el medio inalámbrico y la diferenciación de servicios en redes 802.11, donde se deben desarrollar mecanismos para asegurar a los usuarios QoS, seguridad y movilidad. Se han formado múltiples grupos de trabajo en la organización de estandarización de la IEEE para estos inconvenientes, el grupo ”f” trabajó en los problemas de movilidad, el “i” en temas de seguridad de datos, el “e” en calidad de servicio, entre otros.

2.3.2 Calidad de Servicio en 802.11

En el estándar inicial o básico 802.11, se propusieron dos métodos de acceso al medio: la **Función de Coordinación Distribuida** (*Distributed Coordination Function, DCF*), la cual es muy similar al método utilizado en Ethernet y ofrece un acceso al medio implementando un esquema de acceso aleatorio basado en CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance* o Acceso Múltiple por Detección de Portadora con Prevención de Colisiones) y el otro es la **Función de Coordinación en Punto** (*Point Coordination Function, PCF*) en la que las diferentes transmisiones de datos, entre estaciones de red, son controladas por un punto central de coordinación

(*Point Coordinator, PC*). Este método de acceso está ubicado, generalmente, en el AP y ha sido diseñado para permitir la transmisión de datos sensibles al retardo (por ejemplo, para brindar un nivel de QoS a estaciones que lo requieran). PCF no ha sido ni integrado en dispositivos comerciales ni implementado hasta el momento [8], ya que este tiene varias desventajas entre las que se encuentran:

- La imposibilidad de reconocer retardos entre tramas de control.
- La imposibilidad de reconocer la duración de la transmisión de estaciones autorizadas para transmitir.

El protocolo MAC de 802.11 no tiene forma de diferenciar fuentes o flujos de tráfico. Todo el tráfico es tratado igual [13] (por ejemplo, la misma prioridad para acceder al medio) en el mecanismos DCF, esto significa que no se pueden tener consideraciones especiales para el tráfico en el canal con requerimientos críticos de ancho de banda, retardo, *jitter* y perdida de paquetes. Por ejemplo, un tráfico de baja prioridad puede dejar sin los recursos necesarios una transmisión de video y degradar la experiencia del usuario.

Ante este tipo de inconvenientes la IEEE generó el estándar 802.11e con el cual las estaciones que operan como coordinador central para otras estaciones en el mismo BSS con soporte de QoS se denominan coordinadores hibridos (*Hybrid Coordinator, HC*), similar que en el caso de los PC, los HC residen dentro de un AP (802.11e en éste caso). Las mejoras para incorporar calidad de servicio incluyen una función de coordinación nueva denominada: **Función de Coordinación Hibrida** (*Hybrid Coordination Function, HCF*), la cual solo se puede utilizar en configuración de red QoS (QBSS). HCF combina funciones del DCF y el PCF con mejoras basadas en mecanismos de QoS y subtipos de trama, para permitir utilizar un conjunto de secuencias de intercambio de trama uniforme. Esta función hibrida utiliza un mecanismo de acceso al canal basado en disputas (*Enhanced Distributed Channel Access, EDCA*) y un mecanismo de acceso al canal controlado (*HCF Controlled Channel Access, HCCA*), para transferencias sin disputas. El HCF definido en 802.11e permite hasta ocho prioridades de clases de tráfico las cuales son mapeadas directamente al DSCP.

EDCA define el mecanismo de categoría de acceso (AC) que soporta las prioridades en las estaciones. Cada estación puede tener hasta cuatro AC´s para permitir las ocho prioridades de usuario (UP´s). Una o más UP´s se asignan a un AC. Una estación accede al medio de acuerdo con el AC de la trama que va a ser transmitida, el mapeado desde prioridades hacia AC´s se define en la tabla 2.4. Cada AC es una variante mejorada del DCF, la cual participa por oportunidades de transmisión (*TXOP´s*: es un intervalo de tiempo en el cual una estación en particular tiene el derecho de iniciar transmisiones) utilizando un conjunto de parámetros de EDCA, un AC con mayor prioridad tiene asegurado transmitir primero que prioridades menores.

Tabla 2.4 Mapeo de prioridad a categoría de acceso

Prioridad de usuario (igual que en 802.1d)	**AC**	**Designación**
1	0	*Background*[15]
2	0	*Background*
0	0	*Best Effort*
3	1	*Best Effort*
4	2	Video
5	2	Video
6	3	Voz
7	3	Voz

HCCA fue diseñado para utilizar un coordinador centralizado (HC) apto para QoS y operar bajo reglas diferentes a las del PC del PCF convencional. El HC se encuentra dentro de un AP de QoS (QAP) del QBSS y utiliza la prioridad más alta de acceso al medio inalámbrico para iniciar secuencias de intercambio de tramas y asignar TXOP´s para si mismo y otras estaciones aptas para QoS (QSTA´s), hasta proporcionar fases de acceso controlado (CAP) de duración limitada para transferencias de datos de QoS sin disputas. Pero en mayo de 2006, directivos de la alianza Wi-Fi decidieron acabar con el grupo de trabajo WMM-SA (HCCA), dejando a WMM (EDCA) como el único esquema para QoS con un plan de certificación.

2.3.3 Impacto de la movilidad en la calidad de servicio

Dentro de los inconvenientes de 802.11 para poder ofrecer niveles de calidad de servicio se encuentran los mencionados en la sección 1.2.8: Planificación de paquetes en redes inalámbricas, las características cambiantes del medio inalámbrico y la movilidad, que es la ventaja más importante de las WLAN, permitir a la estaciones moverse con libertad entre locaciones, ya sea dentro de una misma WLAN o en diferentes segmentos WLAN.

Para propósitos de compatibilidad, la MAC 802.11 debe parecer ante las capas superiores de la red como una LAN 802 estándar [16], forzando a la capa MAC a manejar la movilidad de las estaciones de una manera transparente para los niveles de la pila de protocolos LAN 802.

Cuando se habla de portabilidad, se refiere a la ganancia en productividad de usuarios que puede acceder a recursos de información donde les sea conveniente, pero se quitan solo barreras físicas de conectividad y no se cambia el proceso de conexión a una red en cada nueva locación. Sigue siendo necesario conectarse físicamente a la red y reestablecer las conexiones de red, las cuales no pueden utilizarse cuando el dispositivo se está moviendo.

[15] Se refiera a tráfico de baja prioridad, que no tiene requerimientos extrictos de latencia y throughput.

La movilidad remueve barreras más lejanas, la mayoría de las cuales se basan en la arquitectura lógica de la red. Las conexiones de red permanecen activas aun cuando el dispositivo se está moviendo. Esto es crítico para tareas que requieren de conexiones largas y persistentes, por ejemplo, aplicaciones de bases de datos. Con 802.11 se tiene movilidad en el nivel de enlace, lo que no se tiene en IP [16]. Los *hosts* 802.11 pueden moverse libremente dentro de la ultima conexión de red, pero IP, de la forma como se está desarrollando actualmente, no brinda lo posibilidad de moverse a través de las barreras de subred. Para los requerimientos de movilidad, es importante que la dirección IP no cambie al conectarse a cualquier AP. Nuevas direcciones IP interrumpen conexiones activas.

Cuando un enlace inalámbrico se compara con uno cableado, se encuentras dos desventajas importantes: primero la pobre calidad (comparada) que resulta en errores de tasa bit (BER) más altos, implicando que los mecanismos de QoS en Internet móvil deben manejar tasas de pérdida de paquetes más altas. El segundo problema, es la variación de la calidad en el enlace, la cual puede cambiar por varias razones. El clima, interferencias con otros usuarios móviles y barreras como edificios, puentes y montañas puede degradar temporalmente la calidad del canal inalámbrico. La distancia entre el usuario móvil y la estación base o el AP, juega un papel importante en la intensidad de la señal recibida y la calidad. Lo más complicado de estas variaciones en la calidad es que suceden aleatoriamente, aunque algunas se pueden predecir o modelar estadísticamente, la mayoría permanece impredecible. Los mecanismos de QoS deben considerar estos cambios impredecibles en la calidad de la línea durante una sesión de comunicación, consideraciones que no se aplican en el control de tráfico de este proyecto, pues el escenario inalámbrico específico de aplicación involucra estaciones fijas en enlaces empresariales o enlaces de WISP entre el proveedor y un usuario fijo; por esto también la importancia de aclarar el concepto de movilidad y cómo aplica la QoS en redes 802.11.

La movilidad del usuario tiene un impacto directo en la calidad de servicio. Si el usuario tiene libertad de movimiento, la ruta de red de origen a destino puede variar durante una sesión de comunicación. Dado que los recursos de red en una ruta deben ser reservados con anticipación para soportar niveles de calidad de servicio, la tarea de reserva de recursos se vuelve extremadamente complicada si la ruta cambia frecuente e impredeciblemente. El inconveniente del *hand-off* entre estaciones base, es muy problemático, en especial, para aplicaciones multimedia en tiempo real, ya que unos segundos de interrupción puede tener un efecto destructivo en la calidad de servicio de la comunicación.

RESUMEN

En este capítulo se presenta información determinante en el desarrollo de este proyecto; diferentes mecanismos de QoS como clasificación y marcado de paquetes, explicados en éste, junto a la conformación y la planificación estudiados en el primer capítulo, pueden ser utilizados en conjunto para alcanzar niveles de calidad del servicio y deben considerarse en aplicaciones donde se busca mejorar inconvenientes inherentes de las tecnologías de red, clasificar el tráfico y optimizar la utilización de los recursos. Existen en la actualidad arquitecturas como IntServ y DiffServ las cuales incorporan estos mecanismos de una forma coordinada y eficiente.

Este proyecto está enfocado hacia el control de tráfico en la tecnología inalámbrica 802.11, éste capítulo la presenta, junto a los avances que se tienen en calidad de servicio con el estándar 802.11e. El utilizar el medio inalámbrico como medio de transmisión implica considerar factores, que no se presentan en redes cableadas, los cuales afectan directamente la calidad de servicio. Se presentó el concepto de movilidad y las implicaciones directas con el tratamiento de la información; este proyecto no involucra estaciones móviles y los mecanismos utilizados para el control de tráfico no tienen consideraciones especiales ante aspectos como el estado del canal, el *hand-off*, la posición de la estación, entre otros.

3. CONTROL DE TRÁFICO EN LINUX Y DESCRIPCIÓN DEL TRÁFICO IP

El tráfico IP es fundamentalmente diferente comparado con el tráfico telefónico clásico [1]; por esto, en aplicaciones que lo involucran se debe caracterizar y posteriormente clasificar.

Actualmente existen medidas útiles del tráfico IP en redes cableadas; con la introducción de Internet inalámbrico y móvil, los servicios ofrecidos en Internet no cambian dramáticamente de un momento a otro, obviamente muchos de los sitios de Internet que se enfocan hacia usuarios móviles deben ofrecer servicios adaptados para las pantallas de los terminales móviles. El uso masivo y los bajos precios actuales de los dispositivos móviles como teléfonos celulares, *PDA's* y computadores portátiles, presumen la utilización de servicios similares a los ofrecidos en redes cableadas, en redes inalámbricas y redes móviles.

El primer paso hacia el análisis y diseño de redes IP inalámbricas es especificar claramente la descripción del tráfico de red y los requisitos para el funcionamiento de las aplicaciones; una descripción completa y definitiva puede ser inadecuada por varias razones, entre ellas: diseñar una red futura con base en información del tráfico de aplicaciones actuales, pues existe la posibilidad de que se desarrollen nuevas aplicaciones radicalmente diferentes a las actuales. Para este proyecto, se asume que las aplicaciones tienen ciertas propiedades inherentes, que resultan del comportamiento de los usuarios y su interacción, las cuales son independientes de la infraestructura de red y no se espera que cambien notablemente en el futuro.

En Internet y en general en cualquier tipo de red existen servicios heterogéneos [1] con diferentes demandas de tráfico y de calidad de servicio. Algunos servicios requieren comunicación en tiempo real, otros no. Además existe heterogeneidad en la necesidad de ancho de banda, en los flujos de tráfico generados por diferentes aplicaciones. Por esto, se debe determinar la naturaleza del tráfico y considerar sus características.

3.1 TRÁFICO DE TIEMPO REAL

Las aplicaciones de tiempo real, son las aplicaciones en las cuales tiene particular importancia el tiempo y el orden en la entrega de los paquetes. Los paquetes deben alcanzar su destino dentro de un periodo de tiempo determinado y dentro de límites variables [2]. Los ejemplos más comunes incluyen aplicaciones como audio y video.

La característica fundamental de este tipo de tráfico, que puede ser de vital importancia en enlaces como los considerados en este proyecto, es la sensibilidad de estas aplicaciones ante los retardos y el *jitter*. El grado de la misma, depende del tipo específico de aplicación. Este tipo de tráfico se puede clasificar básicamente en dos categorías: tráfico interactivo (intolerante) y no interactivo (tolerante) [3]; las diferencias entre estas categorías radica en como las aplicaciones reaccionan ante retardos y *jitter*.

3.1.1 Tráfico Interactivo (intolerante)

Aplicaciones de conversaciones de dos vías, como en el caso de una video-conferencia, son interactivas. En éstas, señales de audio/video son empaquetadas en el origen y transportadas a través de la red, hasta el receptor. Tanto el orden temporal, como la latencia extremo a extremo de los paquetes, tienen un impacto fundamental en la calidad de servicio experimentada por los dos usuarios finales.

Estas aplicaciones son intolerantes (o con un muy bajo nivel de tolerancia) al *jitter* y el retardo de paquetes [2]; si estos fenómenos ocurren por encima de ciertos limites, la señal se puede degradar hasta el punto de alcanzar una calidad inaceptable [4].

3.1.2 Tráfico no Interactivo (tolerante)

Aplicaciones de tiempo real, como la distribución de audio/video, que son básicamente transferencias de información de una vía y que contrario a las aplicaciones interactivas, para los usuarios no es tan determinante escuchar o ver los eventos algunos milisegundos (o incluso segundos) después que estos ocurren. Estas aplicaciones son más tolerantes ante los retardos y el *jitter* [4].

Los requerimientos para la latencia total y el *jitter*, de estas aplicaciones, son menos exigentes, que en las aplicaciones intolerantes [4]. Ciertos niveles de latencia y *jitter*, que pueden ser inaceptables en señales interactivas de audio, pueden ser aceptados en distribuciones de audio no interactivas.

3.1.3 Soporte de Tráfico de Tiempo Real

Este tipo de aplicaciones, como se ha mencionado a lo largo de este documento, requieren de control sobre la calidad de servicio que reciben de la red. Para soportar los diferentes requerimientos ante los retardos, el *jitter* y el ancho de banda, las redes pueden utilizar parámetros de QoS y aceptar el tráfico de una aplicación dándole un nivel de prioridad más alto, comparado con otro tipo de tráfico en la red.

Como es de esperarse, existen muchos acercamientos hacia la forma de brindar calidad de servicio en aplicaciones de tiempo real, esto se muestra en la figura 3.1.

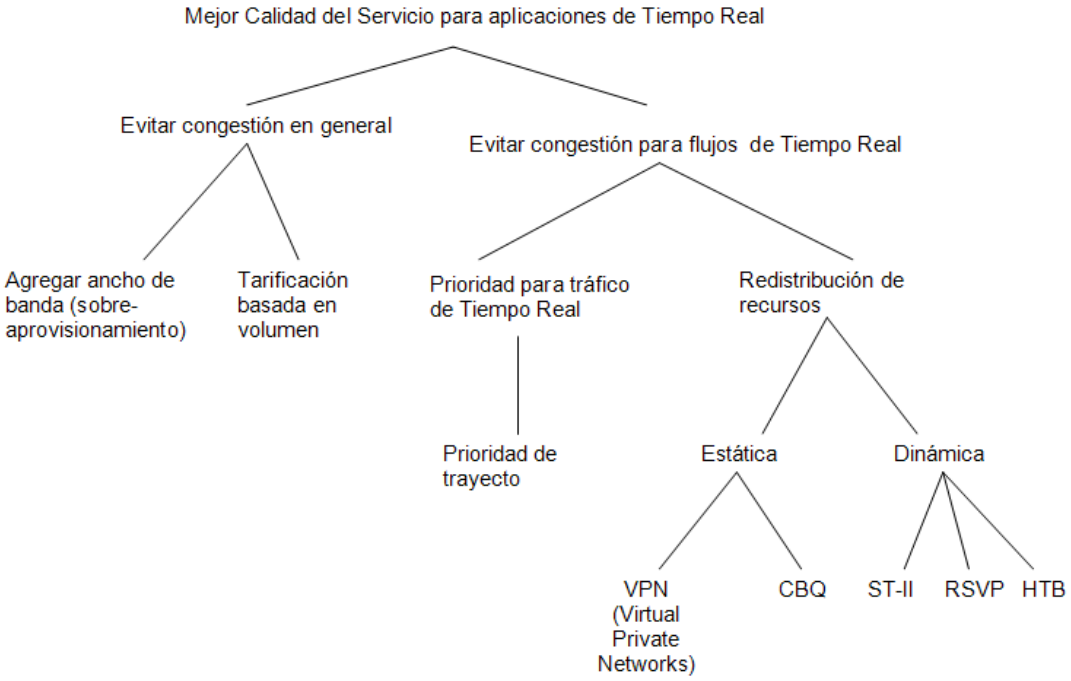

Figura 3.1 Estrategias para alcanzar QoS

Inicialmente, se puede pensar en evitar la congestión, mediante un sobre-aprovisionamiento de la red (por ejemplo, brindando más ancho de banda, del que pueda ser necesitado). Esta solución puede funcionar por un tiempo, pero reduce la escalabilidad de la red, ya que, es muy probable que eventualmente, el incremento en el número de usuarios y de aplicaciones que estos utilicen, así como la información transmitida, requiera del ancho de banda disponible. También puede ser muy costosa.

Otro enfoque, para prevenir la congestión es, cobrar a los usuarios con base en el volumen de su tráfico de red; esto en el caso de enlaces de empresas proveedoras de servicios de Internet (ISP). Normalmente en Internet y en redes corporativas, cuando los servicios son gratis (se paga una tarifa fija por conexión) los servicios se sobre utilizan [5]; al cobrar por volumen de tráfico, se reduciría la utilización de los recursos. Esta alternativa no es muy aceptada [5] dado que los servicios tarificados en Internet no son muy populares y para inconvenientes específicos, como en los enlaces inalámbricos tratados en este proyecto, no es una alternativa viable.

La perspectiva utilizada en el desarrollo de este proyecto, para alcanzar niveles de calidad de servicio en enlaces 802.11, no es evitar completamente la congestión, sino tratar de evitar la congestión para el tráfico de tiempo real y los servicios que se consideran más demandantes o prioritarios. Esto se puede alcanzar, ya sea, brindando más prioridad al tráfico de tiempo real o redistribuyendo los recursos entre el tráfico considerado de mayor prioridad y el de menor prioridad.

Existen varias formas de diferenciar entre tipos de servicio, una de ellas basándose en el campo de precedencia dentro de la cabecera de cada paquete, en el campo Tipo de Servicio (*Type of Service*, TOS). Con este método, paquetes con precedencia de alta prioridad, son atendidos mejor que paquetes de precedencia de menor prioridad. Otra forma para diferenciar entre tipos de servicio es identificar el tráfico de acuerdo al

puerto destino de la comunicación, entre otras. También, posteriormente en éste capítulo, se menciona el funcionamiento de los filtros, los cuales son muy utilizados para identificar la precedencia del tráfico; aplicaciones como el control de tráfico en Linux, tienen la capacidad de implementar filtrajes acertados y detallados.

Si se opta por coordinar la asignación de recursos en la red, entre servicios prioritarios y no prioritarios (o entre servicios de tiempo real y no tiempo real), entonces, se pueden tener dos opciones: asignar los recursos estáticamente, donde se combinan las ventajas de compartir recursos, con las ventajas de líneas dedicadas, pero con la pérdida de flexibilidad, ya que, sino se utiliza completamente el ancho de banda disponible, este no se puede utilizar para otro tráfico; o como en el caso de este proyecto asignar los recursos dinámicamente.

3.2 TRÁFICO DE NO TIEMPO REAL

El tráfico de no tiempo real, o elástico, incluye aplicaciones las cuales esperan por la llegada de los paquetes antes de procesar la información [4]. En contraste a las aplicaciones de tiempo real, las cuales descartan paquetes retardados y que no retransmiten paquetes perdidos, las aplicaciones elásticas retransmiten paquetes perdidos o con retrasos [4], hasta que estos lleguen a su destino.

Dentro de estas aplicaciones se tiene, por ejemplo, servicios de Internet como correo electrónico, FTP y exploración de sitios WEB. Este tipo de servicios se ve menos afectado ante retardos y *jitter* [2]; estos procesan los paquetes inmediatamente después de su llegada.

La diferencia fundamental entre estos dos tipos de aplicaciones, es básicamente, la forma de controlar los flujos y de corregir los errores. En las aplicaciones de tiempo real, el flujo se controla por las características de la aplicación que lo envía [2], ya que ésta determina la tasa a la cual son enviados los paquetes; después, la red intenta minimizar la latencia y la pérdida de paquetes, basándose en condiciones impuestas por la aplicación fuente de la información. Por otra parte, las aplicaciones elásticas utilizan algoritmos adaptativos para el control de flujos [4], en los cuales, la tasa de flujo de paquetes, se basa en la visibilidad de la aplicación origen de la información, sobre la capacidad disponible en la red y la disponibilidad de espacio en el extremo receptor, en ese preciso momento.

En cuanto al manejo de los errores, una aplicación de tiempo real tiene un factor temporal asociado; los paquetes deben ser entregados, dentro de un periodo de tiempo determinado y dentro de una secuencia especificada por la aplicación, de lo contrario, los paquetes pasan a ser irrelevantes y pueden ser descartados. Paquetes errados, fuera de secuencia y con latencia mayor de los requerimientos de la aplicación, deben ser descartados. Una aplicación no tiempo real, utiliza técnicas de detección de errores y de retransmisión, para reponerse de errores de transmisión.

3.3 MÉTRICAS DE CALIDAD DE SERVICIO

Para entender las métricas de Calidad de Servicio y los inconvenientes de red que afectan la percepción del usuario de Calidad de Servicio, es importante tratar los componentes básicos de un Sistema de Comunicaciones Digital (*Digital Communications System, DCS*). Para transportar señales analógicas como voz y video, sobre redes de paquetes en este tipo de sistemas, éstas deben seguir el siguiente proceso, en el destino inicial y final [6]:

- Codificación/Decodificación.
- Paquetización/Despaquetización.
- Codificación de Canal/Decodificación.
- Modulación/Demodulación.
- Conmutación.
- Señalización.

Cada uno de estos procesos transforma de una forma u otra la señal digital, así como también, agrega factores que pueden generar errores en la transmisión [1]; para este documento y para el proyecto en general, la explicación de cada uno de estos procesos es irrelevante, pues, son conceptos tratados previamente en otros documentos, trabajos de grado y bibliografía de clase.

3.3.1 Qos en Servicios de Tiempo Real

Cada nivel de la pila de protocolos de comunicación introduce inconvenientes [7]. Los factores que afectan la calidad de servicio en aplicaciones de tiempo real en las redes de paquetes incluyen:

- Ruido de Cuantificación.
- Tasa de Bit errado (*Bit Error Rate*).
- Retardos.
- Variación de retardos o "*Jitter*".
- Pérdida de paquetes.
- Selección del *Codec.*
- Control de eco.
- El diseño de la red.

3.3.1.1 Ruido de Cuantificación

Los errores de cuantificación se manifiestan como ruido en la señal analógica reconstruida. Estos son uno de los impedimentos básicos en los sistemas de comunicaciones digitales y además son inevitables.

Existe una relación entre el ancho de banda del canal y los errores de cuantificación: al utilizar más bits para representar las amplitudes muestreadas, menor el ruido de cuantificación producido, pero también, al utilizar más bits es más alta la tasa de transmisión y más alto el ancho de banda requerido.

3.3.1.2 Retardos

Se refiere a los retrasos de tiempo inducidos a la señal original. Estos afectan principalmente aplicaciones de tiempo real como la voz y el video [1] [7].

El retardo de una vía indica la cantidad de tiempo medido desde el momento en que se genera la señal hasta el momento en que es recibida, el de dos vías, es la suma de los dos tiempos de una vía. Las principales fuentes de retardos incluyen:

- Codificación fuente
    - Retardos en la conversión Analógica/Digital o viceversa.
    - Retardo de Trama.
- Retardo de paquetización.
- Codificación de canal
    - Detección de errores y corrección (es casi instantáneo y puede ser despreciable).
    - Intercalamiento de bits (*Bit Interleaving*).
- *Jitter.*
- Retardo de encolamiento de paquetes.
- Retardo de propagación.
- Retardo de transmisión o serialización.
- Retardo de *dejitter.*

○ Retardo de Trama

Dado que las muestras de habla se procesan trama a trama, la duración de la trama es un componente clave, inducido por la fuente de codificación. Actualmente existen diferentes duraciones de trama, dependiendo del mecanismo de codificación utilizado (PCM, ADPCM, etc.).

○ Retardo de Paquetización

Este retardo incluye el tiempo empleado en formatear los bits en paquetes. El principal retardo proviene de llenar la carga (*payload*) del paquete. En [5] se calcula 5.88mSg como el retardo de empaquetar voz en redes ATM, asumiendo que el principal retardo de paquetización está en llenar la carga, para cargas de igual tamaño, la paquetización de *VoIP* tomaría el mismo tiempo que para *VoATM*. La despaquetización en el extremo receptor necesita el mismo tiempo [6], por lo tanto el retardo total para paquetización y despaquetización es aproximadamente 11.8msg.

○ Retardo de Intercalamiento

El retardo de intercalamiento de bits (*Bit Interleaving*) es directamente proporcional al número de bits que sean organizados [6]. Existe un proceso inverso en el extremo receptor, por lo cual el retardo total es la suma de los dos procesos.

○ Probabilidad de Pérdida de Paquete

La explicación de este posible impedimento requiere de cálculos matemáticos, cuya explicación detallada no es trascendental en este proyecto, pero es importante

mencionar la naturaleza de este problema. La ley de *Poisson,* para números muy grandes, permite interpretar la probabilidad de pérdida de paquete como un valor, equivalente a la tasa entre el número de paquetes perdidos y el total de paquetes transmitidos.

Para servicios de tiempo real, la pérdida de paquetes se manifiesta como ruido en la señal decodificada, lo cual resulta en mala calidad, por ejemplo, voz entrecortada, saltos, degradación de la calidad del video, entre otros.

Las siguientes, son las principales fuentes de la pérdida de paquetes:

- Errores de *bit* debido a impedimentos de la línea de transmisión.
- Colisión de paquetes en el nivel de enlace.
- Errores de procesamiento en el nivel de red.
- Sobrecarga de los *buffers* del nivel de red.
- Descarte aleatorio de paquetes en el nivel de red.

En [6] se demuestra que al aumentar el tamaño de los *buffers* la probabilidad de pérdida de paquetes se reduce; pero esto aumenta el retardo, afectando los servicios de tiempo real.

3.3.2 Jitter o Variación de Retardo

Los paquetes de una conexión particular entre origen y destino, transportados por la red, son enrutados dinámicamente y no necesariamente atraviesan los mismos nodos de interconexión [4], esto significa que dos paquetes cualquiera, derivados del mismo bloque de información, pueden viajar por trayectos completamente distintos, en cuanto al número de nodos y la distancia física atravesada, lo que genera retardos diferentes en los tiempos de llegada de los paquetes; a esto es lo que se conoce como *Jitter*.

Este fenómeno puede introducirse de tres formas diferentes [7]:

- Variaciones en tiempos de transmisión, debido a las diferencias en las distancias físicas atravesadas.

- Variaciones en el tiempo de procesamiento, debido a diferencias en el número de nodos visitados.

- Variaciones en el tiempo de procesamiento en nodos en particular, debido a las diferentes longitudes de colas de paquetes esperando enrutamiento y transmisión en cada nodo.

Este problema afecta notablemente los servicios de tiempo real [4], se puede eliminar con un *buffer* en el dispositivo receptor, pero si el *Jitter* sobrepasa el tamaño del *buffer*, el buffer se desborda y genera pérdida de paquetes.

3.4 DESCRIPCIÓN DE LOS DIFERENTES TIPOS DE TRÁFICO

El tráfico de las redes actuales y de Internet, basado en IP, incluye varios servicios multimedia los cuales tienen diferentes características, considerados sus parámetros de tráfico: tasa de *bit*, *burstiness*[16], tiempo de duración de la conexión (*connection duration time*), entre otros, así como, demandas ante diferentes métricas de rendimiento como pérdida de paquetes, retardos y *throughput*.

Tabla 3.1 Características del tráfico de aplicaciones en Internet

Aplicación	Características del tráfico	Requerimientos de QoS
Correo electrónico (SMTP) *Transferencia de archivos (FTP)* *Telnet*	Archivos pequeños y transferidos por tandas	Retardo: Muy tolerantes Ancho de banda: Bajo Tipo de servicio: Best Effort
Exploración de páginas Web (HTTP)	Series de archivos pequeños y transmisiones en ráfaga	Retardo: Moderado Ancho de banda: Variable Tipo de servicio: Best Effort
Comercio electrónico	Transmisión de dos vías de varios archivos pequeños	Retardo: Sensible Ancho de banda: Moderado Pérdida de paquetes: Muy sensible Tipo de servicio: Debe ser confiable
VoIP *Audio en tiempo real*	Tasa de bit constante o variable	Retardo: Altamente sensible Ancho de banda: Bajo/moderado Pérdida de paquetes: Baja y predecible Tipo de servicio: Alta prioridad
Video	Tráfico *streaming* con tasa de bit variable	Retardo: Altamente sensible *Jitter*: Altamente sensible Ancho de banda: Alto Pérdida de paquetes: Baja y predecible Tipo de servicio: Alta prioridad

3.4.1 Tráfico agregado de Internet

El tipo de servicio dominante en el Internet actual es el servicio *best effort*, lo que significa que se comparte todo ancho de banda entre todos los flujos de tráfico. Todo el control de transporte se transfiere a los nodos finales del tramo de comunicación. Existen nodos de red que brindan servicios particulares para usuarios de Internet, los

[16]*Burstiness* es una medida de la naturaleza del tráfico. Una fuente de información que envía el tráfico utilizando tasas de transmisión diferentes en tiempos diferentes, se conoce como *Bursty*, mientras que una fuente que siempre envía la información a la misma tasa de transmisión se denomina no *Bursty [14]*, *Bursitiness*= Tasa pico/Tasa promedio.

servidores, como también aplicaciones corriendo en los terminales de usuario (computadores personales o terminales móviles) las cuales demandan servicios.

Los nodos de red y los terminales segmentan la información a transmitir en paquetes, después se agregan cabeceras a la información contenida en ellos, para incluir información de direccionamiento y de control, esto se hace desde la capa de aplicación hacia el medio físico; en el extremo contrario se extraen las cabeceras y la información de control para entregar la información correcta a la aplicación deseada.

Varias aplicaciones utilizan diferentes protocolos de transporte dependiendo de la demanda de su respectivo tráfico (por ejemplo TCP y UDP), como se sabe, estos protocolos usan *sockets* para comunicarse con la capa de aplicación [1]; en Internet, entre los protocolos del nivel de transporte y los protocolos del nivel enlace (como SONET y ATM) se tiene el protocolo IP, por esto se referencia el tráfico agregado de Internet como tráfico IP o tráfico Internet.

En [8] los autores reportan medidas en un *backbone* comercial de Internet sobre dos rangos: 24 horas y 7 días. Las medidas capturan tráfico agregado de Internet, como tráfico por protocolo. El reporte muestra que: el tráfico WEB[17] domina, como la aplicación de Internet más utilizada. El protocolo TCP es responsable por la mayoría del tráfico: 95% o más de los bytes, de 85% a 95% de los paquetes y de 75% a 85% de los flujos. La mayoría del tráfico TCP es usualmente tráfico WEB, con aplicaciones cliente-servidor como responsables por más de la mitad de los bytes (65-80%) y de los paquetes (55-75%).

Antes del *boom* del WEB, la mayor parte del tráfico TCP se generaba por aplicaciones de transferencia de archivos (FTP), correo electrónico y algunas aplicaciones interactivas [9]. Después de la introducción del WWW, que está basado en el protocolo HTTP en la capa de aplicación y TCP en la capa de transporte, el tráfico WEB se convirtió en el dominante de la composición del tráfico agregado de Internet.

Hasta ahora, los análisis de tráfico en Internet muestran que el tráfico TCP es el dominante [1], lo que es predecible, basándose en los principios del Internet actual, el cual fue creado para brindar básicamente un tipo de servicio, el *best effort*, para todos los servicios y no tiene mecanismos probados para el soporte de calidad de servicio.

La figura 3.2 muestra medidas de tráfico en un enlace entre un Proveedor de Servicios de Internet (*Internet Service Provider, ISP*) y el Internet obtenidas de [1]. Las medidas muestran separación de tráfico, entre protocolos de transporte utilizados por diferentes aplicaciones. La misma conclusión para el papel dominante del tráfico TCP en Internet, se encuentran en [8][9].

[17] Se refiere al tráfico generado por un usuario final que accede al contenido de páginas disponibles en el Internet

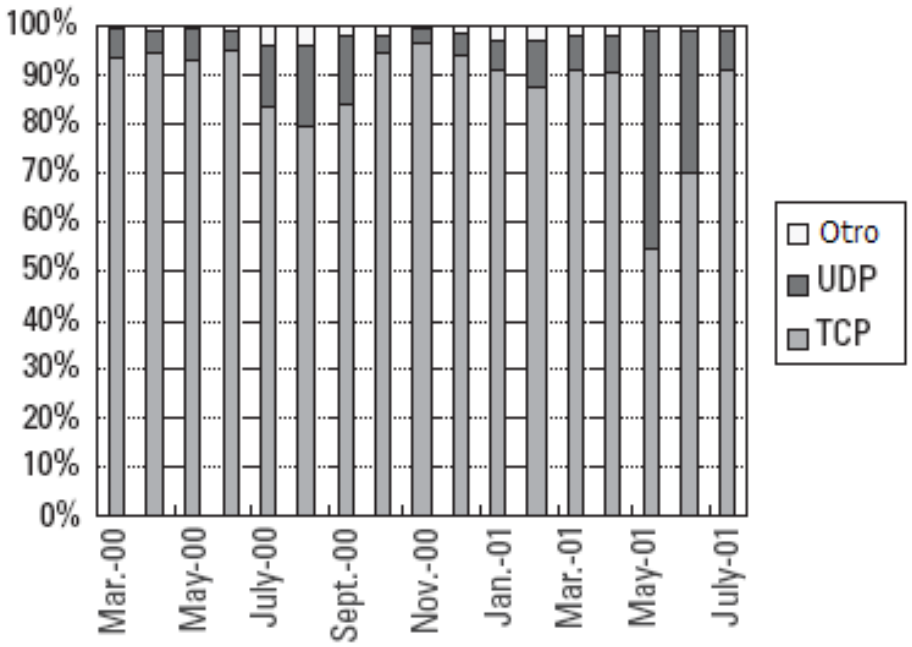

Figura 3.2 Análisis del tráfico de Internet basado en protocolos [1]

3.4.2 Componentes del tráfico de Internet

Usualmente el tráfico de Internet se clasifica con base en su protocolo de transporte (TCP y UDP) o según la aplicación (WEB, FTP, *Telnet* o correo electrónico) utilizada. Cada uno de estos tipos de tráfico consiste de varios flujos multiplexados de diferentes conexiones.

Como se mencionó anteriormente TCP es el protocolo dominante en el Internet actual; según el análisis realizado en [1], la distribución del tráfico agregado TCP, según aplicación, pertenecía entre un 55% y un 90% al WWW, otro pequeño segmento de este tráfico es generado por FTP, el protocolo SMTP (*Simple Mail Transfer Protocol, SMTP*) y otros protocolos de la capa de aplicación; en la Figura 3.3 se muestra la distribución de tráfico en un año.

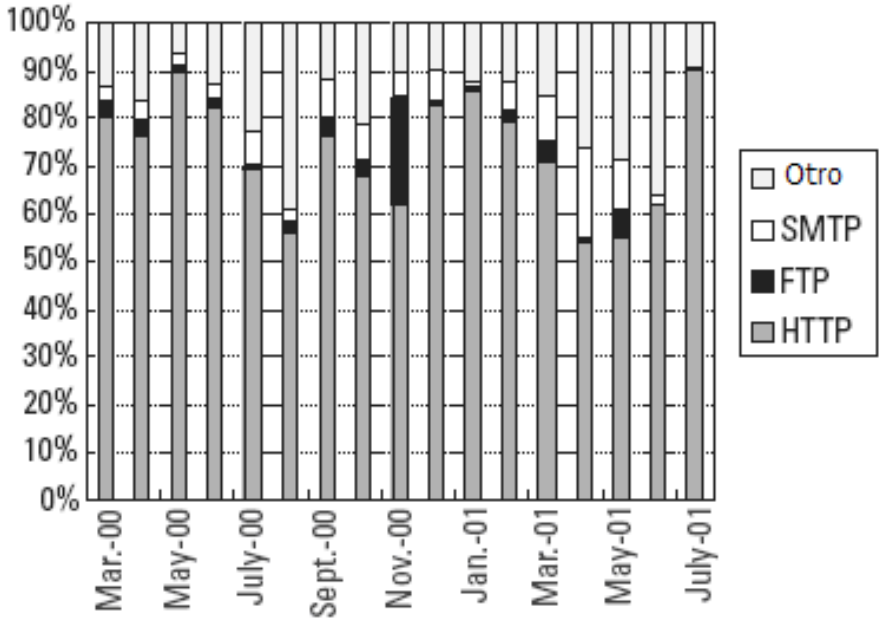

Figura 3.3 Análisis del tráfico TCP [1]

Actualmente el protocolo UDP es utilizado principalmente para comunicaciones entre servidores y para tráfico del Servicio de Nombres de Dominio (*Domain Name Service, DNS*). UDP es adecuado para servicios de tiempo real y puede ser utilizado en combinación con el Protocolo de Tiempo Real (*Real-Time Protocol, RTP*).

3.5 CLASIFICACIÓN DEL TRÁFICO, CON BASE EN LAS EXIGENCIAS DE CALIDAD DE SERVICIO

La heterogeneidad del tráfico en las redes actuales y en Internet es una necesidad que se debe considerar; el resultado de esta heterogeneidad es un amplio rango de servicios con diversas características y diferentes demandas a la red. Para diseños efectivos de red, en especial cuando se tiene un acceso a un WISP o un enlace inalámbrico, como en el caso específico de este proyecto, se debe clasificar el tráfico que existe actualmente, como también el tráfico que se espera en la red en el futuro.

Se ha considerado una clasificación del tráfico, con base en su demanda de Calidad de Servicio para diferentes servicios. Esta caracterización de los mismos se basa en:

- Tipo de servicio (audio, video, datos y multimedia).
- Distribución de información.

Tabla 3.2 Clasificación de Servicios basada en tipo, requerimientos de tiempo real y demanda de Calidad de servicio

APLICACIÓN	AUDIO	VIDEO	DATOS	Demanda de tiempo real	QoS
WWW			X	2	3
Telefonía IP	X			1	1
Video Conferencia (Video y audio)	X	X	X	1	1
Audio	X			2	2
Video	X	X		2	2
Descarga de archivos			X	3	3
Correo Electrónico			X	3	3
Correo Multimedia	X	X	X	3	2
Comercio electrónico			X	1	1
Servicios *On Demand*	X	X	X	2	2
Requerimientos: 1-Alto; 2-Medio;3-Bajo					

La tabla 3.2 [1] no incluye todos los posibles servicios, no es ni siquiera posible hacerlo, se consideraron servicios de diferentes tipos y demandas de QoS; como se ha mencionado, las aplicaciones actuales más comunes no tienen requerimientos de tiempo real (el correo electrónico y el WWW, las cuales utilizan servicios *best effort*), pero aplicaciones como la voz sobre IP (VoIP) y el video son cada vez más comunes y utilizadas.

Como se mencionó previamente en la sección 3.1.3, las aplicaciones interactivas (como la telefonía IP) tienen requerimientos altos ante los retardos de paquetes y el *jitter*. Cuando la gente interactúa en tiempo real, retardos o *jitter* mayores a unos cuantos cientos de milisegundos, causan un impacto significativo en la calidad percibida de la comunicación.

En el caso de la VoIP, según [10], retardos desde 0 a 150ms son aceptables, entre 150 y 400ms pueden ser tolerados, pero retardos de más de 400ms causarían un efecto negativo notable; este tiempo aceptable, comprende todo el tiempo en los nodos de comunicación entre el emisor y el receptor. Otras comunicaciones interactivas de audio y video son también muy exigentes en cuanto a la tolerancia de retardos y las variaciones de retardos.

Las pérdidas no son deseadas en ningún tipo de servicio, pero éstas pueden pasar desapercibidas utilizando técnicas de enmascaramiento de errores.

El principal servicio interactivo, que es la telefonía, requiere bajo ancho de banda [1], dadas las características de la voz humana. Esta característica, junto a la sensibilidad ante los retardos, la pérdida de paquetes y el *jitter*, llevan a que, los paquetes de este servicio ingresen a *buffers* prácticamente vacíos. Esto es posible si, los paquetes de la telefonía IP y servicios similares no se mezclan con otro tipo de tráfico en los *buffers* (por ejemplo, con el tráfico masivo como el de aplicaciones *peer-to-peer*).

Si se ponen todos los paquetes en los mismos *buffers*, no se podría aplicar efectivamente la planificación y se agregaría a la red retardos inmanejables y posibles pérdidas de la información sensible al tiempo. Una solución efectiva es colocar los datos de "alta prioridad", como los de telefonía IP o los del tráfico que se considere prioritario en la red, en *buffers* separados y atender estas colas antes que otros datos; este es un esquema de prioridad. Es importante mencionar que existen muchos otros esquemas para separar y proteger información afectada por el tiempo o las pérdidas en las aplicaciones interactivas de tiempo real.

Por otra parte, servicios como el correo electrónico, no tienen mayores requerimientos ante el retardo de paquetes y el *jitter*, este tipo de servicios pueden ser atendidos por el enlace cuando se tienen suficientes recursos libres. La aplicación WWW no demanda bajos retardos ni bajo *jitter*, pero no es tan tolerante como el correo, esto se debe a que es una aplicación cliente-servidor interactiva donde el usuario que hace un click espera cuanto antes se complete la página solicitada. Si se presentan muchas pérdidas o retardos, se deteriora notablemente el servicio al causar discontinuidad en la transmisión de datos y retardos inaceptables en la comunicación, por lo que se considera que este servicio, demanda una calidad de servicio más alta que el clásico *Best Effort* utilizado en el Internet actual.

Servicios de distribución, como la distribución de audio y video, se pueden considerar algo tolerantes ante los retardos y las variaciones de retardos. Los retardos aceptables están en el rango de los segundos, lo que depende de los *buffers* de reproducción en los receptores.

Los umbrales de tolerancia para estos retardos, son más altos que los niveles para comunicaciones interactivas. La tolerancia ante pérdidas depende de cada servicio en particular, por ejemplo, para la distribución de video se requiere pérdidas más bajas que para la video-conferencia. Los paquetes perdidos reducen la calidad percibida del video, porque la información está ya comprimida cuando se envía por el enlace de transmisión. Los codificadores de video utilizan codificación espacial y temporal para eliminar información redundante en los cuadros de video [11].

Es importante considerar que las aplicaciones de video tienen los requerimientos más altos de ancho de banda por conexión, por lo cual, no se puede tener la misma prioridad que con servicios interactivos, los cuales tienen requerimientos diferentes y demandan menos ancho de banda.

3.6 CONTROL DE TRÁFICO EN LINUX

Linux es una variante de UNIX que data de 1991, el cual ha estado en continuo desarrollo por la comunidad de código abierto. Su código fuente está disponible gratis bajo la licencia GPL[18]. Estas características y que Linux es conocido por la alta flexibilidad de código de interconexión el cual permite la configuración de *firewalls* y enrutadores (incluyendo soporte para todos los protocolos de red conocidos) son las principales razones para escoger este sistema operativo en la implementación del proyecto.

El módulo del sistema operativo Linux responsable de los procesos de división de ancho de banda y de planificación de paquetes, es el de Control de Tráfico (*Traffic Control, TC*) el cual ha estado disponible desde el kernel 2.1.90; a partir la versión 2.3 se convirtió en un componente de la versión estándar del kernel. Aunque TC se enfoca a la conformación del tráfico de salida, también tiene soporte para el ajuste del tráfico de entrada.

Como en la mayoría de los sistemas operativos UNIX, la implementación del código de interconexión (*networking*) sigue una versión simplificada del modelo de referencia OSI. De las capas que se ven en la figura 3.4, las funciones de la capa de enlace, de la capa de red y de la de transporte son ejecutadas en el kernel, los protocolos de la capa de aplicación y la aplicación en sí corren en espacio del usuario. La planificación de paquetes y el código de control de tráfico se implementan debajo de la capa de red, justo encima del *driver* o controlador del dispositivo. Así diferentes protocolos de nivel de red como IP, IPX y *Appletalk* pueden soportarse y el planificador puede reordenar los paquetes justo antes que sean enviados a la interfaz de red [12].

Cabe aclarar que la identificación de tráfico se puede hacer de distintas formas y en varios niveles, sin embargo, el control de tráfico se implementa en el nivel más bajo posible del sistema operativo, el cual es independiente del dispositivo, existen muchas ocasiones, especialmente en ambientes inalámbricos, donde TC puede beneficiar del soporte de conformación de tráfico a las capas inferiores [12].

[18] General Public License O Licencia para Público General

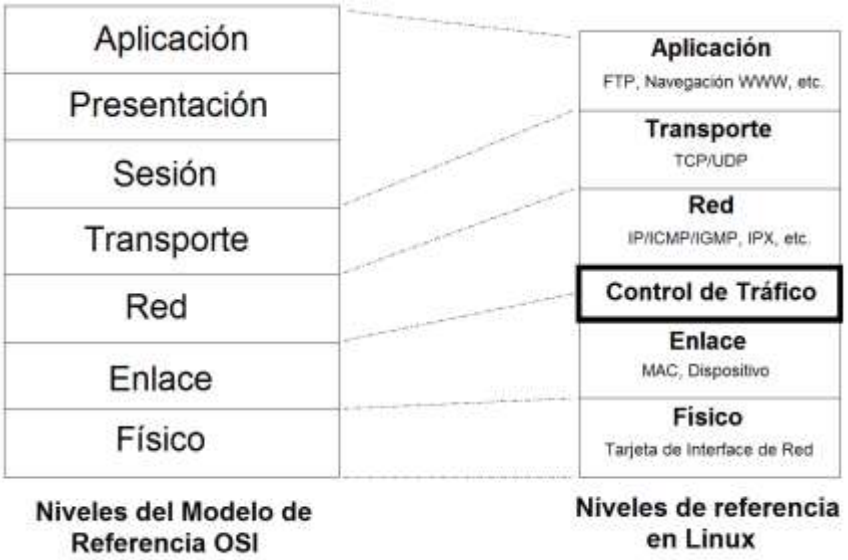

Figura 3.4 Interconexión en Linux y el Modelo de referencia OSI

La figura 3.5 muestra el camino de un paquete de datos a través de la pila de la arquitectura de red en Linux. Cuando un paquete llega a la interfaz de red y es pasado al nivel de red, este puede ser clasificado y puede asignársele determinado tratamiento (por ejemplo, puede ser eliminado si cierta tasa de transferencia es excedida). Entonces éste puede, o dirigirse hacia las capas superiores, o a la parte de redireccionamiento IP (*IP Forwarding*), donde es determinado el próximo salto para el paquete. Cuando éste alcanza la parte de salida en el control de tráfico es clasificado y conformado para su transmisión. La decisión de conformación puede tener varias opciones como son su eliminación, retraso, envío inmediato, entre otras [12] [13].

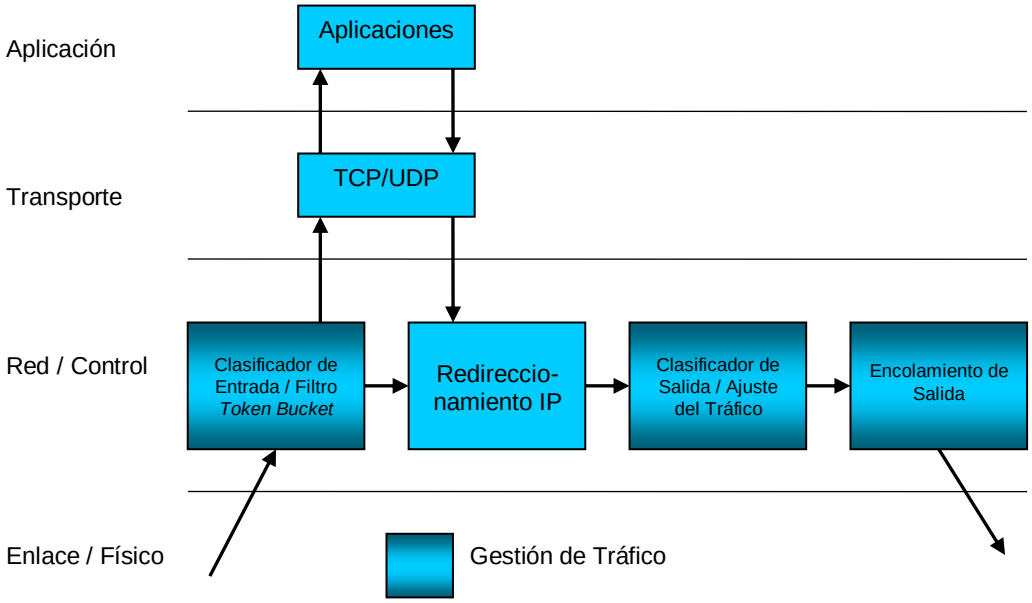

Figura 3.5 Niveles donde se ve involucrado el control de tráfico

3.6.1 NetFilter

Netfilter es el conjunto de *hooks*[19] dentro del Kernel de Linux para interceptar y manipular los paquetes de red. El principal componente superior a *netfilter* es el *firewall*, el cual filtra los paquetes.

Cada paquete que llega al sistema desde la red, atraviesa toda la pila de protocolos, la cual convierte las señales eléctricas que llegan a la tarjeta de red en mensajes que los servicios o *daemons* entiendan. Cuando un *daemon* envía una respuesta, el Kernel genera un paquete y lo envía a través de la tarjeta de red. Mientras los paquetes de entrada se procesan hacia arriba en la pila de protocolos y los de salida hacia abajo, *Netfilter* crea *hooks* hacia el Kernel de Linux y permite controlar el destino de los paquetes.

Se puede decir que un paquete atraviesa el sistema *Netfilter*, por qué estos *hooks*, brindan la posibilidad de modificar la información de los paquetes en varios puntos entre la llegada y la salida del procesamiento de paquetes.

Netfilter utiliza las listas de reglas *INPUT, FORWARD, OUTPUT, PREROUTING y POSTROUTING* e intenta comparar cada paquete que atraviesa el Kernel; cada una de estas reglas se describe en la tabla 3.3. Si encuentra una comparación acertada, el paquete se procesa inmediatamente de la forma descrita por la regla [14], sin intentar comparar con otras reglas. Si ninguna de las reglas es comparable con el paquete, el destino del paquete se determina por la configuración especificada por el administrador de red.

Tabla 3.3 Reglas por defecto en NetFilter

Cadena	**Descripción**
INPUT	Reglas para paquetes de entrada
OUTPUT	Reglas para paquetes de salida
FORWARD	Reglas para paquetes reenviados
PREROUTING	Reglas para redireccionar o modificar paquetes de entrada
POSTROUTING	Reglas para redireccionar o modificar paquetes de salida

3.6.2 IPtables

Es una aplicación de línea de comandos utilizada para configurar el módulo de filtraje de paquetes IPv4 en Linux 2.4.x y 2.6.x. Dado que la traducción de direcciones de red o *Network Address Translation (NAT)* se configura también desde el módulo de filtrado de paquetes, *iptables* también se utiliza para esta función [15]. El paquete que contiene a *iptables* incluye una aplicación denominada *ip6tables*, que es utilizada para la configuración de paquetes IPv6.

[19] En programación HOOKING es una técnica que emplea *hooks* para implementar una cadena de procesos

Los filtros IP, operan más que todo en el nivel 2 de la pila de protocolos TCP/IP, sin embargo, *iptables* tiene la capacidad de trabajar también en el nivel 3, algo que la mayoría de filtros IP actuales pueden hacer, pero por definición los filtros IP trabajan en el nivel 2 [15].

Las implementaciones de filtros IP, se basan en las cabeceras IP de los paquetes para realizar el proceso de filtrado (Direcciones de fuente y destino, TOS/DSCP/ECN, etc.), *iptables* no solo utiliza las cabeceras sino que también puede basarse en otra información dentro del paquete como TCP, UDP e incluso la dirección MAC fuente. La configuración específica utilizada en este proyecto se incluye en el capítulo 4, la información para utilizar y descargar esta herramienta está disponible en [16] [17] respectivamente.

En general y para el proyecto, Iptables es la herramienta que permite integrar las funciones de un planificador de tráfico en un *script*.

3.6.3 Tabla *Mangle*

De las funcionalidades de *Netfilter*, la tabla *Mangle*, es la base de las instrucciones del *script* de código abierto utilizado en este proyecto, que puede ser visto en el ANEXO 1. Esta tabla se utiliza para manejar la información sobre los paquetes y conexiones. Cuando se configura el control de tráfico, hay reglas que son aplicadas específicamente hacia ésta tabla, estas reglas son diseñadas para controlar el comportamiento de los paquetes, como por ejemplo, enrutamiento, prioridad, etc. Las reglas que modifican los paquetes, en lugar de simplemente redireccionarlo o detenerlo, pueden ser utilizadas únicamente en la tabla *Mangle*. La tabla también es usada para establecer la marca, cuando la identificación de conexiones se hace basada en el puerto destino de la comunicación.

Por ejemplo, para cambiar la prioridad de un paquete con base en la bandera del campo ToS, se puede hacer directamente en la tabla sin necesidad de modificar el paquete IP, como sucede en el caso concreto de esta aplicación, donde las marcas se hacen directamente en la tabla, dejando la trama intacta. A continuación se muestra un ejemplo del *script* que realiza la acción mencionada:

```
for PROTO in $TOS_EXPR; do
iptables -t mangle -A egress_chain -m tos --tos $PROTO -j MARK --set-mark 1
done
```

En el ejemplo se identifica el campo ToS y según éste se establece una marca en la tabla *Mangle,* la cual corresponde a la prioridad que recibe el paquete, en éste caso es “1”, que en el caso particular significa máxima prioridad.

3.6.4 Targets

Los *Targets* especifican la acción que se debe tomar cuando se encuentra una coincidencia entre la regla y el paquete. Estos tienen funciones incorporadas, pero también, pueden especificar una cadena definida por el usuario, la cual se debe pasar a ejecutar. Sí no se encuentra ninguna de las reglas que coincida, en la cadena

definida por el usuario, la ejecución vuelve a la cadena que la llamó. La tabla 3.4 muestra los *targets* de la tabla *Mangle*.

Tabla 3.4 Targets de la tabla Mangle

Target	**Descripción**
TOS	Modificar el tipo de servicio para controlar la prioridad del paquete
TCPMSS	Modificar la cantidad de paquetes permitidos para cada conexión, permitiendo transmisiones más largas
DSCP	Modificar bits DSCP
ROUTE	Modificar información de destino en el paquete
MARK	Su propósito es fijar un marcador en la tabla *Mangle*, el cual puede ser leído posteriormente cuando, por ejemplo, se toma una decisión de enrutamiento. Este crea la posibilidad de priorizar tráfico.

En la siguiente expresión, tomada del *script* utilizado en el proyecto:

```
for PORT in $UDP_EXPR; do
    iptables -t mangle -A mark_chain -m mark --mark 0 -p udp --dport $PORT
-j MARK --set-mark 1
  done
```

El *target* es el puerto destino de la comunicación, de acuerdo a él se establece una marca en la tabla *Mangle,* esta marca corresponde a la prioridad que recibe el paquete, la cual es “1”, que en el caso particular significa máxima prioridad. Sin que se haya puesto una marca o cambiado algún campo en el paquete, se consultará la tabla *mangle* para darle el tratamiento correspondiente a la conexión establecida en dicho puerto.

El módulo IPtables está diseñado para ser escalable, permitiendo agregar *targets* posteriormente según la aplicación; esto implica aplicar parches al Kernel y reconfigurarlo. En la tabla 3.5 se describen los *targets* por defecto de IPtables.

Tabla 3.5 Targets de IPtables

Target	**Descripción**
ACCEPT	Permite a el paquete atravesar el *firewall*
DROP	Denegar acceso por paquete
REJECT	Denegar acceso y notificar al emisor
QUEUE	Enviar el paquete a espacio del usuario
RETURN	Ir al final de la cadena y permitir que sea procesado por el *target* por defecto

3.6.5 Extensiones de comparación

Además de las funciones de comparaciones genéricas y específicas de protocolo que se cargan automáticamente, otras comparaciones se pueden agregar utilizando el comando –m.

La extensión *mark,* utilizada en la configuración del *script* parte de este proyecto, permite comparar paquetes que han sido marcados previamente utilizando el *target mark* descrito en la tabla 3.4. Es muy importante resaltar que los marcadores son creados y manejados por IPtables y no alteran la información del paquete de ninguna manera.

En la siguiente sintaxis del *script* utilizado, se hace una comparación al verificar si un paquete tiene la marca "0", lo cual significaría que aún no ha sido marcado, de ser así, el planificador cambia su marca a "3".

```
iptables -t mangle -A mark_chain -m mark --mark 0 -j MARK --set-mark 3
```

3.6.6 Componentes del Control de Tráfico en Linux

La flexibilidad de la arquitectura del control de tráfico en Linux se obtiene mediante módulos de calidad de servicio y de división del enlace, los cuales pueden ser combinados en múltiples formas para proveer los requerimientos de planificación en nodos específicos. Estos módulos incluidos en el Kernel pueden ser configurados mediante un control de usuario el cual se comunica con el Kernel utilizando una clase especial de *socket* de red (el *Netlink socket*) [18]. Además de entregar parámetros al Kernel, como los requerimientos de ancho de banda, el control del usuario también especifica la estructura de la arquitectura de calidad de servicio para el funcionamiento del nodo.

Los componentes básicos de la arquitectura de control de tráfico en Linux son:

- Disciplinas de colas
- Clases
- Filtros

Todas las otras configuraciones de planificación complejas son implementadas, combinando estos tres componentes, de una forma que se obtenga el comportamiento deseado.

3.6.6.1 Disciplinas de Colas

La disciplina de colas (*qdisc*) tiene una función de encolamiento y desencolamiento y realiza reordenamiento de los paquetes almacenados. La función de encolamiento se invoca cada vez que el nivel de red del sistema operativo desea transmitir un paquete y la función de desencolamiento se invoca cuando está en capacidad de transmitir el siguiente paquete.

Qdisc puede también funcionar como un contenedor con filtros y clases. Un filtro es un módulo el cual se utiliza para determinar la clase del paquete. La clase es un grupo

lógico de paquetes los cuales comparten una misma propiedad. Lo que hace este concepto flexible, es que una clase usualmente utiliza otra vez una disciplina de colas diferente, para encargarse de almacenar los paquetes.

Las disciplinas de colas disponibles pueden dividirse en dos grupos [18]: Las *qdisc* simples que no tienen clases internas (conocidas como *queues* en la terminología de Linux TC) y las que tienen clases (conocidas como *schedulers*).

Dentro de las *queues* se encuentran:

- *pfifo_fast*: Es una cola FIFO con 3 bandas[20] de prioridades. Los paquetes de la banda más baja serán enviados siempre de primeros; dentro de cada banda se utiliza el mecanismo FIFO común. Esta disciplina funciona por defecto y es la utilizada para cada clase recién creada.

- *sfq*: Disciplina de encolamiento Estocástico Justo[21] (*Stochastic Fair Queuing*). Esta disciplina es una implementación del mecanismo de encolamiento justo (FQ), no es tan acertada como otros mecanismos, pero requiere de menos cálculos y es casi perfectamente justa. Reparte equitativamente los recursos del canal.

- *tbf*: Filtro *Token Bucket*, el cual pasa los paquetes solo a una tasa previamente especificada pero tiene la posibilidad de permitir algunas ráfagas.

- *red*: Implementa un mecanismo denominado Detección Temprana Aleatoria (*Random Early Detection, RED*), el cual empieza a descartar paquetes antes de que la cola esté completamente llena, con la finalidad de que las conexiones TCP adapten sus tasas de transmisión [19].

- *gred*: Una generalización del mecanismo RED utilizada para soportar el modelo *DiffServ*.

- *ingress*: Una disciplina utilizada para implementar *policing* en el tráfico entrante.

Las disciplinas basadas en clases, más utilizadas son las siguientes:

- *cbq*: Implementación del mecanismo CBQ.

- *atm*: Una disciplina especial la cual soporta el redireccionamiento de flujos hacia canales virtuales (VC) ATM. Cada flujo puede ser mapeado en un canal virtual diferente o múltiples flujos pueden compartir un VC.

- *csz*: Disciplina de colas Clark-Shenker-Zhang[22].

- *dsmark*: Es un mecanismo utilizado para el soporte de Servicios Diferenciados.

- *wrr*: Implementa el mecanismo WRR.

---

[20] Se refiere a niveles de prioridad

[21] Es una modificación al encolamiento justo (FQ). Es menos preciso que otros algoritmos de FQ pero requiere de menos cálculos y es eficientemente justo.

[22] Mecanismo de planificación el cual se basa en crear flujos WFQ para cada servicio garantizado.

- *htb*: Implementación del mecanismo HTB

3.6.6.2 Clases

Una *qdisc* que soporta clases, generalmente tiene una clase raíz la cual puede tener una o más subclases [18]. Una clase se puede identificar de dos maneras: ya sea por la identificación asignada por el usuario (*class ID*) la cual está compuesta de un número principal correspondiente a la instancia *qdisc* a la cual pertenece y un número secundario único; o por su identificación interna (*internal ID*). Como se mencionó previamente cuando se crea una clase, la disciplina de cola por defecto es la pfifo_fast; esto se puede modificar mediante una operación de asignación de disciplinas, la cual asigna una disciplina de colas a la clase. Una clase también puede soportar operaciones para atar una instancia de un filtro a una clase. Aun así, las clases y los planificadores son dos componentes diferentes, los métodos para manejarlos se implementan, generalmente, en un módulo del Kernel, en cambio, los filtros, en la mayoría de los casos, utilizan módulos independientes.

En la siguiente sintaxis, se puede ver un ejemplo sencillo de la definición de clases y sus características, en la que básicamente se crea una cola HTB madre con 3 clases hijas.

```
tc class add dev $DEV0 parent 1: classid 1:1 htb rate ${MAX}kbps
ceil ${MAX}kbps
tc class add dev $DEV0 parent 1:1 classid 1:10 htb rate ${MAX}kbps
ceil ${MAX}kbps prio 1
tc class add dev $DEV0 parent 1:1 classid 1:20 htb rate 4000kbps
ceil 4000kbps prio 2
tc class add dev $DEV0 parent 1:1 classid 1:30 htb rate 2kbps ceil
${MAX}kbps prio 3
```

La notación 1:1 se refiere a la cola madre, los parámetros "`ceil`" y "`rate`" están relacionados con el uso del ancho de banda disponible y se explican en la sección 3.6.9, en este caso la palabra "`MAX`" hace alusión a una variable con este nombre, la cual debe tener un valor numérico.

A continuación se hace referencia a las clases hijas mediante la notación "`1:10, 1:20, 1:30`", a las cuales, se les asigna el ancho de banda correspondiente y un nombre, por así decirlo, ya que en el contexto del planificador, corresponde a la prioridad, esto es lo que expresa el parámetro "`prio 1, 2 y 3`".

3.6.6.3 Filtros

Las disciplinas de colas y las clases utilizan filtros para clasificar los paquetes entrantes. Tienen una lista, en orden de prioridad, de los filtros para cada protocolo por donde pasa el paquete, hasta que uno de los filtros indica un acierto. Los filtros para protocolos iguales deben tener diferentes prioridades [20]. Linux soporta los siguientes filtros:

- Filtros Genéricos: Pueden clasificar paquetes para todas las clases de una disciplina de colas.

- *cls_fw*: El paquete se clasifica según una marca (*tag*) asignada por el *firewall*/código *netfilter*.

- *cls_route*: El paquete se clasifica utilizando información de tabla de enrutamiento.

- Filtros específicos: Pueden clasificar paquetes solo para una clase específica.

  - *cls_rsvp*: Un filtro para soportar el protocolo RSVP.
  - *cls_u32*: Este filtro clasifica paquetes de acuerdo con un byte arbitrario de muestra en la cabecera del paquete (como en los campos ToS y DSCP, explicados en el capítulo 2) o en la carga (*payload*).

- Filtro L7: Este filtro es una herramienta de clasificación para *Netfilter* el cual identifica paquetes basándose en la comparación e identificación de patrones de la capa de aplicación [21] en la trama IP. Puede clasificar paquetes según la aplicación como por ejemplo *Kazaa*, HTTP, *Jabber*, *Citrix*, *Bittorrent*, FTP, *eDonkey*, etc., indiferentemente del puerto que se esté utilizando. L7 sirve como complemento a clasificadores existentes los cuales pueden comparar direcciones IP, números de puerto y otros.

- IPP2P: Se utiliza para identificar datos *peer-to-peer*[23] (P2P) en el tráfico IP. Extiende la arquitectura de *iptables/netfilter* con un nuevo módulo de comparación. IPP2P se puede integrar a Linux y su funcionalidad puede ser utilizada agregando reglas de filtraje apropiadas [22].

  IPP2P utiliza patrones de búsqueda apropiados para identificar tráfico P2P, obteniendo un resultado creíble de tráfico proveniente de varias redes P2P [22]. Una vez identificado, se puede tratar este tráfico de varias maneras, ya sea descartándolo, poniéndolo en clases de baja prioridad o conformándolo a un límite de ancho de banda.

3.6.7 IPROUTE2

La mayoría de distribuciones Linux y Unix, utilizan los comandos de configuración *arp, ipconfig* y *route*, estas herramientas funcionan, pero han mostrado un comportamiento impredecible en los Kernel 2.2 y superiores, los cuales incluyen un subsistema de red completamente rediseñado. Este nuevo código de aplicaciones de red le brinda a Linux un buen desempeño, es más, las funciones de enrutamiento, filtraje y clasificación incluyen más funciones que las de algunos enrutadores, *firewalls* y dispositivos de control de tráfico en el mercado. Iproute2 es una colección de utilidades para controlar la interconexión TCP/IP y el control de tráfico en los Kernel de Linux más recientes [23].

[23] Un tipo de red de computadores que recae básicamente en la capacidad de cómputo y el ancho de banda de los equipos participantes en la red, en lugar de los servidores.

3.6.8 Administración y configuración de Linux TC

Como se mencionó previamente en este capítulo, el control de tráfico (TC) es parte de las versiones del Kernel Linux más recientes, entonces, no es necesario utilizar parches especiales o hacer recompilaciones del Kernel, si se utiliza una versión 2.3.x o superior, a menos que sea necesario el uso de módulos adicionales. La única precondición para utilizar mecanismos de planificación, es asegurarse que la configuración correspondiente se mantiene, para la estructura de los componentes de control de tráfico deseados.

Dado que el control de tráfico es parte del Kernel de Linux pero tiene que ser administrado corriendo programas en espacio de usuario, se desarrolló una forma estandarizada en la cual los programas de usuario se pueden comunicar con el kernel del sistema operativo: la información es encapsulada en un formato especial de mensajes (mensajes *Netlink*) y enviados por un *socket* especial llamado *socket Netlink* [18]. El kernel recibe y analiza los mensajes y responde con un mensaje de confirmación en el *socket*. Ya que cada parte funcional del Kernel necesita diferentes parámetros, se utiliza un protocolo especial para cada área. Para manipular enrutamiento e información de control de tráfico se utiliza el protocolo *RTNetlink* [18].

A pesar de que, básicamente cada programa de usuario puede generar mensajes *Netlink* (si tiene el permiso necesario), el programa de configuración usualmente utilizado para configurar los componentes de calidad de servicio es el programa de control de tráfico *tc*, el cual es parte del paquete *iproute2* [18] [20]. La información ampliada acerca de este paquete está disponible en [24], este documento explica posteriormente la aplicación específica utilizada en el proyecto.

3.6.9 *Hierarchical Token Bucket* (HTB)

El *Token Bucket* Jerárquico (HTB) es un planificador de paquetes [25], con un conjunto de parámetros de configuración similares a CBQ. Tanto HTB como CBQ ayudan a controlar el uso del ancho de banda de un enlace determinado; también permiten utilizar un solo enlace físico para simular enlaces más lentos y enviar diferentes clases de tráfico en diferentes enlaces simulados. En los dos casos se requiere especificar cómo dividir el enlace físico en los enlaces simulados y cómo decidir cuál de los enlaces simulados utilizar para enviar determinado tipo de paquetes.

CBQ es un mecanismo complejo y que no funciona bien en varias situaciones típicas [18], de ahí nace la necesidad de un mecanismo mejor, que se traduce en la creación de HTB. HTB es una evolución de CBQ, maneja menos variables de funcionamiento y la sintaxis de codificación es mucho más simple, estas características le dan a HTB más escalabilidad y flexibilidad.

Conceptualmente, HTB es una disciplina de encolamiento con clases, con un número arbitrario de *Token Buckets* organizados jerárquicamente. Considerando un escenario simple, la principal disciplina de salida en un dispositivo es conocida como "*root qdisc*" o disciplina de encolamiento raíz. La disciplina raíz contendrá una clase (en escenarios complejos puede tener múltiples clases como ramas de la disciplina raíz). Esta clase HTB tiene dos parámetros principales de configuración: "*rate*" y "*ceil*", valores que representan el ancho de banda total disponible en el enlace.

En HTB, "`rate`" significa el ancho de banda garantizado para una clase dada, y "`ceil`" es el máximo que la clase puede consumir. Cualquier requerimiento para el ancho de banda en el rango de estos dos parámetros es tomado del asignado a otras clases hermanas, que no está siendo usado. Cabe anotar que en la clase principal el valor para "`ceil`" y "`rate`" debe ser el mismo.

Un número de clases hijas pueden establecerse sobre la principal, cada una de ellas tiene asignada una porción del ancho de banda disponible en las clases madre. En estas clases hijas, "`rate`" y "`ceil`" no necesariamente deben tener el mismo valor; de esta manera se puede reservar recursos para cada una de las clases, y compartir el ancho de banda que no está siendo usado entre las clases que lo requieran, planificando la distribución del tráfico y haciendo uso eficiente de los recursos de la red [20].

HTB puede coexistir con colas que sigan disciplinas diferentes, lo que permite ejercer control sobre parámetros adicionales, al igual que hace posible ajustar los sistemas de planificación de tráfico hasta satisfacer las necesidades del administrador de red. HTB utiliza el planificador DRR real [26], explicado en el capítulo 1 en la sección 1.2.6.

Al igual que el marcador de tres colores de tasa única y de dos tasas, explicados en las secciones 2.2.1 y 2.2.2 respectivamente, HTB también utiliza el concepto de los colores, para identificar el modo de una clase. Con base en los parámetros: tasa actual[24] (R), *ceil* (CR) y *rate* (AR) se definen los tres colores:

- **Rojo**: R>CR, no puede transmitir.
- **Amarillo**: R≤CR y R>AR, puede tomar ancho de banda prestado.
- **Verde**: Otro caso, puede transmitir.

La configuración específica del control de tráfico en Linux utilizada en este proyecto se explica en el capítulo 4. Esta configuración sirve también como ejemplo para entender el funcionamiento de TC y cómo puede ser administrado.

3.6.10 *Intermediate Queueing Device* (IMQ)

Esta aplicación no es exactamente un qdisc, pero su uso la relaciona con la funcionalidad de qdiscs. En Linux, qdiscs se adhieren a los dispositivos de red y todo el tráfico que es encolado al dispositivo se encola primero en los qdiscs; a partir de este concepto se encuentran dos limitaciones [27]:

1. Solo es posible conformación de salida (*egress*). Existe una cola de ingreso (*ingress*) la cual únicamente puede implementar limitación de tasa. Con IMQ es posible utilizar qdiscs de *egress* para conformación real *ingress*.

2. qdiscs se adhieren a los dispositivos, como consecuencia de esto una qdisc solo puede manejar tráfico que va hacia la interfaz a la cual esta adherida. Algunas veces se requiere tener límites globales en múltiples interfaces, con IMQ se puede utilizar *iptables* para especificar que paquetes el qdisc ve, permitiendo utilizar limites globales.

[24] Es la tasa del flujo de paquetes saliendo de la clase y se mide en periodos de tiempo cortos

En la figura 3.6 se explica la interacción entre los componentes hasta ahora mencionados, NetFilter e iptables actúan como puente entre todos los componentes, integrando los módulos en un solo sistema de control de tráfico. Un paquete que va a entrar dentro de este sistema, debe seguir una secuencia, primero es identificado, acción que puede ejecutarse usando distintas técnicas (filtros de nivel 7, campo tos, puerto destino, etc.), de acuerdo a la identificación es marcado, se enruta o se descarta, después de este proceso ingresa a la cola de salida de la interfaz y sigue las políticas de tratamiento establecidas.

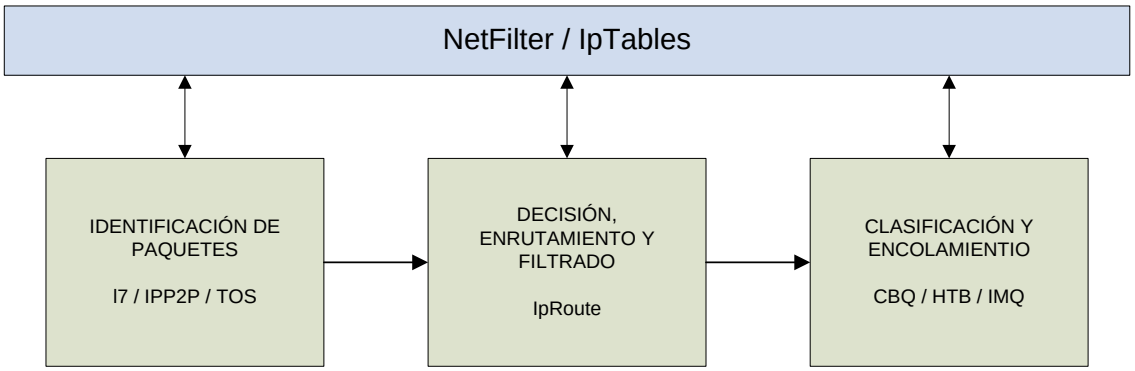

Figura 3.6 Interacción entre las herramientas de control de tráfico Linux

3.6.11 Control de Tráfico de Nueva Generación (TCNG)

El proyecto de Control de Tráfico de Nueva Generación (TCNG) se enfoca en brindar un lenguaje de configuración más amigable para el usuario y soportar aceleradores de *hardware* en control de tráfico [20]. Los dos componentes principales desarrollados son el Compilador de Control de Tráfico (TCC) y el Simulador de Control de Tráfico (TCSIM).

El concepto de Compilador de Control de Tráfico consiste en aceptar una amplia variedad de lenguajes de entrada (desde lenguajes amigables para el usuario hasta lenguajes más amigables para la máquina como XML) y convertirlos en una estructura de datos unificada. Esta información es enviada a una variedad de procesadores de salida, como por ejemplo *tc* utilizando el *socket* de *Netlink* para configurar el Kernel.

TCSIM es un ambiente de simulación de los componentes de control de tráfico en Linux; funciona en el código original del Kernel. Al utilizar la interfaz tc para configurar los módulos simulados, se puede examinar la comunicación del protocolo Netlink. Aun así TCSIM es parte de TCNG y tiene la capacidad de simular componentes corriendo en la arquitectura TC actual [28].

Ya que la simulación se realiza utilizando casi el mismo código del Kernel de Linux, los resultados son más cercanos al compartimiento actual del sistema que los obtenidos con modelos abstractos de simulación. El uso de TCNG resultaría importante para proyectos que involucren simulaciones.

RESUMEN

En este capítulo, se recopilaron los principales factores a los cuales se exponen los flujos de tráfico en las redes de paquetes actuales, así como, las características y demandas de estos flujos, las cuales deben ser consideradas en proyectos de ésta indole, donde se busca transportar determinados tipos de tráfico, considerados como prioritarios en la red.

Una vez se analizan los efectos, de los posibles factores a los que se expone cada flujo especifico de tráfico, se puede definir el tratamiento y la prioridad con la cual será manejado. También, dada la importancia que tienen los servicios de tiempo de real para este proyecto y para las redes actuales, se debe conocer el grado de sensibilidad que tienen ante inconvenientes que se pueden presentar en la red y las consideraciones pertinentes para transpórtalo efectivamente.

Es difícil predecir las proporciones de cada tipo de tráfico que se espera transportar en la red y estos son temas que se tocan con profundidad en la Ingeniería de Tráfico; la información recopilada en este capítulo revela las tendencias de los usuarios, reflejadas en las aplicaciones que aportan tráfico de red, principalmente los servicios de Internet. A pesar de los varios servicios interactivos ofrecidos actualmente, la mayor parte del tráfico sigue siendo el aportado por la navegación de páginas WEB, pero el crecimiento de los servicios de tiempo real y de aplicaciones como SSH y FTP es notorio y es considerado en el diseño de esta implementación.

También se presentó el control de tráfico en Linux, así como sus componentes y respectivas capacidades, dentro de las cuales se encuentra HTB, principal herramienta para el control de tráfico de este proyecto. En el contexto de Linux TC, HTB es una disciplina de encolamiento con un número arbitrario de *token buckets*, la cual permite simular en un enlace físico varios virtuales y compartir de una forma ordenada los recursos de ancho de banda; ésta utiliza DRR real para planificar el tráfico de salida de las colas hacia el enlace. Es importante resaltar que dentro de IPtables se encuentra la tabla *Mangle*, la cual maneja las marcas establecidas a los paquetes.

4. IMPLEMENTACIÓN Y PRUEBAS DEL PILOTO DE GESTIÓN DINÁMICA DE ANCHO DE BANDA

Este capítulo describe los detalles de implementación de una arquitectura de gestión de ancho de banda para enlaces inalámbricos basada en las capacidades del control de tráfico soportadas en el sistema operativo Linux. Se presentan los requisitos para la adaptación del *framework* de control de tráfico de Linux a ambientes inalámbricos y los detalles de la implementación del planificador de tráfico HTBi, pruebas y resultados de su funcionamiento.

4.1 CONTROL DE TRÁFICO DE LINUX EN REDES INALÁMBRICAS

Para implementar un algoritmo HTB que realice las funciones necesarias para optimizar el funcionamiento de un enlace inalámbrico, el *framework* de control de tráfico de Linux debe cumplir con algunos requisitos:

- Soportar la implementación de una variedad de disciplinas de encolamiento.
- Soportar varias técnicas de identificación del tráfico.
- Configuración de los componentes para la conformación del tráfico con los mismos mecanismos usados en redes cableadas.
- Minimizar el hardware necesario para su correcto funcionamiento.
- Independencia de las características del hardware (por ejemplo, dependencia de controladores o integrados de determinados fabricantes).

Cumpliendo con estos requisitos, la arquitectura de funcionamiento de HTBi se ve en la figura 4.1.

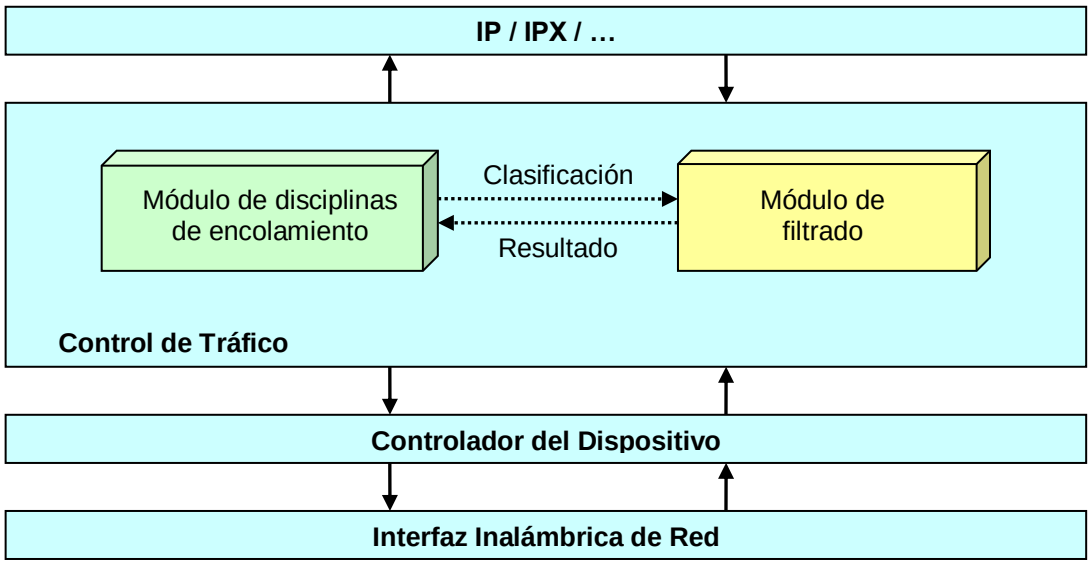

Figura 4.1 Representación esquemática de la arquitectura de funcionamiento de HTBi

Para la integración de las disciplinas de encolamiento en el *framework* de control de tráfico existente sin requerir modificaciones en las capas superiores, es muy importante manejar la interacción estándar de un *scheduller*, visto en 3.6.6.1, de esta manera es posible reutilizar componentes que hagan uso de configuraciones incorporando los modulos disponibles para la clasificación y *policing* para redes cableadas pero usándolos en el ámbito de las redes inalámbricas.

De esta forma, dividiendo la transmisión de datos en cuatro bloques, se deja de lado el tipo de protocolo, los controladores de dispositivos y el tipo de interfaz de red, que en este caso sería una interfáz inalámbrica, se separa un bloque de control de tráfico donde se identifica el tráfico, se clasifica de acuerdo a un módulo de filtrado y de acuerdo a esta clasificación se "reordena" la distribución de los paquetes en la cola de salida de acuerdo a una disciplina de encolamiento.

Sería deseable contar con un monitor del estado del canal inalámbrico, de manera que el funcionamiento del planificador de tráfico pudiera variar dependiendo de la disponibilidad de recursos, sin embargo, como se dijo anteriormente, hasta ahora el módulo de control de tráfico de Linux TC no soporta capacidades de este tipo y está fuera del alcance de este proyecto.

4.2 IMPLEMENTACIÓN DEL PLANIFICADOR DE TRÁFICO HTB

A continuación se exponen los detalles de la implementación del planificador de tráfico para redes inalámbricas HTBi, éste está basado en el planificador de paquetes HTB por Martin Devera, incluida en el kernel estándar de Linux desde la versión 2.4.20 [1] y el *script* utilizado puede encontrarse en [2].

El principal fin del planificador es el de hacer un buen uso del ancho de banda disponible distribuyéndolo dinámicamente entre distintas clases de tráfico dependiendo de la prioridad asignada. La identificación del tráfico se hace, principalmente, con base en el puerto destino de la comunicación. A continuación en la figura 4.2 se observa el tratamiento que recibe cada paquete:

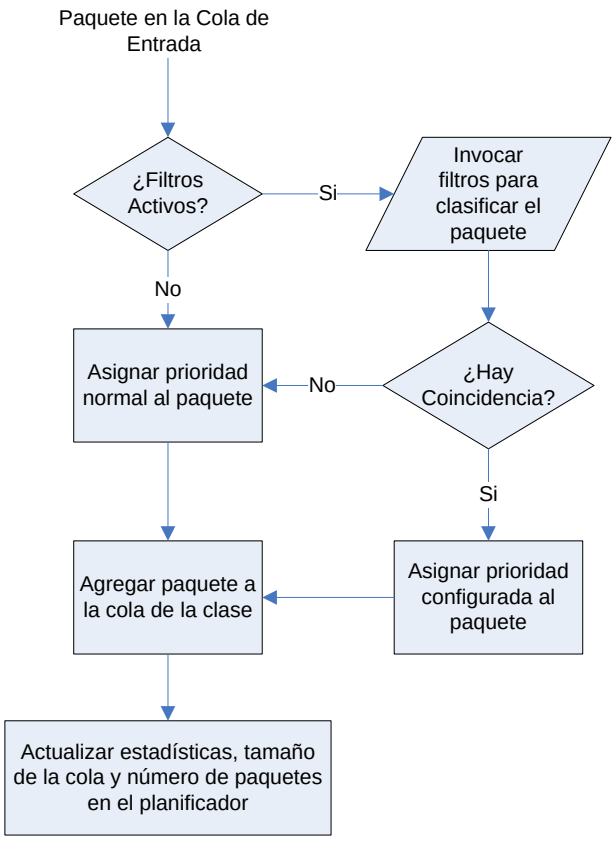

Figura 4.2 Diagrama de flujo del funcionamiento del planificador en las colas de entrada.

Las funciones más importantes del planificador de paquetes son la organización de paquetes a la entrada, función que es llamada cuando en alguna parte de la pila de protocolos de red se presenta un paquete y se ejecutan sobre él las funciones de conformación de tráfico y la organización de paquetes a la salida, función que es puesta en marcha si el dispositivo está listo para enviar el siguiente paquete. Las funciones a la entrada y salida son simples: su principal tarea es determinar la clase a la que corresponde cada paquete y actualizar el estado de la clase en el planificador como se ve en la figura 4.2.

4.2.1 El *script* de funcionamiento

El *script* que se expone en el ANEXO 1 se incorpora a la arquitectura de control de tráfico de este proyecto, la cual permite mejorar el desempeño de los enlaces 802.11, para soportar servicios que presenten un inadecuado funcionamiento cuando se le da tratamiento *best effort* a todo el tráfico. El control de tráfico utiliza una arquitectura jerárquica basada en HTB, que tiene la capacidad de gestionar dinámicamente el ancho de banda en enlaces inalámbricos punto a punto ó punto - multipunto.

Cuando se habla de gestionar dinámicamente el ancho de banda se refiere a la situación donde se le asigna a cada clase de tráfico determinados recursos de ancho de banda, los cuales pueden variar según la disponibilidad de los recursos asignados

a otras clases. El ancho de banda inicialmente destinado a una clase puede ser utilizado por otra mientras la primera no lo utilice.

Como se vio en detalle en la sección 3.6.3, al hacer la identificación del tráfico basándose en el puerto destino y utilizar marcas en la tabla *mangle* de *Iptables*, es posible diferenciar distintas clases de tráfico e implementar mecanismos de calidad de servicio (lo cual es requerido en las nuevas tecnologías de comunicación) sin la necesidad de hacer cambios a nivel MAC. De esta forma, se ofrece una optimización de los enlaces a un bajo costo.

De acuerdo a los objetivos de la planificación de paquetes, expuestos en la sección 1.2, las funcionalidades implementadas permiten mejorar el *throughput* para cada aplicación o tipo de tráfico, dándole un mejor comportamiento según sea su prioridad dentro de la red de área local. La ecuanimidad y eficiencia dependen de la configuración que determine el administrador de red. Con las variables de configuración del *script,* es posible darle un trato "justo" a las necesidades de cada usuario o entidad. Se implementaron 4 prioridades, de esta forma el tráfico puede ser: de máxima prioridad, de prioridad, tráfico normal y tráfico masivo), entre las que se puede distribuir a voluntad el tráfico existente, sin embargo, si se desea incluir una o más prioridades, es posible hacerlo y no implica mayor dificultad, de esta forma, el planificador de tráfico cumple con ser escalable y no resulta complejo, aun para quien no tiene experiencia con el código. En cuanto al consumo de energía, la implementación está orientada a dispositivos con capacidades de enrutamiento, que por lo general tienen una fuente de alimentación permanente; los dispositivos móviles no están contemplados (Teléfonos WiFi, Computadores Portátiles). El planificador es robusto en el sentido en que puede tolerar variaciones repentinas en los flujos de datos y proteger el ancho de banda garantizado para cada clase. Para desacoplar los retardos del ancho de banda disponible, es responsabilidad del administrador de red, establecer una configuración en la que se de la prioridad que cada tipo de tráfico merece, por ejemplo, al dejar todo el tráfico VoIP clasificado como máxima prioridad y las transferencias FTP en tráfico masivo, el segundo no interferirá para nada con el primero, ofreciendo disminución de los tiempos de respuesta del tráfico de mayor prioridad. El comportamiento de varios flujos de tráfico puede verse mejor en la sección 4.3.

El *script* está dividido en dos partes o archivos, la primera parte contiene los parámetros de configuración y la segunda las instrucciones de funcionamiento. El código del script comentado puede encontrarse en el ANEXO 1.

En la primera parte, **el archivo de configuración del planificador de tráfico**, por lo menos, debe dársele un valor a las variables "*UPLOAD*" y "*DOWNLOAD*". Configurarlas ligeramente por debajo del valor real del enlace es absolutamente necesario para obtener un buen desempeño en el control de ancho de banda. Cuando se configuran valores muy altos para los canales "*Upstream y Downstream*", las colas de tráfico en los extremos se llenan muy rápido y pueden ser muy largas, hablando de varios segundos, al llenarse no se podría hacer una conformación significativa del tráfico.

La configuración por defecto, con el parámetro apropiado en *UPLOAD*, debería ser la adecuada para la mayoría de situaciones en las que se requiere separar tráfico de baja prioridad como el de tipo P2P (*peer-to-peer)* o FTP, del tráfico interactivo como navegación web y sesiones SSH.

En general, la configuración del *script* de control de ancho de banda depende de diferentes variables. La mayoría de ellas deben llevar un espacio entre cada elemento de la lista (puertos, direcciones IP, protocolos). Adicionando un elemento a una lista, basándose en el nombre de la variable, aumentará o disminuirá la prioridad de una conexión entre las otras. La prioridad por defecto para cualquier tráfico es "normal", esto quiere decir que si un tipo de tráfico no se especifica en ninguna de las 3 prioridades (EXPR, PRIO ó BULK) estará clasificado dentro de esta prioridad. Un ejemplo de configuración de una variable de multiples entradas para clasificar tráfico es la línea:

```
TCP_PRIO="80 443"
```

Incluyendo esta línea en la configuración asegurará que todo el tráfico TCP en los puertos listados (en este caso para los protocolos: http, puerto 80, y HTTPS, puerto 443) sea tratado como tráfico de prioridad.

Otro ejemplo (de la configuración por defecto) donde se puede ver la configuración de un rango de puertos es:

```
TCP_BULK="1024: 21"
```

Esta línea adiciona el puerto 21 (el puerto para FTP) y todos los puertos desde el 1024 en adelante a la lista de conexiones que serán marcadas como tráfico '*bulk*'. El resultado es que todo el tráfico FTP tendrá la más baja prioridad, así mismo todo el tráfico presente en los puertos no reservados (por lo general tráfico de protocolos peer-to-peer). La notación "1024:" indica un rango de puertos, en este caso incluye todos los puertos incluyendo el 1024 y sucesivos. Otro ejemplo para definir un rango de puertos sería ":10", expresión que lo define desde el puerto 0 hasta el 10. Un rango desde el puerto 10 hasta el 20 se define de esta forma: "10:20". La utilización de rangos puede resultar muy útil en el caso de aplicaciones que utilizan puertos aleatorios, pero que estos se encuentran entre dos valores, como es el caso de aplicaciones de voz que utilizan protocolo RTP.

A lo largo del archivo de configuración se encontrarán varios segmentos, cada uno de estos representa una técnica de identificación del tráfico, según la cual se puede establecer la prioridad de determinado tipo de tráfico.

En este archivo también se encuentra la variable “QOS_IF” la cual sirve para especificar la interfaz sobre la que se realiza el control de tráfico, la cual puede ser, en el caso del Linksys WRT54GL, eth1 para la interfaz inalámbrica, vlan1 para la interfaz ethernet WAN ó vlan0 para los 4 puertos ethernet LAN, para otros modelos puede consultarse el listado de modelos compatibles con OpenWRT y la distribución de sus puertos, disbonible en [8]. Si la variable se encuentra comentada, se toma por defecto el puerto físico que esté asignado a la interaz WAN.

Por ultimo se encuentra la variable “UDP_LENGTH”, en ella se configura el tamaño máximo de paquetes UDP a los que se les da máxima prioridad, esto puede ser de gran utilidad en aplicaciones como voz, video o juegos en línea, si no se desea activar esta capacidad de identificación de tráfico simplemente se debe comentar la línea.

Continuando con la segunda parte, **el archivo de instrucciones de funcionamiento**, al principio del archivo se encuentran cuatro parámetros de configuración que no se incluyen en el archivo explicado previamente, se ponen en este archivo por que se

utilizan para una configuración avanzada y es mejor no experimentar variándolas, ya que una incorrecta configuración puede ocasionar un mal funcionamiento, “DBURST_U” y “DBURST_D”, son variables para la configuración de HTB, y varían el *burstiness* de la comunicación, el cual expresa la máxima duración permitida para “ráfagas” de tráfico en milisegundos, sin embargo, llevando su significado más allá de los efectos de su configuración sobre el planificador de tráfico, éste es el tamaño del *bucket* HTB para el tráfico en ms. Con él, el *script* calcula el número de bytes para el *buffer* basándose en la máxima velocidad de bajada. Por lo general, el algoritmo HTB se asegurará que se satisfaga la tasa de transferencia especificada en la línea de comando, pero también permite que existan pequeñas ráfagas de datos. Si se permitieran ráfagas de datos muy grandes es posible que la conformación del tráfico sea poco notable, sin embargo, es necesario permitirlas para evitar que las colas se saturen.

“MTU” es la máxima unidad de transferencia y “MIN_RATE” es el porcentaje mínimo del enlace que una clase podrá utilizar, esta variable se incorpora pensando en las clases de prioridad inferior. Cuando clases de prioridad superior estén activas y requieran de los recursos del enlace, el tráfico de la prioridad más baja no será descartado automáticamente, éste podrá usar como máximo, el porcentaje del enlace que tenga esta variable.

Ya en la parte de ejecución de sentencias, después de cargar todos los módulos necesarios, la primera medida es eliminar todas las disciplinas de encolamiento previamente existentes en la interfaz y se procede a definir las políticas de funcionamiento del planificador de tráfico, es en esta parte donde se define la jerarquía que funcionará sobre el enlace, para esto, el *script* utilizado crea una cola HTB madre con 4 clases, a la clase de mayor prioridad se le asigna el 50% del enlace, a la segunda el 20%, a la tercera el 20% y a la prioridad más baja el 10%, este porcentaje se debe interpretar como la porción del ancho de banda que cada clase utilizará si todas tienen tráfico activo, en otras palabras, es el ancho de banda que cada clase tiene garantizado.

```
    tc qdisc add dev $QOS_IF root handle 1: htb default 40 r2q $RTOQ_U
    tc class add dev $QOS_IF parent 1: classid 1:1 htb rate
${UPLOAD}kbit ceil ${UPLOAD}kbit burst $(($BURST_U*3)) cburst
$(($BURST_U*3)) mtu $MTU
    tc class add dev $QOS_IF parent 1:1 classid 1:10 htb rate
$(($UPLOAD*5/10))kbit ceil ${UPLOAD}kbit burst $(($BURST_U*1)) cburst
$(($BURST_U*1)) prio 1 mtu $MTU
    tc class add dev $QOS_IF parent 1:1 classid 1:20 htb rate
$(($UPLOAD*2/10))kbit ceil ${UPLOAD}kbit burst $(($BURST_U*1)) cburst
$(($BURST_U*1)) prio 2 mtu $MTU
    tc class add dev $QOS_IF parent 1:1 classid 1:30 htb rate
$(($UPLOAD*2/10))kbit ceil ${UPLOAD}kbit burst $(($BURST_U*1)) cburst
$(($BURST_U*1)) prio 3 mtu $MTU
    tc class add dev $QOS_IF parent 1:1 classid 1:40 htb rate
$(($UPLOAD*1/10))kbit ceil ${UPLOAD}kbit burst $(($BURST_U*1)) cburst
$(($BURST_U*1)) prio 4 mtu $MTU
```

A continuación, se añaden políticas SFQ, vistas previamente en la sección 3.6.6.1, a las clases hijas para hacerlas más equitativas y evitar que una bloquee a las demás.

```
tc qdisc add dev $QOS_IF parent 1:10 sfq quantum $MTU perturb 10
    tc qdisc add dev $QOS_IF parent 1:20 sfq quantum $MTU perturb 10
    tc qdisc add dev $QOS_IF parent 1:30 sfq quantum $MTU perturb 10
```

```
    tc qdisc add dev $QOS_IF parent 1:40 sfq quantum $MTU perturb 10
```

Después, de acuerdo al número de categorías en que se clasifica el tráfico se asigna a cada categoría una cola, en el caso de este proyecto se manejaron marcas numéricas, por esto, en esta parte se da la instrucción de que los paquetes con la marca “1” se dirijan a la cola 1:10, los que llevan la marca “2” a la cola 1:20, y asi sucesivamente.

```
tc filter add dev $QOS_IF parent 1: prio 1 protocol ip handle 1 fw
flowid 1:10
    tc filter add dev $QOS_IF parent 1: prio 2 protocol ip handle 2 fw
flowid 1:20
    tc filter add dev $QOS_IF parent 1: prio 3 protocol ip handle 3 fw
flowid 1:30
    tc filter add dev $QOS_IF parent 1: prio 4 protocol ip handle 4 fw
flowid 1:40
  }
```

De la misma forma se organiza la cola de ingreso IMQ, explicada en la sección 3.6.10, esta cumple la función de evitar que se sature la cola de salida HTB.

El *script* limita la tasa de transmisión de la clase de tráfico masivo o *bulk* al 80% del ancho de banda disponible. Esto asegura que siempre haya recursos para el tráfico de mayor prioridad. Sin embargo, teóricamente, es posible que el tráfico masivo ocupe todo el ancho de banda hasta que exista tráfico de mayor prioridad, en la práctica, los retardos del viaje de ida y vuelta en Internet, hacen que la respuesta a este cambio de condiciones no sea inmediata.

Se procede con la inicialización de las reglas de marcado de tráfico para la cola de salida, la marcación de los paquetes es de 1 a 4 dependiendo de las opciones configuradas por el usuario en la primera sección, cuando el puerto de una conexión no esté especificado en ninguna de las 3 categorías (“*bulk”* para tráfico masivo, “expr” para la máxima prioridad y “prio” para tráfico de prioridad), se asignará la marca de tráfico normal, así todo el tráfico que pase por la interfaz será marcado. Además, se le asigna automáticamente la máxima prioridad al tráfico ICMP, de forma que se disminuya la influencia de interferencias con otro tipo de tráfico en los resultados de las latencias, así mismo se procede para los paquetes ACK, de forma que la velocidad de subida no limite la velocidad de bajada.

Para mejorar el filtrado del tráfico también se da la posibilidad de filtrar por IP y puerto destino, por ejemplo, para incluir el soporte para VoIP, se debe adicionar la dirección IP del servidor o adaptador VoIP a la variable "IP_EXPR" (por ejemplo: IP_EXPR="192.168.1.10"). De esta forma, se le da máxima prioridad a todo el tráfico que se dirige hacia y desde el adaptador VoIP.

También se configura el soporte para filtros capa 7 [3], con estos filtros el planificador examina coincidencias entre la cabecera de la trama y archivos “.pat” que se guardan en la carpeta “/etc/l7-protocols”, cada uno de estos archivos especifica características de un protocolo, de esta forma, se puede identificar tráfico independientemente del puerto que utilice basándose en la información de la capa de aplicación. Una lista de los protocolos soportados y sus archivos “.pat” puede encontrarse en [4].

Existen otro tipo de filtros similares a los filtros capa 7 que también se incorporan, estos son los filtros ipp2p [5], los cuales están orientados a identificar protocolos del

tipo “peer-to-peer”, el planificador de tráfico también puede identificar paquetes haciendo uso de ellos.

Otra manera de identificar el tipo de conexiones es por el tamaño de los paquetes. Se tiene la posibilidad que los paquetes pequeños UDP puedan tener alta prioridad, usualmente, el tráfico interactivo como juegos y voz IP se benefician de esto, el tamaño de los paquetes se varía usando la variable “UDP_LENGTH”.

A nivel IP existen dos tipos de marcas que también se pueden reconocer en la trama, estas son las banderas en los campos TOS (Type of Service) y DSCP (Diffserv Codepoint), para los valores que pueden tomar las variables se pueden ver introduciendo el siguiente comando para el campo TOS:

```
>iptables --match tos --help
```

Y para DSCP:

```
>iptables --match dscp --help
```

Es importante notar que algunas variables de filtrado tienen preferencia sobre otras. Esto es muy significativo en casos en los que el mismo tráfico es identificado por diferentes reglas. Un ejemplo sería poner un puerto UDP por encima de 1024 que es utilizado por una aplicación específica a la lista de puertos con máxima prioridad. En la configuración por defecto, todos los puertos altos (1024:) están incluidos en la variable UDP_BULK. Sin embargo, el tráfico a dicho puerto será clasificado como prioritario debido a que la variable "UDP_EXPR" tiene preferencia sobre "UDP_BULK".

El orden jerárquico de las variables, en orden de menor a mayor, es el siguiente:

- L7_BULK
- L7_PRIO
- L7_EXPR
- IPP2P_BULK
- IPP2P_PRIO
- IPP2P_EXPR
- TCP_BULK
- UDP_BULK
- TCP_PRIO
- UDP_PRIO
- TCP_EXPR
- UDP_EXPR
- TOS_BULK
- TOS_PRIO
- TOS_EXPR
- DSCP_BULK
- DSCP_PRIO
- DSCP_EXPR
- IP_BULK
- IP_PRIO
- IP_EXPR

Cabe recalcar que el *script* solo ejerce control completo sobre el canal de subida del enlace, para tener un enlace punto a punto o punto multipunto con una conformación del tráfico debidamente hecha es necesario implementar HTBi en los extremos de transición LAN-WLAN y WLAN-LAN como lo ilustra la figura 4.3.

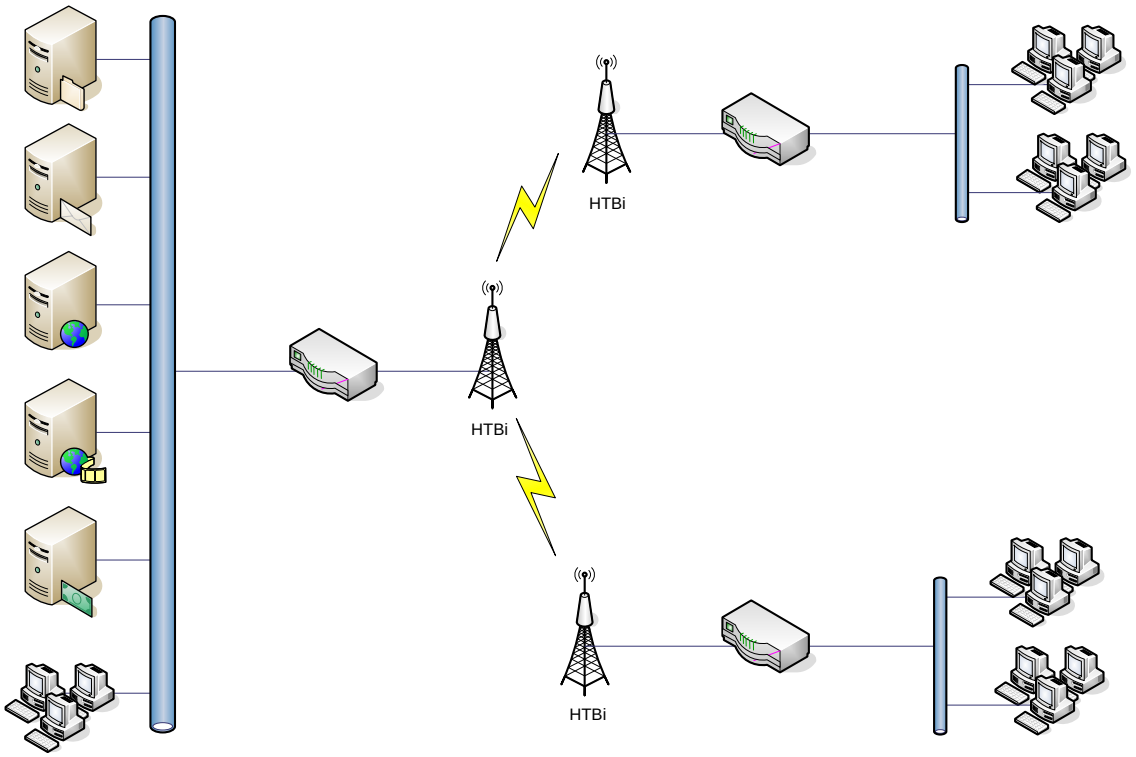

Figura 4.3 Escenario de funcionamiento HTBi

Se utilizan 3 *scripts* adicionales, para iniciar, parar y revisar el estado del planificador de tráfico, los cuales se ubican en el directorio /usr/bin, de esta forma, sin importar la ruta del sistema de archivos donde se encuentre, se pueden realizar estas tres tareas, el nombre que se le da a los archivos es indiferente, el código de estos *scripts* se encuentran al final del ANEXO 1 después de los códigos de funcionamiento y configuración.

4.2.2 Implementación en puntos de acceso inalámbricos

La implementación en los enrutadores inalámbricos tiene más requisitos que una instalación en una distribución Linux para equipos de escritorio, se deben cumplir las condiciones que a continuación se listan:

- Para poder ejercer control sobre el tráfico, éste debe pasar de una interfaz de red a otra, en el caso particular, el tráfico pasa de la interfaz LAN a la WAN en un extremo y en el otro de la WAN a la LAN.

- Es necesario cambiar el *firmware* del equipo, ya que sin importar el fabricante, el original carece de muchas funcionalidades necesarias.

- No a todos los enrutadores inalámbricos se les puede hacer cambio del *firmware*, por lo que hay que seleccionar el hardware cuidadosamente.

- Es necesario instalar paquetes adicionales a los que contiene el *firmware,* para poder ejecutar las instrucciones del *script*.

La fase de implementación en enrutadores inalámbricos se llevó a cabo teniendo como base OpenWRT, que es la distribución para ésta clase de dispositivos con menos restricciones de uso entre las de su tipo, y además, había sido utilizada anteriormente en otro proyecto [6] para redes inalámbricas realizado en el marco del área de Comunicaciones Inalámbricas del GNTT, el cuál trabajó con el análisis y pruebas de desempeño en dispositivos AP, obteniendo excelentes resultados.

4.2.2.1 OpenWRT

OpenWRT toma una ruta diferente, en lugar de empezar con las fuentes de los fabricantes de hardware, el desarrollo comenzó desde cero. Pieza por pieza de software fue agregada para dar funcionalidad al *firmware*, usando las versiones más recientes disponibles. Lo que hace OpenWRT realmente único es el hecho de que emplea un sistema de archivos modificable ya que el *firmware* es una recopilación estática de software pero puede ajustarse para encajar en las necesidades particulares de cada situación.

En una distribución Linux como OpenWRT se tiene el poder para hacer lo que se necesite con hardware barato evitando software propietario. En el momento la distribución contiene más de 100 paquetes software. Además, la comunidad de OpenWRT proporciona más paquetes que se pueden adicionar. Para desarrolladores, el proyecto proporciona un sistema robusto, que puede usarse para hacer modificaciones en la fuente del *firmware*. El manejo de nuevos paquetes software se simplifica con el uso del SDK.

Para el desarrollo del proyecto se usó la versión OpenWRT *White Russian* 0.9 basado en el Kernel de Linux 2.4.30, la cual es la última versión a la fecha.

4.2.2.2 Enrutador de Banda Ancha Linksys WRT54GL

De acuerdo con la lista de compatibilidad de hardware de OpenWRT [7] se decidió utilizar el enrutador de banda ancha *Linksys WRT54GL* por ser más asequible dado que los utilizados tuvieron que ser adquiridos con recursos propios, y por no tener conflicto por la versión del hardware. Muchos equipos similares, como el WRT54G, que solo se puede modificar su *firmware* en versiones de hardware 4 e inferiores [7], presentan dificultades en versiones recientes.

El Linksys WRT54GL, en la figura 4.4, es un equipo de calidad, es único entre los dispositivos de consumo doméstico debido a que los desarrolladores de Linksys tuvieron que liberar el código fuente del *firmware* del enrutador para cumplir con las obligaciones de la GNU GPL. Este hecho permite a los entusiastas de la programación

modificar el *firmware* para añadir o cambiar funcionalidades del dispositivo. Existen varios proyectos de desarrollo que proveen versiones mejoradas del *firmware* para la serie WRT54GX.

Figura 4.4 Enrutador de banda ancha Linksys WRT54GL

El WRT54GL está equipado con un procesador MIPS a 200 MHz con 16 megabytes de memoria RAM y 4 megabytes de memoria flash para almacenar el *firmware*. Todos los modelos vienen con un *switch* de 5 puertos (el puerto para Internet está en el mismo *switch* pero en una VLAN diferente) y con un *chip* controlador inalámbrico de Broadcom. El WRT54GL tiene dos antenas externas conectadas al dispositivo a través de conectores de polaridad inversa.

4.2.2.3 Instalación del Planificador de Tráfico

Después de haber accedido por consola al enrutador con OpenWRT, se instalan los paquetes necesarios para interpretar y ejecutar los comandos del *script*, los cuales son:

- tc → Utilidad de control de tráfico IpRoute2
- iptables → Software Netfilter para cortafuegos en IPv4
- kmod-sched → Módulos adicionales del kernel para control de Tráfico
- kmod-ipt-conntrack → Módulos Netfilter para seguimiento de conexiones
- iptables-mod-conntrack → Extensión Iptables para seguimiento de conexiones
- kmod-ipt-ipopt → Módulos de comparación y cambio de atributos de paquetes IP
- iptables-mod-ipopt → Extensión Iptables para comparación y cambio de paquetes IP
- kmod-ipt-extra → Otros módulos kernel Netfilter (IPv4)
- iptables-mod-extra → Otras extensiones Iptables

- ip → Análogo al paquete IProute en otras distribuciones Linux
- kmod-imq → Módulos kernel para el soporte de IMQ
- iptables-mod-imq → Extensiones Iptables para el soporte IMQ
- kmod-ipt-filter → Módulos kernel para inspección del contenido de los paquetes (soporte para ipp2p y filtros l7)
- iptables-mod-filter → Extensiones Iptables para filtrado de paquetes e inspección del contenido.

La instalación de estos paquetes puede hacerse mediante la ejecución del siguiente comando:

```
>ipkg install tc iptables kmod-sched kmod-ipt-conntrack
iptables-mod-conntrack kmod-ipt-ipopt iptables-mod-ipopt
iptables-mod-extra ip kmod-imq iptables-mod-imq kmod-ipt-filter
iptables-mod-filter kmod-ipt-extra
```

El editor con el que se cuenta en OpenWRT es "VI", en él se copia el *script* del ANEXO 1, se guarda en un archivo y luego se le asigna permiso de ejecución con el comando:

```
>chmod +x nombredelarchivo
```

Se recomienda guardar el archivo en la ruta "`/etc/hotplug.d/iface/`", si la ruta no existe, se deben crear los directorios faltantes. En el directorio "`hotplug.d`" se almacenan todos los *script* que inicializan los diferentes componentes hardware del enrutador, de esta forma, el *script* iniciará automáticamente cada vez que se encienda el equipo. En OpenWrt, en directorios como "`/etc/init.d`" o "`etc/hotplug.d`" se almacenan *scripts* que se ejecutan cuando el equipo es encendido, el primero se relaciona con los servicios y el segundo con los dispositivos hardware soportados. Para darle un orden secuencial a la ejecución de estos *scripts* se utiliza un prefijo numérico para el nombre de los archivos, por ejemplo para el caso del planificador de tráfico se utiliza el nombre "`20-qos`", si se usa otro nombre se deben cambiar las referencias en el archivo de configuración.

Así mismo para el archivo de configuración, se guarda en la ruta "`/etc`", el nombre de archivo utilizado es "`qos.conf`".

El control de tráfico en la mayoría de los casos se hace sobre la interfaz asociada al puerto WAN, para saber cuál es ésta, se ejecuta el comando:

```
>nvram get wan_ifname
```

La salida de este comando será el que se asigna a la variable "`QOS_IF`" en la configuración del *script.* De no asignarle ningún valor a la variable el planificador de tráfico trabajará sobre la interfaz WAN configurada por defecto.

Para la ejecución del *script* se usa:

>./*nombredelarchivo*

Para revisar el estado de las colas:

```
>tc class show dev eth0
```

Para detener el *script* se introducen los siguientes comandos:

```
>iptables -t mangle -F
>iptables -t mangle -X
>tc qdisc del dev eth0 root
```

Donde se supone que "`eth0`" es la interfaz sobre la que se está haciendo el control de tráfico. También es posible incluir estas tres líneas en un archivo de texto plano y darle permisos de ejecución, así funcionaría como un *script* para detener el planificador de tráfico.

En ese punto, se tiene instalado el planificador de tráfico y está listo para usarse, sin embargo, la configuración del enlace es muy importante, mientras se esté enrutando el tráfico que pasa entre dos interfaces, éste deberá funcionar correctamente. En la siguiente sección se muestra una configuración utilizada.

4.2.3 Configuración del Enlace Inalámbrico

En el marco de desarrollo del proyecto se contemplan dos escenarios generales de implementación: enlaces punto a punto y punto - multipunto. Para comprobar el funcionamiento de la gestión dinámica de ancho de banda y llevar a cabo la fase de pruebas se implementó el enlace punto a punto, la configuración de un enlace punto – multipunto se plantea en esta sección al igual que las consideraciones necesarias para la configuración del *script* en este tipo de escenarios.

Las dos configuraciones presentadas a continuación utilizan enrutamiento estático, sin embargo, con las capacidades de OpenWRT y la instalación de paquetes adicionales es posible utilizar protocolos de enrutamiento dinámico como RIP (*Routing Information Protocol* o Protocolo de Información de Enrutamiento), OSPF (*Open Shortest Path First* o "Primero el camino abierto mas corto") o BGP (*Border Gateway Protocol* o Protocolo de *Gateway* de Frontera), incluso es posible soportar estos protocolos de enrutamiento con IPv6, que pueden ser necesarios para topologías de red más complejas.

4.2.3.1 Enlace Punto a Punto

A continuación se dan las indicaciones para configurar un enlace inalámbrico entre dos subredes, esta configuración es aplicable al caso de un enlace empresarial en el que se quieran conectar dos sedes y que los equipos tengan conectividad en ambos sentidos. Este escenario es el que se utilizó para realizar las pruebas que se verán más adelante.

En la figura 4.5 se observa la topología de red implementada.

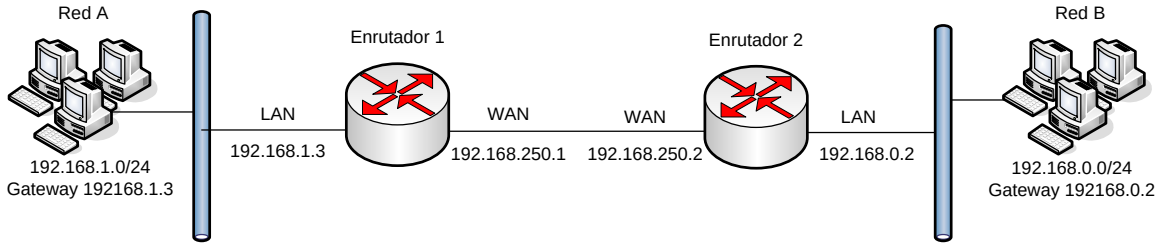

Figura 4.5 Topología del enlace punto a punto

Para limitar el tráfico de *broadcast* se configuran dos subredes, una a cada lado, (192.168.0.0/24 y 192.168.1.0/24), y el enlace WAN de los enrutadores está configurado como una tercera subred (192.168.250.0/30) en la que solo se habilitan 2 direcciones IP para cada uno de los enrutadores.

Antes de empezar a configurar los equipos, es necesario instalar el paquete “IP”, si es que todavía no se lo tiene instalado, este paquete, entre otras cosas, permite adicionar reglas de enrutamiento. Para instalarlo, se ejecuta en una consola el siguiente comando:

```
>ipkg install ip
```

La configuración de los equipos se puede hacer por la interfaz web o por una consola SSH, se prefiere la consola SSH ya que por web no se pueden modificar todas las variables.

El comando para mostrar el valor de una variable es:

```
>nvram get nombre_de_la_variable
```

Para modificar el valor de una variable se usa:

```
>nvram set nombre_de_la_variable=valor_de_la_variable
```

Después de hacer cambios en las variables del sistema es necesario hacer que éste cargue las modificaciones mediante el comando:

```
>nvram commit
```

Para el enlace se necesita asociar la interfaz inalámbrica al puerto WAN en los dos enrutadores, en el caso del Linksys WRT54GL, la interfaz inalámbrica es “eth1” (para otros modelos puede variar [8]). Para asignarla al puerto WAN se introduce el siguiente comando:

```
>nvram set wan_ifname=eth1
```

Se debe tener un enrutador funcionando en modo máster o punto de acceso, y otro funcionando como cliente enrutado.

Tomando como referencia la figura 4.6, para el enrutador 1, en la implementación se tiene:

```
lan_dns=                        #DNS LAN
lan_gateway=                    #Puerta de Enlace LAN
lan_ifname=br0                  #Nombre de la Interfaz LAN
lan_ifnames=vlan0 eth2          #Interfaces asociadas al Puerto LAN
lan_ipaddr=192.168.1.3          #Dirección IP del Puerto LAN
lan_netmask=255.255.255.0       #Mascara de Red LAN
lan_proto=static                #Configuración LAN Estática
wan_dns=                        #DNS WAN
wan_gateway=192.168.250.2       #Puerta de Enlace WAN
wan_hostname=Sarge              #Nombre del Equipo
wan_ifname=eth1                 #Interfaz Asociada al puerto WAN
wl0_ifname=eth1                 #Nombre de la Interfaz Inalámbrica
wan_ipaddr=192.168.250.1        #Dirección IP del Puerto WAN
wan_proto=static                #Configuración WAN Estática
wan_netmask=255.255.255.252     #Mascara de red WAN
wl0_mode=ap                     #Modo de operación Master
wl0_radio=1                     #Tráfico inalámbrico Activado
wl0_ssid=0ptimusNET             #SSID del Enlace
```

El enrutador 2, tiene la siguiente configuración:

```
lan_dns=
lan_gateway=
lan_ifname=br0
lan_ifnames=vlan0 eth2
lan_ipaddr=192.168.0.2
lan_netmask=255.255.255.0
lan_proto=static
wan_dns=
wan_gateway=
wan_hostname=Xaero
wan_ifname=eth1
wan_ipaddr=192.168.250.2
wan_netmask=255.255.255.252
wan_proto=static
wl0_ifname=eth1
wl0_mode=sta                    #Modo de operación Cliente Enrutado
wl0_radio=1
wl0_ssid=0ptimusNET
```

En el enrutador 2 es necesario adicionar una ruta estática hacia la red A (192.168.0.0/24) ya que no se configuró ninguna puerta de enlace. Esto se hace adicionando al archivo "`/etc/init.d/ S95custom-user-startup`" la siguiente línea:

```
/usr/sbin/ip route add 192.168.1.0/24 via 192.168.250.1 dev eth1
```

Si el archivo no existe, se debe crearlo y darle permisos de ejecución (`chmod +x /etc/init.d/ S95custom-user-startup`).

La configuración por defecto del *firewall* de OpenWRT, debe ser completamente desactivada, para esto se le quita el permiso de ejecución al archivo "`/etc/init.d/S35firewall`", este procedimiento debe hacerse en los dos enrutadores y se logra mediante el siguiente comando:

```
>chmod -x /etc/init.d/S35firewall
```

Finalmente, para que se logre una conectividad completa los equipos en la red A deben tener configurada como puerta de enlace la dirección IP del enrutador 1 (192.168.1.3), y los equipos de la red B, la dirección IP del enrutador 2 (192.168.0.2).

La configuración puede verificarse introduciendo en la consola de cada uno de los puntos de acceso el siguiente comando:

```
>ip route show
```

La salida de este comando en los enrutadores es como sigue:

```
root@Sarge:~# ip route show
192.168.250.0/30 dev eth1  proto kernel  scope link  src 192.168.250.1
192.168.1.0/24 dev br0  proto kernel  scope link  src 192.168.1.3
default via 192.168.250.2 dev eth1
root@Sarge:~#

root@Xaero:/# ip route show
192.168.250.0/30 dev eth1  proto kernel  scope link  src 192.168.250.2
192.168.1.0/24 via 192.168.250.1 dev eth1
192.168.0.0/24 dev br0  proto kernel  scope link  src 192.168.0.2
default via 192.168.250.1 dev eth1
root@Xaero:/#
```

Una síntesis de la configuración del enlace se muestra en la figura 4.6.

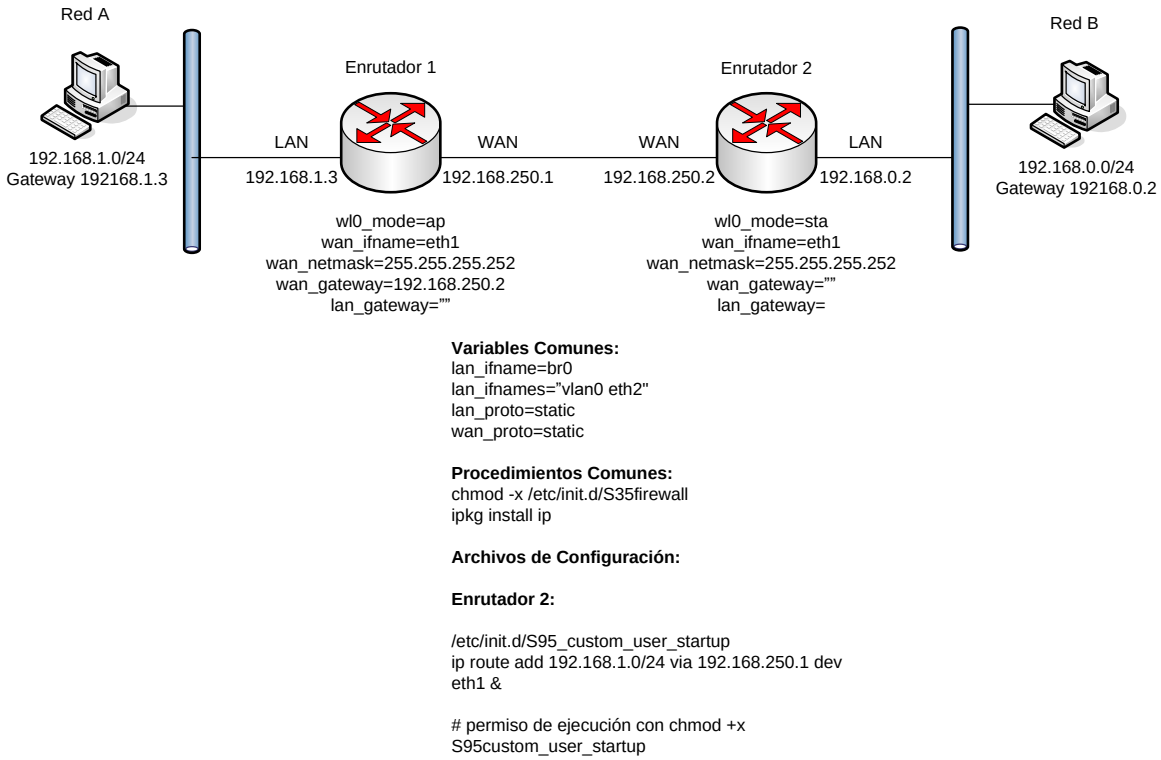

Figura 4.6 Configuración del enlace punto a punto

4.2.3.2 Enlace Punto - Multipunto

Una configuración básica de punto multipunto se presenta en la figura 4.7.

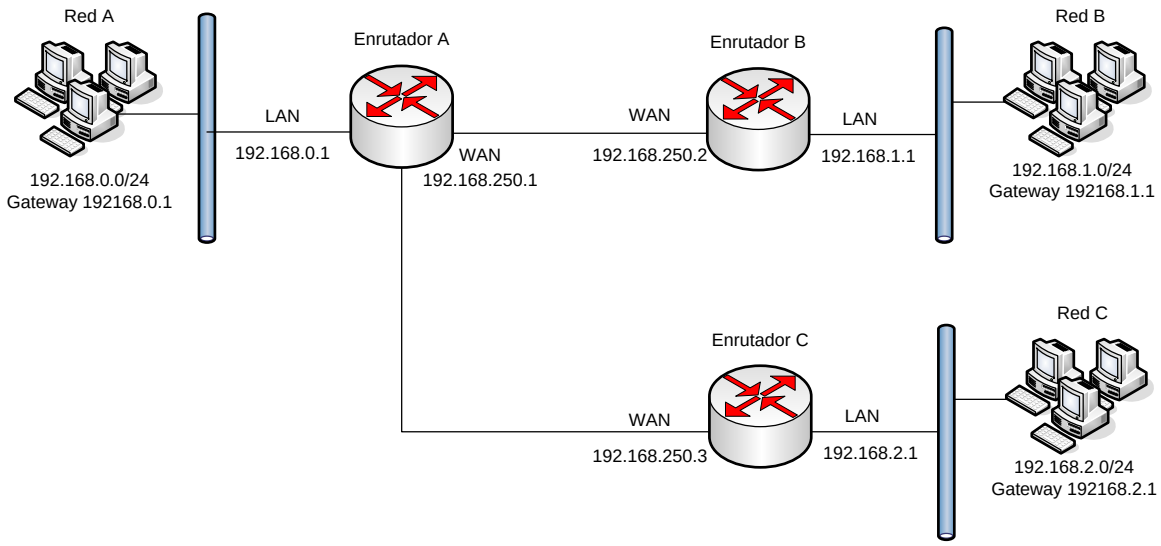

Figura 4.7 Topología de enlace punto - multipunto

Al igual que en una configuración punto a punto se configuran 3 subredes para limitar el tráfico de broadcast (192.168.0.0/24, 192.168.1.0/24 y 192.168.2.0/24), para la subred del enlace WAN se habilitan 6 direcciones ip en la configuración de la máscara de subred (192.168.250.0/29)

El proceso de instalación de paquetes y configuración de los enrutadores es idéntico al procedimiento seguido en la configuración del enlace punto apunto, visto en la sección anterior.

Siguiendo el esquema de la figura 4.7, se debe configurar un enrutador en modo máster y los otros dos como cliente enrutado. Los dos clientes (enrutadores C y B) deben tener configurado como *gateway* al enrutador A, mientras que en el enrutador A se deben poner rutas estáticas hacia las dos subredes, las variables de configuración y otros detalles se muestran a continuación.

Variables de configuración para el enrutador A:

```
lan_dns=
lan_gateway=
lan_ifname=br0
lan_ifnames=vlan0 eth2
lan_ipaddr=192.168.0.1
lan_netmask=255.255.255.0
lan_proto=static
wan_dns=
wan_gateway=
wan_hostname=Sarge
wan_ifname=eth1
wl0_ifname=eth1
```

```
wan_ipaddr=192.168.250.1
wan_proto=static
wan_netmask=255.255.255.248
wl0_mode=ap
wl0_radio=1
wl0_ssid=0ptimusNET
```

Variables de configuración para el enrutador B:

```
lan_dns=
lan_gateway=
lan_ifname=br0
lan_ifnames=vlan0 eth2
lan_ipaddr=192.168.1.1
lan_netmask=255.255.255.0
lan_proto=static
wan_dns=
wan_gateway=192.168.250.1
wan_hostname=Sarge
wan_ifname=eth1
wl0_ifname=eth1
wan_ipaddr=192.168.250.2
wan_proto=static
wan_netmask=255.255.255.248
wl0_mode=sta
wl0_radio=1
wl0_ssid=0ptimusNET
```

Variables de configuración para el enrutador C:

```
lan_dns=
lan_gateway=
lan_ifname=br0
lan_ifnames=vlan0 eth2
lan_ipaddr=192.168.2.1
lan_netmask=255.255.255.0
lan_proto=static
wan_dns=
wan_gateway=192.168.250.1
wan_hostname=Sarge
wan_ifname=eth1
wl0_ifname=eth1
wan_ipaddr=192.168.250.3
wan_proto=static
wan_netmask=255.255.255.248
wl0_mode=sta
wl0_radio=1
wl0_ssid=0ptimusNET
```

En el enrutador A no se configuró una puerta de enlace ya que haí el tráfico no tiene una única opción de enrutamiento, por esto se adicionan dos rutas estáticas hacia las redes B y C. Estas dos rutas estáticas se adicionan en el archivo "`/etc/init.d/S95custom-user-startup`" por medio de las siguientes líneas:

```
/usr/sbin/ip route add 192.168.1.0/24 via 192.168.250.2 dev eth1
/usr/sbin/ip route add 192.168.2.0/24 via 192.168.250.3 dev eth1
```

De no existir el archivo se debe crear y darle permisos de ejecución como se indicó en la sección anterior.

Para la configuración del control de tráfico en un enlace punto - multipunto se debe tener en cuenta la distribución del ancho de banda entre los enlaces, de acuerdo a esto se configuran la variables “UPLOAD” y “DOWNLOAD”. El valor configurado en la variable “DOWNLOAD” en el enrutador configurado como maestro corresponde a la sumatoria de las variables “UPLOAD” en los enrutadores que funcionan como cliente enrutados, así mismo, la variable “UPLOAD” en el maestro corresponde a la sumatoria de las variables “DOWNLOAD” en los clientes.

4.3 PRUEBAS Y RESULTADOS

Para las pruebas de la implementación del planificador de tráfico se utilizó el montaje que se muestra en la figura 4.8. En el PC1 se reciben varios tipos de flujos de datos, y se generan gráficas del comportamiento de los mismos. Como herramienta de monitoreo de tráfico se utilizó el PRTG *Traffic Grapher,* de la casa de *software* Paessler [9]. En los puntos de acceso o enrutadores inalámbricos se corrió el *script* del ANEXO 1 limitando el enlace a 1 Mbps, en el PC2 fueron instalados varios servicios (HTTP, HTTPS, FTP, SSH), los cuales eran accedidos desde el PC1, la prioridad de dichos servicios en la configuración del planificador de tráfico se debe configurar según el criterio del administrador y de las características de la red, la prioridad asignada para las pruebas, puede o no corresponder a la realidad, dependiendo de cada caso particular y del escenario de aplicación de los mecanismos de control de tráfico. Otra aclaración que vale la pena hacer es que se usaron estos servicios únicamente con propósitos experimentales, por ser de los más populares en Internet, el tráfico real de una conexión es más heterogéneo, sin embargo, al mostrar pocos flujos se pueden ver más claramente las acciones del planificador de tráfico.

Para todas las pruebas se utilizó en el *script* una asignación del ancho de banda del 50% del enlace para el tráfico de máxima prioridad, 20% para el tráfico de prioridad, 20% para el tráfico normal y 10% para el tráfico masivo o de prioridad mínima.

Los enrutadores establecían el enlace que se describió en la sección 4.2.3.1.

Figura 4.8 Montaje de prueba.

En primera instancia se examina el comportamiento del ancho de banda al compartirse entre algunos tipos de tráfico sin ningún mecanismo de conformación de tráfico, es lo que se muestra en la figura 4.9.

La prueba consistió en variar los flujos de información como se indica a continuación:

1. Se inicia una transferencia de archivos por HTTP
2. Se inicia una transferencia de archivos por SSH
3. Se inicia una transferencia de archivos por FTP
4. Se detiene el flujo de datos HTTP
5. Se detiene el flujo de datos SSH
6. Se inicia una descarga de archivos por HTTP usando un gestor de descargas, aplicaciones caracterizadas por utilizar varias conexiones para transferir un mismo archivo y usar otras técnicas para tomar el mayor ancho de banda posible, lo que causa mayor dificultad para limitar los recursos que éstas utilizan, para tal propósito se utilizó *Flash Get* [10].
7. Se inicia nuevamente una transferencia de archivos usando SSH
8. Se detiene el flujo de datos FTP
9. Se detienen los flujos restantes.

El inicio y el final de cada flujo de datos se puede apreciar en las gráficas por los cambios en las pendientes de cada línea. Para darle una referencia temporal al lector, las variaciones en los flujos de tráfico se realizaron aproximadamente cada 30 segundos.

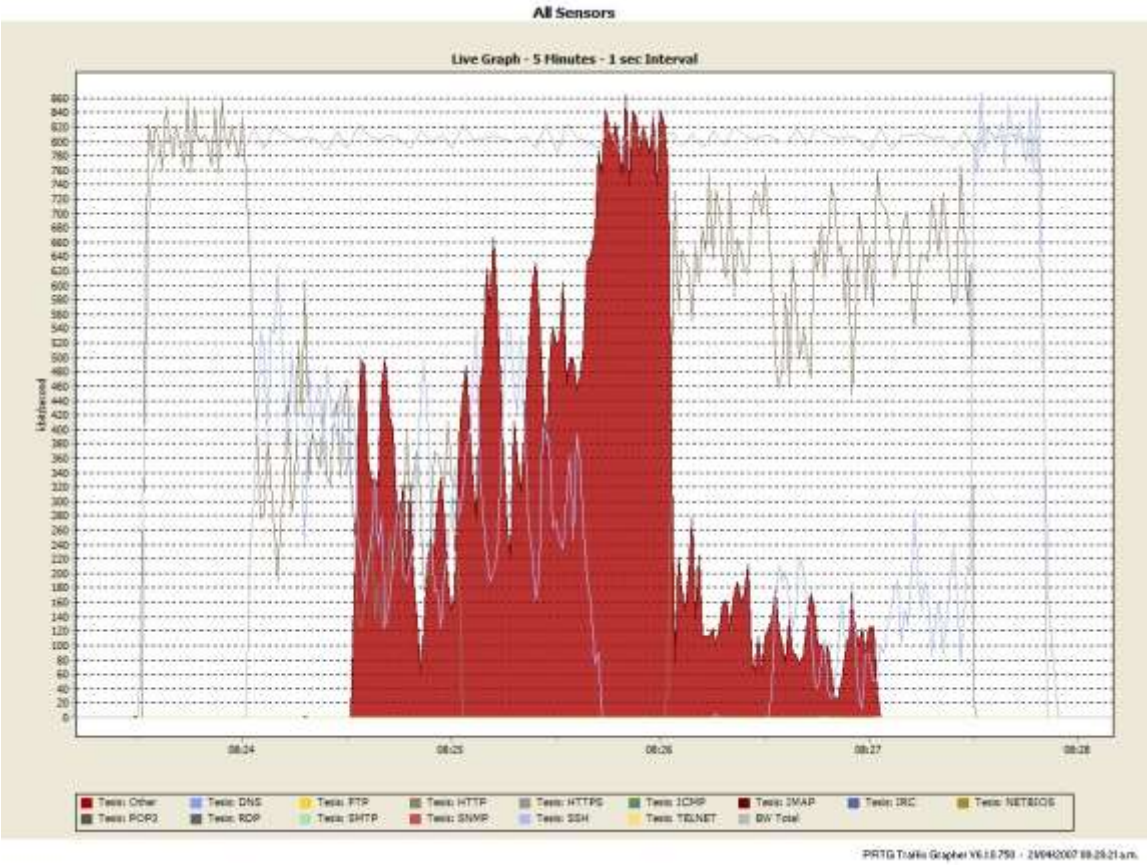

Figura 4.9 Comportamiento del tráfico sin ningún mecanismo de control de tráfico.

Como se puede ver, en el principio, la transferencia de datos HTTP ocupa todo el ancho de banda disponible, cuando inicia la transferencia por SSH hay una competencia entre los dos flujos existentes y los recursos se reparten muy irregularmente, situación que se mantiene cuando inicia el tercer flujo de datos. Sólo

después de que se detienen las transferencias HTTP y SSH, la gráfica muestra un comportamiento regular. Cuando se inicia una descarga usando *Flash Get* la transferencia FTP cae inmediatamente y el tráfico HTTP ocupa casi toda la capacidad del enlace, ni siquiera cuando se inicia el flujo de datos por SSH hay un cambio notable.

Cuando no hay mecanismos que regulen de alguna manera el tráfico, las diferentes conexiones mantienen una constante competencia por obtener los recursos del enlace, lo que hace que las velocidades de transmisión sean muy irregulares, este comportamiento, por lo general ocasiona una mala experiencia para el usuario.

En la siguiente prueba se activó el planificador de tráfico pero sin hacer asignación de prioridades, simplemente, todos los tipos de tráfico se tratan igual dentro de una cola SFQ, la cual se explicó en la sección 3.6.6.1, de esta forma, ningún tipo de tráfico bloquea a otro y el enlace se reparte equitativamente entre las conexiones activas. Situación que se puede ver en la figura 4.10, donde la variación de las conexiones establecidas es la misma que en la prueba anterior.

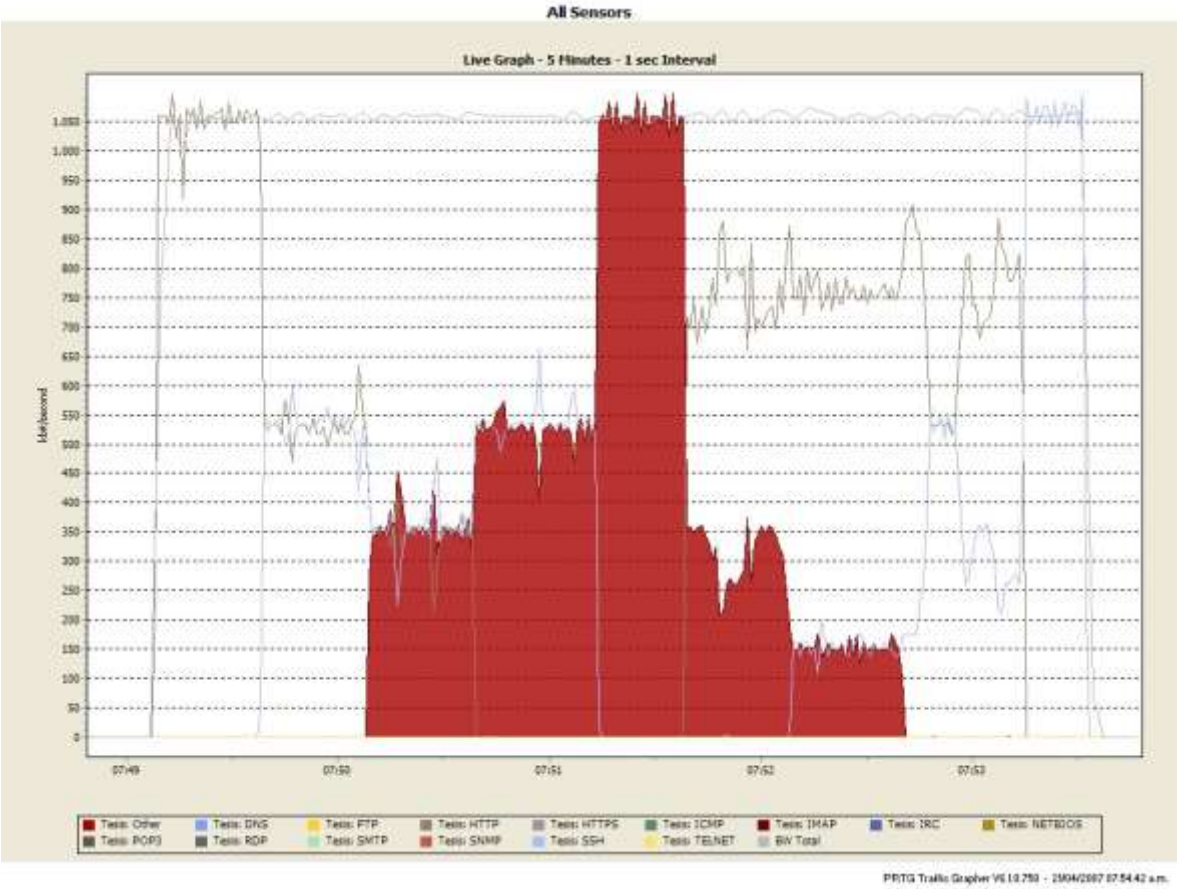

Figura 4.10 Efectos del planificador de tráfico sin asignación de prioridades.

Como se puede ver en la figura 4.10, la incorporación de una sola cola le da un mejor comportamiento al enlace cada vez que cambian las condiciones en él, puede notarse que la repartición equitativa del ancho de banda es efectiva y las “ráfagas” de tráfico son menos pronunciadas, sin embargo, aún no es posible restringir el uso de los recursos demandado por el gestor de descargas, ya que al crear varias conexiones para transferir un mismo archivo, cada conexión de estas participa en la repartición del

ancho de banda. Ante esta situación en particular, la implementación de clases y prioridades se hace necesaria.

Para la siguiente prueba se asignó máxima prioridad al tráfico SSH, y se configuró el tráfico FTP como masivo. Las conexiones HTTP no se asignaron a ninguna clase, sin embargo, según las características de funcionamiento del algoritmo, éste será asignado a la prioridad normal, donde va todo el tráfico no incluido en las variables que definen los puertos destino asignados a cada clase.

En la figura 4.11 pueden verse claramente reflejadas las políticas de funcionamiento del control de tráfico. En un principio la transferencia de datos HTTP ocupa todo el enlace, al iniciar el flujo SSH (el cual tiene máxima prioridad), el tráfico HTTP cae hasta utilizar solamente el 20% del enlace, valor que corresponde al porcentaje garantizado a este tipo de tráfico cuando el enlace está totalmente ocupado. Cuando Inicia la transferencia por FTP (protocolo que tiene la prioridad más baja), sólo puede utilizar el 10% del enlace, el cual es restado del ancho de banda usado por la transferencia SSH, ya que el tráfico HTTP ya está en el mínimo permitido. Cuando se detiene este último flujo, la transferencia SSH toma el ancho de banda que deja libre, sin embargo, cuando se detiene, el tráfico FTP solo sube hasta un 80%, y el 20% restante queda libre, después se inicia la transferencia usando el gestor de descarga y esta toma un 90% de la conexión, al iniciar la transferencia SSH ésta utiliza el 70%, HTTP cae hasta el 20% nuevamente y cuando se termina la transferencia FTP el flujo SSH incrementa su tasa de descarga.

Los resultados en la figura 4.11 indican que el algoritmo funciona como se esperaba y que los mecanismos implementados son suficientes para ejercer control sobre las conexiones establecidas. El gestor de descargas, el cual había sido un problema en el caso anterior, funciona según las políticas especificadas en el *script*.

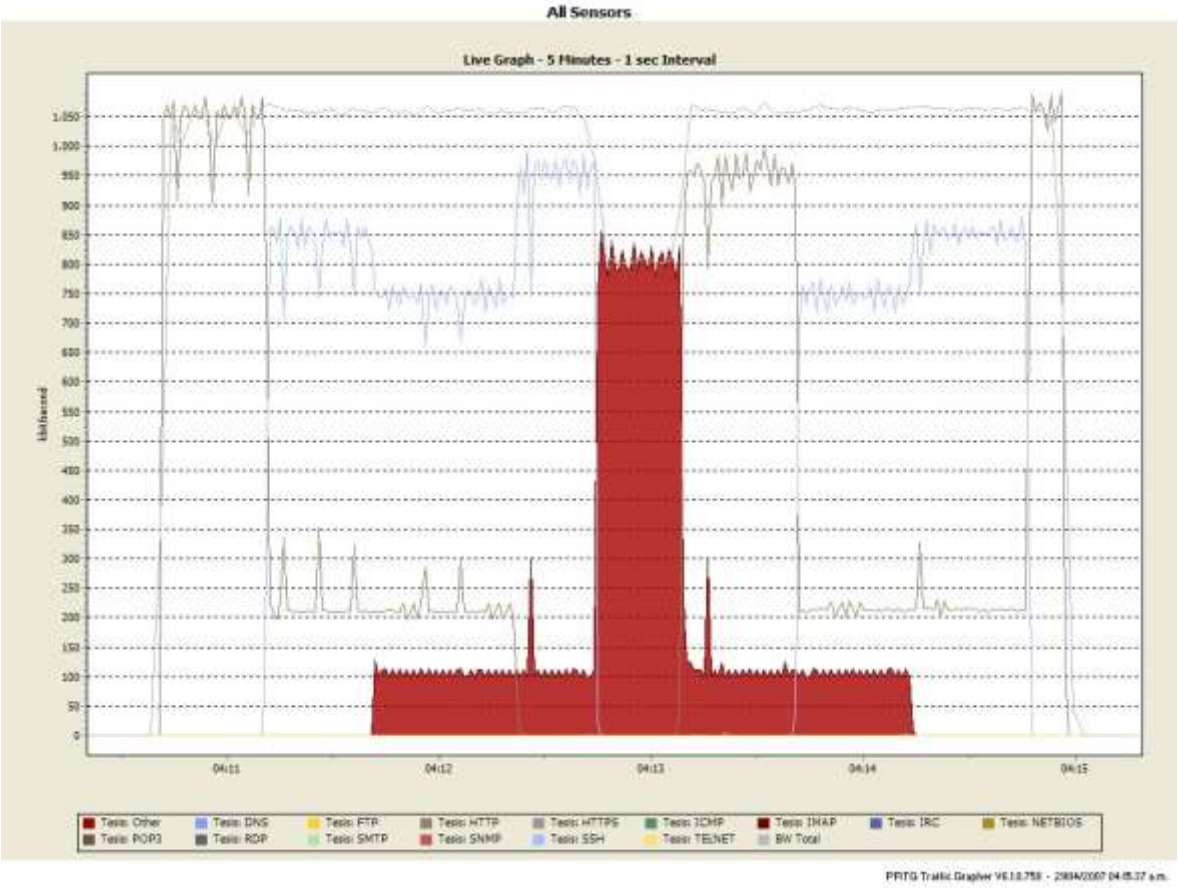

Figura 4.11 Funcionamiento de HTBi.

En la figura 4.12 se observa el tratamiento que recibe el tráfico masivo cuando está utilizando la mayoría del enlace y luego se activan las clases de prioridad superior. El orden en que cambian los flujos de tráfico es el siguiente:

1. Inicia una transferencia de archivos por FTP (tráfico clasificado en la prioridad más baja)
2. Inicia una transferencia de archivos por HTTP usando *Flash Get* (prioridad normal)
3. Inicia una transferencia de archivos por SSH (máxima prioridad)
4. Termina la transferencia por HTTP
5. Termina la transferencia por SSH
6. Termina la transferencia de datos por FTP

Igual que en las pruebas anteriores, las variaciones se hicieron aproximadamente cada 30 segundos.

Se puede notar que en el inicio, el tráfico FTP, aunque es el único tipo de tráfico activo en el momento, no está usando todo el ancho de banda, según el diseño del algoritmo, este podría ocupar todo el canal de transmisión solo si no hay paquetes de otra naturaleza, sin embargo, esta es una situación que no se da muy comúnmente, dicha configuración del algoritmo de funcionamiento, resulta conveniente para que el enlace siempre tenga recursos disponibles para el tráfico de prioridad. De todas maneras, es un factor que el administrador de red puede modificar a voluntad en las líneas de

código del planificador de tráfico, incluso, es posible no ofrecerle garantías a este tipo de tráfico.

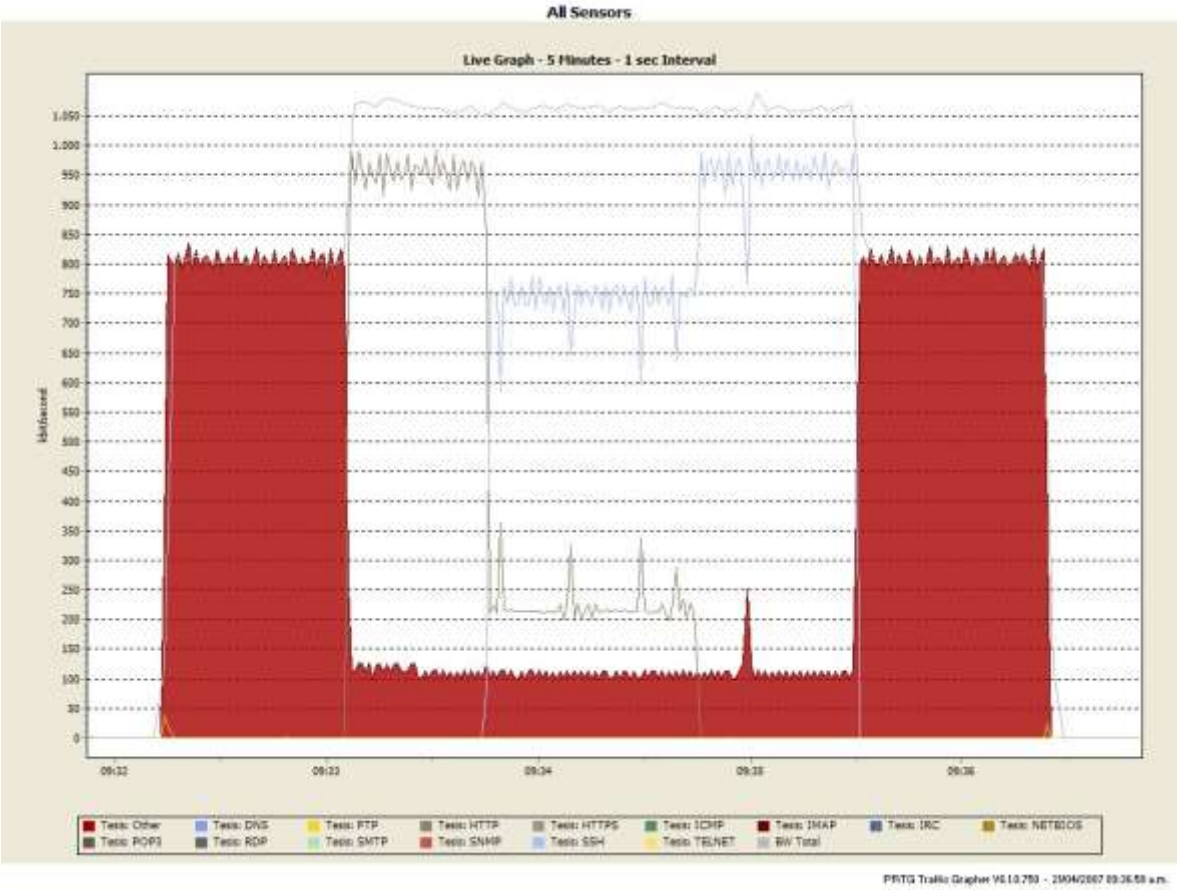

Figura 4.12 Tratamiento para el tráfico masivo

Para probar la efectividad de los filtros de capa 7, se probó con la identificación de VoIP bajo protocolo SIP. En este tipo de llamadas, la comunicación se inicia usando el puerto UDP 5060, sin embargo, la llamada puede establecerse dentro de un rango de puertos RTP, lo que hace difícil su identificación. Sin embargo, como puede verse en la figura 4.13, los filtros de nivel 7 funcionaron perfectamente. Para dicho propósito, como se mencionó al inicio de la sección 4.2.5, fue necesario descargar el correspondiente archivo .pat de la página de protocolos soportados [6], copiarlo en el directorio `/etc/l7-protocols/` en la estructura de archivos de los enrutadores y activar el filtrado en el archivo de configuración del planificador de tráfico.

Los cambios en las transferencias se hicieron aproximadamente cada 45 segundos y son los siguientes:

1. Inicia una transferencia de datos HTTP con *Flash Get* (tráfico normal)
2. Inicia una transferencia de datos SSH (tráfico de prioridad)
3. Inicia la llamada VoIP usando protocolo SIP (máxima prioridad)
4. Finaliza la transferencia de datos por SSH
5. Finaliza la transferencia de datos por HTTP

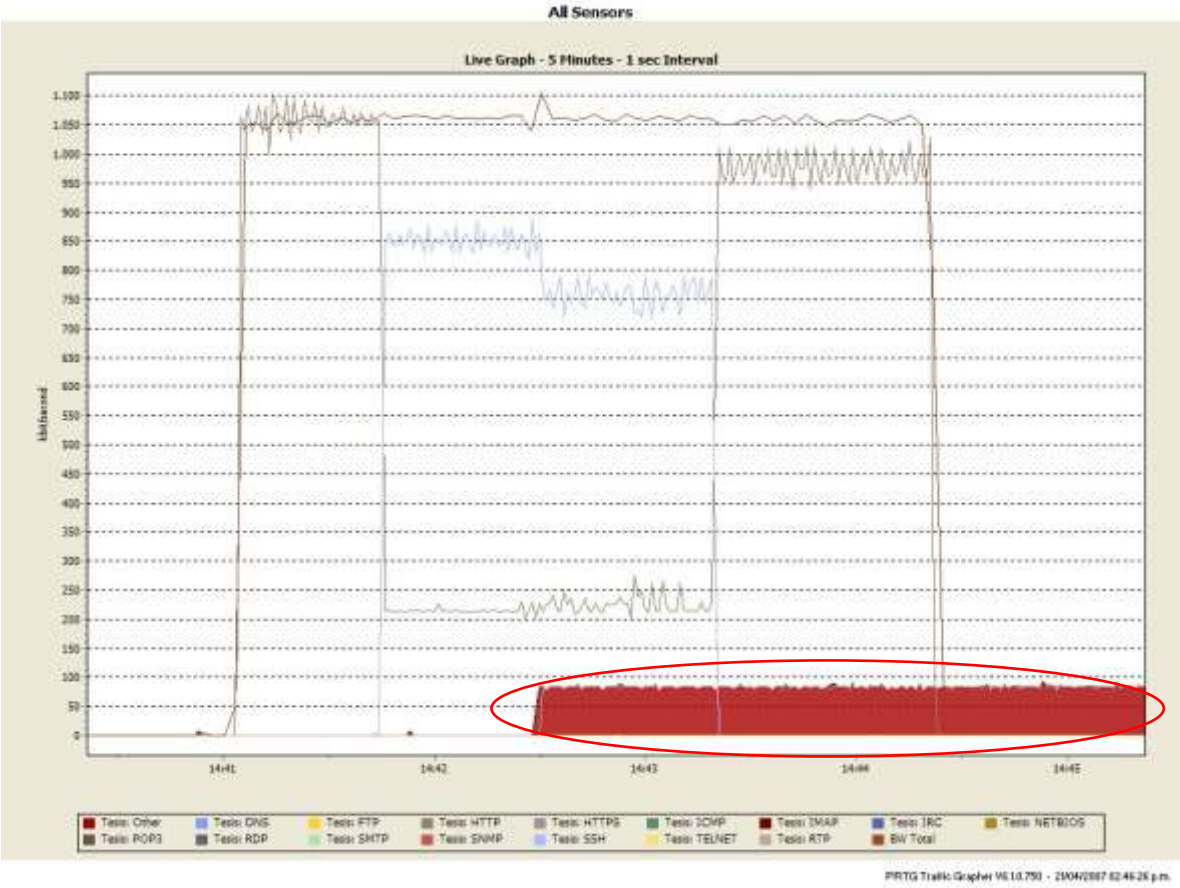

Figura 4.13 Identificación y control de tráfico frente a una llamada VoIP

En la gráfica, es claro que la llamada experimenta muy poca interferencia ocasionada por los otros dos flujos, aun cuando el enlace está todo el tiempo a su máxima capacidad.

Otra forma de filtrar la llamada de VoIP, hubiera sido definir el rango de puertos en la variable de configuración “UDP_EXPR” o configurar el servidor de la aplicación que solo utilice un puerto para las llamadas, pero tomando la primera opción se corre el riesgo de que otras aplicaciones como las “peer-to-peer” utilicen estos puertos y bloqueen los otros tipos de tráfico, por lo que se generaría un fallo de seguridad. Y la segunda opción no es viable por el hecho de que solo se podría hacer una llamada a la vez, y el restringir el rango se traduciría en la restricción del número de las mismas.

Para observar mejor la distribución de los porcentajes del ancho de banda cuando el enlace está totalmente copado se realizó la siguiente prueba, donde se verán las 4 prioridades configuradas en el control de tráfico y su respectivo porcentaje asignado del enlace, para esto, se tienen 4 tipos de tráfico: SSH, HTTP, HTTPS y FTP. SSH tiene la máxima prioridad, HTTPS es tráfico de prioridad, HTTP es tráfico de prioridad normal y FTP tiene la prioridad más baja.

Los cambios en los flujos de tráfico son los siguientes, cada cambio se da aproximadamente cada 30 segundos, a excepción del lapso en el que se encuentran activos los 4 tipos de tráfico, este dura 60 segundos:

1. Se inicia una transferencia de datos por HTTP

2. Se inicia una transferencia de datos por SSH
3. Se inicia una transferencia de datos por FTP
4. Se inicia una transferencia de datos por HTTPS
5. Se termina la transferencia por SSH
6. Se termina la transferencia por HTTPS
7. Se termina la transferencia por HTTP
8. Se termina la transferencia por FTP

Los resultados se muestran en la figura 4.14.

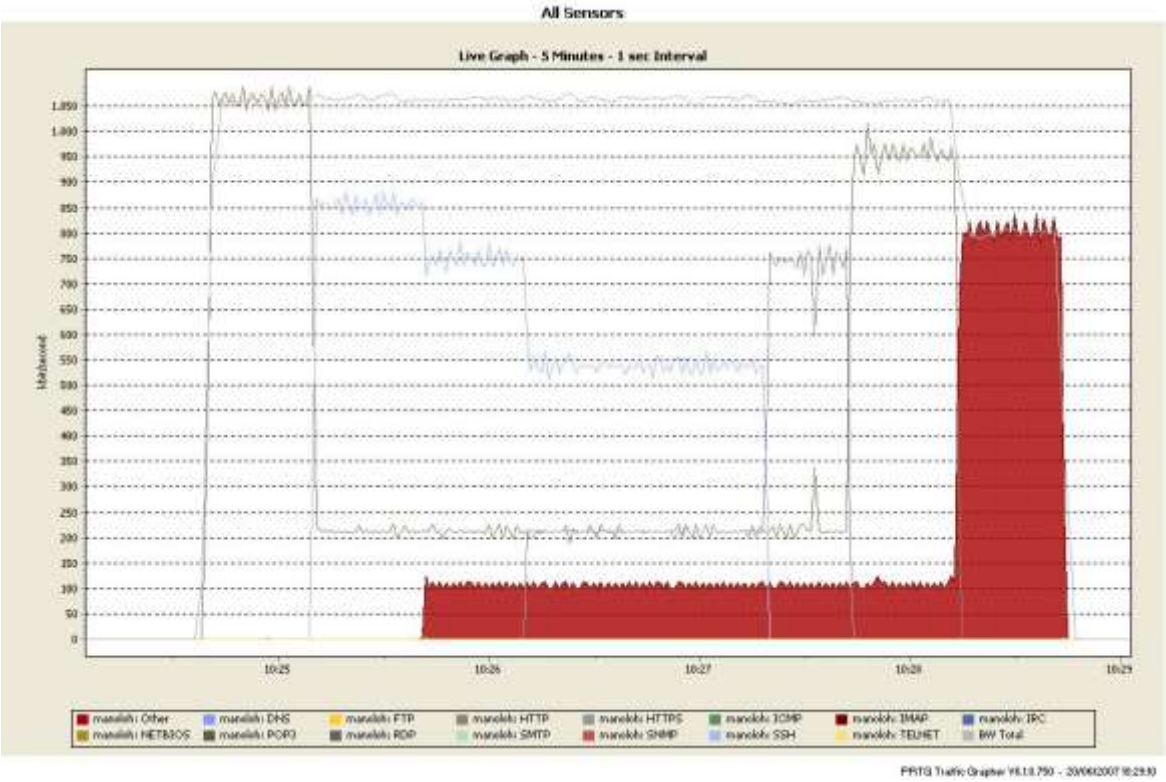

Figura 4.14 Distribución del enlace en los porcentajes asignados

Después de tener iniciados los 4 flujos de tráfico, en la mitad de la figura, se puede ver de abajo hacia arriba, que el tráfico FTP está ocupando aproximadamente un 10% del enlace de 1 Mb, HTTP y HTTPS ocupan un 20% cada uno, y SSH ocupa un 50%.

Otro parámetro que es posible variar es el *burstiness* de la comunicación, esto se hace cambiando las variables “DBURST_U” y “DBURST_D”, vistas en la sección 4.2.1. En las pruebas anteriores se trabajó con un valor de 10 ms para las dos variables, a continuación en las gráficas 4.15, 4.16 y 4.17, se puede ver el comportamiento del enlace con estas variables configuradas en un valor de 300 ms, 1000 ms y 2000 ms respectivamente. Los flujos de tráfico se ven de la siguiente forma:

1. Se inicia una transferencia de datos por SSH
2. Se inicia una transferencia de datos por HTTPS
3. Se inicia una transferencia de datos por HTTP
4. Se inicia una transferencia de datos por FTP
5. Se termina la transferencia por FTP

6. Se termina la transferencia por HTTP
7. Se termina la transferencia por HTTPS
8. Se termina la transferencia por SSH

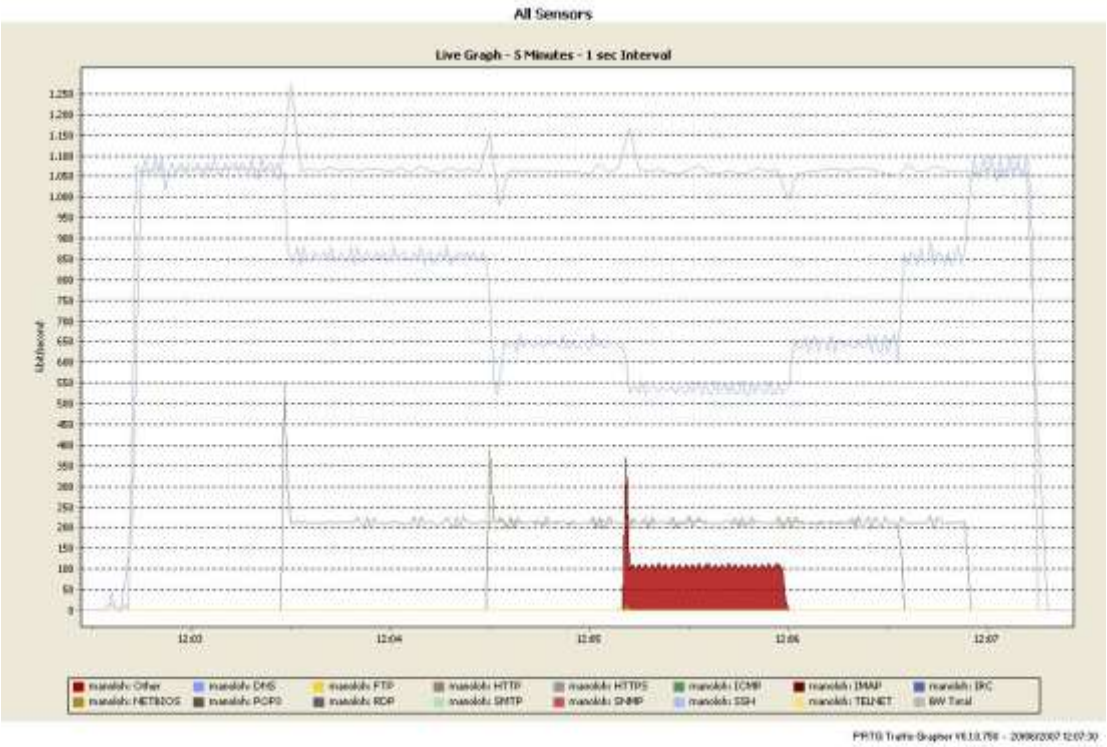

Figura 4.15 Comportamiento del tráfico con burstiness configurado a 300 ms

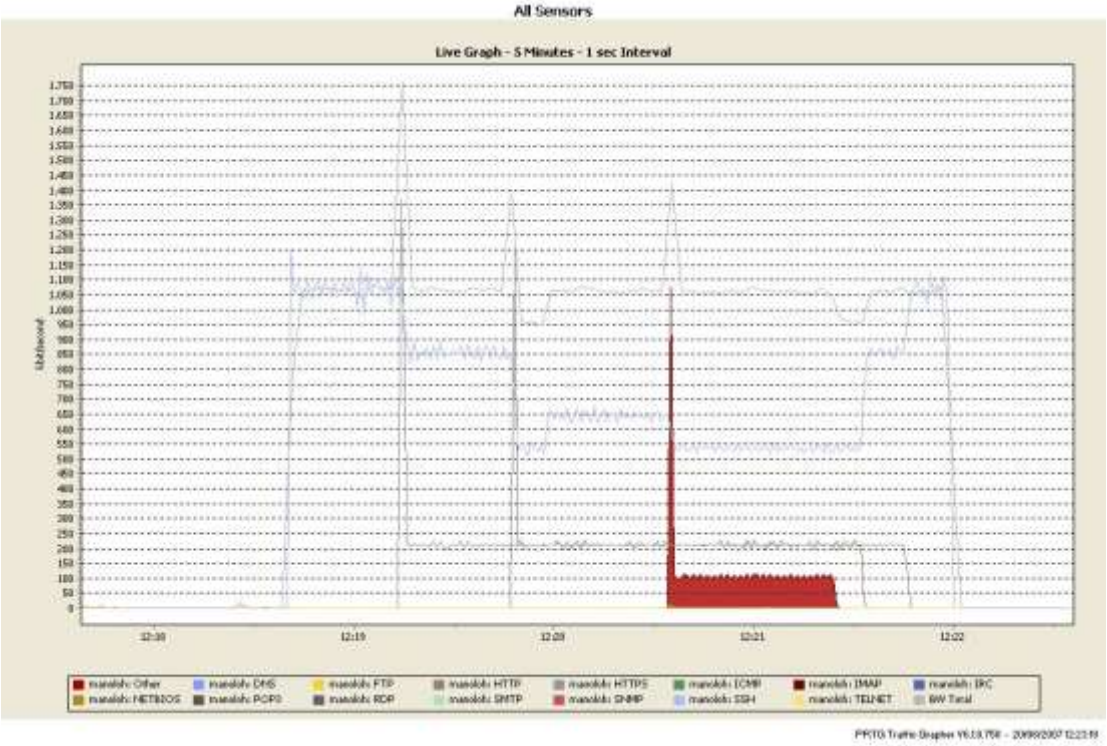

Figura 4.16 Comportamiento del tráfico con burstiness configurado a 1000 ms

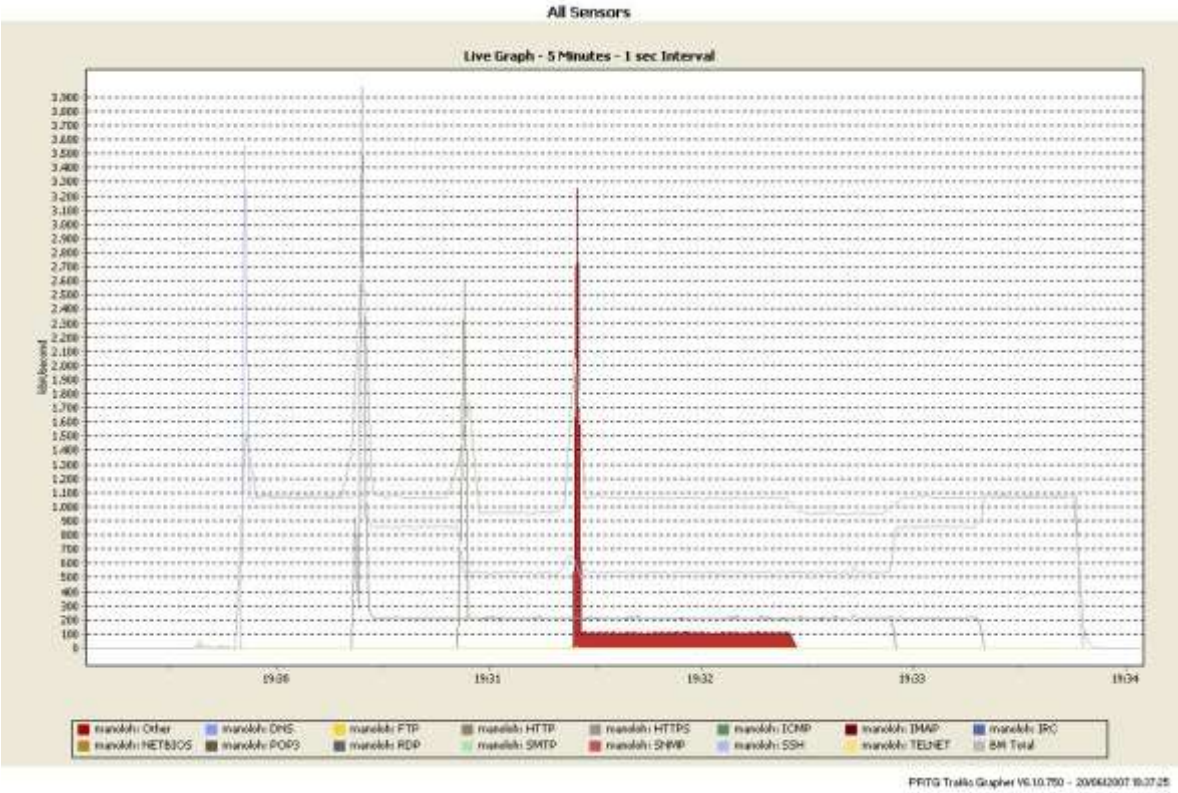

Figura 4.17 Comportamiento del tráfico con burstiness configurado a 2000 ms

Los valores configurados son lo suficientemente altos como para ver una variación notable. Lo interesante de la prueba es notar que cuando se hace una transición en la distribución del ancho de banda disponible se genera un pico, la duración de este pico es lo que se controla con el *burstiness,* esto no quiere decir que solo en ese instante se dé el fenómeno, estos picos se generan todo el tiempo, sin embargo es cuando hay cambios de la distribución del ancho de banda que se vuelve más evidente.

En esta última prueba también se puede notar como la clase de prioridad máxima va "devolviendo" el ancho de banda que hace uso cuando una conexión de prioridad menor lo requiere.

RESUMEN

El capítulo 4 hace uso de la teoría vista en los capítulos anteriores para sustentar la implementación del planificador de tráfico HTBi. Los resultados demuestran que los algoritmos y las herramientas para el control de tráfico, disponibles en Linux, combinados con los bajos costos de los puntos de acceso pueden generar una solución robusta y eficiente para la optimización de la infraestructura de red en entornos inalámbricos.

CONCLUSIONES Y RECOMENDACIONES

Además del aporte que se le hace a la sociedad con la implementación del esquema piloto que permite optimizar los enlaces 802.11 que funcionan actualmente o los que se requieran implementar con esta tecnología para soportar nuevos servicios, trabajar en la Investigación aplicada a nuevos métodos para lograr un acercamiento a la calidad de servicio deseada en redes inalámbricas es una actividad de la cual se pueden beneficiar los usuarios actuales y futuros de este tipo de redes, y los proveedores de dispositivos para estas tecnologías; también se hace un aporte hacia la academia y la investigación de la Universidad, ya que los resultados que se obtuvieron en este proyecto enriquecerán el conocimiento actual sobre del control de tráfico en Linux, el *firmware* de los puntos de acceso inalámbricos, los mecanismos de conformación y los modelos jerárquicos para diferenciación de tráfico.

Mientras la arquitectura de control de tráfico de Linux es capaz de soportar casi todos los escenarios de aplicación de control de tráfico en redes cableadas (IntServ, DiffServ, MPLS, etc.) y muchos sistemas de este tipo han sido implementados, presenta desventajas cuando es usada en ambientes inalámbricos. Los principales problemas son:

- TC Linux, estudiado en la sección 3.6.8, es un ejemplo de una arquitectura de conformación de tráfico implementada sobre la capa MAC. Por lo tanto, los mecanismos de regulación de ancho de banda trabajan asumiendo que el medio de transmisión tiene un ancho de banda constante y no tienen en cuenta el consumo de recursos específico del destino.

- Los algoritmos de planificación de tráfico para redes inalámbricas, como se explicaron en la sección 1.2.8, asumen un conocimiento perfecto del estado del canal de transmisión. El concepto de un monitor del canal inalámbrico está fuera de las capacidades de TC y por lo tanto una disciplina de encolamiento no tiene acceso a la información sobre el estado del canal inalámbrico de una estación o cliente.

- Las disciplinas de encolamiento, específicamente diseñadas para soportar ambientes inalámbricos aún no existen.

En el núcleo de cualquier arquitectura de calidad de servicio debe haber un algoritmo de planificación que distribuya el ancho de banda disponible no solo entre usuarios y compañías, sino también, entre las aplicaciones y protocolos. Es aquí donde las herramientas de gestión y control del tráfico se convierten en un elemento imprescindible en la arquitectura funcional de una red de comunicaciones.

En el proyecto llevado a cabo se ha presentado un control de tráfico que provee un mecanismo escalable y simple para brindar calidad de servicio, además de soportarse en dispositivos que no tienen la capacidad para reconocer o marcar los campos en la trama IP que hacen parte de mecanismos de calidad de servicio.

Es clara la importancia de utilizar un dispositivo adecuado, ya que no todos los enrutadores inalámbricos disponibles admiten cambios de *firmware*. Dadas las

condiciones del mercado de infraestructura para redes, es más conveniente adquirir hardware 802.11 con características adecuadas que hacer el montaje del control de tráfico sobre computadores de escritorio.

El conjunto de herramientas para red de Linux, utilizadas para la implementación, ofrecen la flexibilidad y escalabilidad necesarias para adaptar la solución propuesta a otros escenarios y extender su funcionalidad a diversas topologías de red.

Para futuras implementaciones y continuidad del proyecto sería deseable la incorporación de un monitor del estado del canal inalámbrico, el cual ofrecería la posibilidad de poder hacer la planificación del tráfico de acuerdo con la disponibilidad real de los recursos.

Las pruebas realizadas demuestran que el control de tráfico para redes inalámbricas de este proyecto, permite hacer un completo control sobre el ancho de banda a la vez que ofrece un filtrado acertado y eficiente, dadas las diferentes técnicas de identificación de tráfico que incorpora, por lo que representa una plataforma flexible para cualquier proveedor de servicios o administrador de red. La gestión dinámica del ancho de banda indudablemente mejora el rendimiento del enlace, el hecho de que ésta se base en que si hay ancho de banda disponible en el enlace, no forza la pérdida de paquetes para aquellos flujos que no tienen garantizado completamente los recursos necesarios, supone que no hay la necesidad de restringir las tasas de transferencia estáticamente para darle predicitibildad al enlace.

El control de tráfico también permite la convivencia de flujos TCP y UDP. Evita la reducción de velocidad de transmisión debida al control de congestión de los flujos TCP garantizándoles un ancho de banda mínimo, frente a flujos UDP de altas tasas de transmisión; se puede afirmar con confianza que presenta una solución robusta ante la presencia de flujos interferentes.

Con las pruebas realizadas y la incorporación de Linux al hardware 802.11 se puede brindar calidad de servicio, sobre una plataforma de bajo costo como lo representan los enrutadores que se usaron en el proyecto o un computador con dicho sistema operativo.

La existencia de nuevas tecnologías inalámbricas para la comunicación de datos aún no ha hecho avances significativos como para dejar obsoleta a Wi-Fi, ésta sigue siendo una tecnología actual y en evolución, que con la incorporación de las herramientas Linux al hardware, ha potencializado sus capacidades mejorando la respuesta a los servicios que requieren más que un tratamiento *best effort.*

Las redes inalámbricas pueden significar ventajas de productividad importantes a las empresas y estas ventajas pueden ser multiplicadas por la convergencia de servicios de voz en la infraestructura WLAN. Cuando las aplicaciones de voz son adicionadas a la WLAN, la infraestructura debe ser diseñada para diferenciar voz y datos, y de esta forma dar el tratamiento adecuado a cada tipo de tráfico.

Los proveedores de telecomunicaciones colombianos ofrecen enlaces con anchos de banda de poca capacidad y calidad deficiente, la implementación de soluciones como la que se desarrolló con este trabajo ofrecen a las organizaciones una excelente alternativa para que puedan hacer un uso óptimo de su infraestructura de red, específicamente, de los enlaces inalámbricos.

Así mismo, los dispositivos disponibles en el mercado tienen costos elevados y muchas, veces son limitados respecto al ancho de banda que son capaces de controlar, con software libre y dispositivos de bajo costo se puede realizar un control de tráfico de alta calidad sin una inversión económica elevada.

El *software* libre actualmente ofrece aplicaciones potentes y escalables, que en el caso específico del control de tráfico, son comparables con el procesamiento realizado en modernos enrutadores. La comunidad de usuarios de *software* libre, crece aceleradamente, pues a ella se adhieren no solo entusiastas de la programación, sino también, desarrolladores que encuentran en este tipo de licencias la herramienta más adecuada para aplicaciones específicas. Por la naturaleza abierta de este software la documentación disponible es considerable, además, la facilidad para encontrar tanto las versiones de Kernel, como los parches requeridos, hacen de el una opción para considerar en cualquier tipo de proyecto.

La notable mejoría en el enlace, encontrada en las pruebas, es el resultado de más de un año de trabajo de los estudiantes junto al director y una muestra del alcance de la implementación. Para el desarrollo fue clave la novedad (selección de la herramienta y de los dispositivos utilizados en las pruebas), el escenario (solución real a un problema común) y el impacto del resultado global, en estos momentos cuando la redes inalámbricas necesitan un mejor control de tráfico y muy pronto cuando el posicionamiento de los dispositivos 802.11e o Wi-Max, permita que se incremente la variedad de servicios ofrecidos, el usuario no tendrá la necesidad de reemplazar su dispositivo 802.11 previo y podrá recibir estos servicios eficientemente.

REFERENCIAS BIBLIOGRÁFICAS CAPÍTULO 1

[1] FISCHE, Georges. HEBUTERNE, Gerard. Communicating Systems & Networks: Traffic & Performance. Kogan Page, 2004

[2] PARK Kun I. Ph.D, QoS in Packet Networks, Springer, 2005

[3] WISELY, Dave. EARDLEY, Philip. BURNESS. IP for 3G: Networking Technologies for Mobile Communications. John Wiley & Sons, 2002.

[4] CARDEI, Michaela. CARDEI, Iount. DU, Ding-Zhu. Resource Management in Wireless Networking. Springer, 2005

[5] DAIGLE, John. Queuing Theory with application to packet telecommunications. Springer Science+Business Media, 2005.

[6] GIAMBENE, Giovanni. Queuing Theory and telecommunications: Networks and Applications. Springer Science, 2005.

[7] JANEVSKI, Toni. Traffic Analysis and Design of Wireless IP Networks. Artech House, 2003

[8] SHREEDHAR, M. VARHESE, George. Efficient Fair Queuing Using Deficit Round-Robin

[9] Información disponible en el sitio WEB: http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm

[10] FLOYD, Sally. VAN, Jacobson. Link-sharing and Resource Management Models for Packet Networks. Documento que apareció en IEEE/ACM Transactions on Networking, Vol. 3 No. 4, August 1995

[11] Sally Floyd. Notes on cbq and guaranteed service, 1995.

[12] M. Srivastava, C. Fragouli, V. Sivaraman. Controlled multimedia wireless link sharing via enhanced class-based queueing with channel-state-dependent packet scheduling. En desarrollo de: *the Conference on Computer Communications (IEEE Infocom)*, Abril 1998

[13] Ion Stoica, Hui Zhang, T.S. Eugene Ng. A hierarchical fair service curve algorithm for link-sharing, real-time and priority service. En desarrollo de *SIGCOMM'97*, 1997. http://www.cs.cmu.edu/¥ zhang/HFSC/

[14] T. S. Eugene Ng, Donpaul C. Stephens, Ion Stoica, Hui Zhang. Supporting besteffort traffic with fair service curve. In *Measurement and Modeling of Computer Systems*, 1999

[15] WONG, Daniel K. Wireless Internet Telecommunications. Artech House, 2005

[16] Jean-Pierre Ebert, Andreas Willig. A *gilbert-elliot* bit error model and the efficient use in packet level simulation. Reporte Técnico, Departamento de Ingeniería Eléctrica, Technical University of Berlin, 1999

[17] ROSHAN, Pejman. LEARY, Jonathan. 802.11 Wireless LAN Fundamentals. Cisco Press. 2003.

[18] Ferguson, P., G. Huston, *Quality of Service: Delivering QoS on the Internet and in Corporate Networks,* New York: John Wiley & Sons, 1998.

REFERENCIAS BIBLIOGRÁFICAS CAPÍTULO 2

[1] WONG, Daniel K. Wireless Internet Telecommunications. Artech House, 2005

[2] Ferguson, P., G. Huston, *Quality of Service: Delivering QoS on the Internet and in Corporate Networks,* New York: John Wiley & Sons, 1998.

[3] Braden, R., et al. “Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification," RFC 2205, Septiembre, 1997. Disponible en línea en: http://www.ietf.org/rfc/rfc2205.txt

[4] IntServ/DiffServ Framework. Documento digital en version acrobat reader disponible en el sitio web: http://www3.ietf.org/proceedings/99jul/slides/issll-framework-99jul.pdf, 2006.

[5] Documento disponible en el sitio WEB: www.faqs.org/rfcs/rfc1349.html

[6] Documento disponible en el sitio WEB: www.faqs.org/rfcs/rfc791.html

[7] Documento disponible en el sitio WEB: www.faqs.org/rfcs/rfc2474.html

[8] PUJOLLE, Guy. Management, Control and Evolution of IP Networks. International Scientific and Technical Encyclopedia, 2007

[9] JANEVSKI, Toni. Traffic Analysis and Design of Wireless IP Networks. Artech House, 2003

[10] PARK Kun I. Ph.D, QoS in Packet Networks, Springer, 2005

[11] FLANNAGAN, Michael., DURAND, Benoit., SOMMERVILLE, Jerry., BUCHAMANN, Mark., FULLER, Ron. Administering Cisco QoS for IP Networks. Syngress Publishing, 2001

[12] Documento disponible en el sitio WEB: http://www.faqs.org/rfcs/rfc2481.html

[13] PRASAD, Anand., PRASAD, Neeli. 802.11 WLANs and IP Networking: Security, QoS, and mobility. Artech House, 2005

[14] HEINANEN, J. FINLAND, Telia. GUERIN, R. ¨A Single Rate Three Color Marker¨, RFC 2697. Septiembre, 1999

[15] HEINANEN, J. FINLAND, Telia. GUERIN, R. ¨A Two Rate Three Color Marker¨, RFC 2698. Septiembre, 1999

[16] OHRTMAN, Frank. Voice over 802.11. Artech House, 2004

REFERENCIAS BIBLIOGRÁFICAS CAPÍTULO 3

[1] JANEVSKI, Toni. Traffic Analysis and Design of Wireless IP Networks. Artech House, 2003

[2] Syed Mahbubur Rahman. Multimedia Networking: Technology, Management and Applications. Idea Group Publishing, 2002

[3] PRASAD, Anand., PRASAD, Neeli. 802.11 WLANs and IP Networking: Security, QoS, and mobility. Artech House, 2005

[4] Antony Oodan et al. Telecommunications Quality of Service Management: From Legacy to Emerging Services. Institution of Electrical Engineers, 2003

[5] Dave Kosiur. IP Multicasting: The Complete Guide to Interactive Corporate Networks. John Wiley & Sons, 1998

[6] PARK Kun I. Ph.D, QoS in Packet Networks, Springer, 2005

[7] Hardy, William C. QoS: Measurement and Evaluation of Telecommunications Quality of Service. Wiley & Sons, Ltd., 2001

[8] Thompson, K., G. J. Miller, R. Wilder. Wide-Area Internet Traffic Patterns and Characteristics. *IEEE Network*, Noviembre/Diciembre 1997

[9] Claffy, K. C., “Internet Traffic Characterization,” Ph.D. dissertation, University of California, San Diego, CA, 1994.

[10] ITU-T, *Transmission Systems and Media, General Recommendation on the Transmission Quality for an Entire International Telephone Connection: One-Way Transmission Time*, Recomendacion G.114, Telecommunication Standardization Sector de ITU, Geneva, Switzerland, Marzo 1993.

[11] McDysan, David E. QoS & Traffic Management in IP & ATM Networks. New York McGraw-Hill Professional, 2000

[12] Werner Almesberger. Linux traffic control - implementation overview. Technical report, EPFL, Nov 1998. http://tcng.sourceforge.net

[13] Werner Almesberger, Jamal Hadi Salim, and Alexey Kuznetsov. Differentiated services on linux, June 1999. http://diffserv.sourceforge.net

[14] KOPPER, K. The Linux Enterprise Cluster. No Starch Press, 2005

[15] IP Tables Tutorial 1.2.2 Información disponible en el sitio Web: http://iptables-tutorial.frozentux.net/iptables-tutorial.html#NATINTRO

[16] Información disponible en el sitio WEB: http://netfilter.org/documentation/index.html#documentation-howto

[17] Información disponible en el sitio WEB: http://netfilter.org/projects/iptables/downloads.html

[18] Hubert, B. Linux Advanced Routing & Traffic Control HOWTO. DocBook Edition, 2002.

[19] Sally F. Van J. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413. Agosto, 1993

[20] BROWN Martin A., Traffic Control using tcng and HTB HowTo, Version 1.0, Abril 2003, Disponible en linea en: http://www.tldp.org/HOWTO/Traffic-Control-tcng-HTB-HOWTO/

[21] Información disponible en el sitio Web: http://l7-filter.sourceforge.net/

[22] Información disponible en el sitio Web: http://www.ipp2p.org/

[23] Información disponible en el sitio Web: http://linux-net.osdl.org/index.php/Iproute2

[24] Alexey Kuznetsov. iproute2 package. ftp://ftp.inr.ac.ru/ip-routing/

[25] Información disponible en el sitio Web: http://luxik.cdi.cz/~devik/qos/htb/

[26] Información disponible en el sitio Web: http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm

[27] Información disponible en el sitio Web: http://www.linuximq.net

[28] Información disponible en el sito Web: http://tcng.sourceforge.net

REFERENCIAS BIBLIOGRÁFICAS CAPÍTULO 4

[1] DEVERA, Martin. "HTB Home" 2003. Información disponible en sitio web: http://luxik.cdi.cz/~devik/qos/htb/

[2] Información disponible en web: http://files.eschauzier.org/20-qos-htb

[3] SOMMER, Ethan. STRAIT, Matthew. Application Layer Packet Classifier for Linux. 2007. Información disponible en sitio web: http://l7-filter.sourceforge.net/

[4] SOMMER, Ethan. STRAIT, Matthew. L7-filter Supported Protocols. 2007. Información disponible en sitio web: http://l7-filter.sourceforge.net/protocols

[5] About IPP2P. Información disponible en sitio web: http://ipp2p.org/

[6] RODRÍGUEZ, Alberto. MORENO, William. "ANÁLISIS Y PRUEBAS DE DESEMPEÑO EN DISPOSITIVOS AP MEDIANTE IMPLEMENTACIÓN DE UN FIRMWARE BASADO EN EL KERNEL DE LINUX", Trabajo de Grado para optar al título de Ingeniero en Electrónica y Telecomunicaciones, Universidad del Cauca, 2007.

[7] OpenWRT, TableOfHardware. Información disponible en sitio web: http://wiki.openwrt.org/TableOfHardware

[8] OpenWRT, OpenWrtDocsConfiguration. Información disponible en sitio web: http://wiki.openwrt.org/OpenWrtDocs/Configuration#NetworkInterfaceNames

[9] PRTG Traffic Grapher, Bandwidth and Network Monitoring Made Easy. Paessler. Información disponible en sitio web: http://www.paessler.com/prtg

[10] Fash Get, The Best Download Manager. Información disponible en sitio web: http://www.flashget.com/en/download.htm

ANEXO 1: *SCRIPT* DE FUNCIONAMIENTO

El código de los *script* utilizado en los enrutadores se muestra a continuación, es de libre uso y modificación y puede encontrarse en [2].

Las variables previamente configuradas en este archivo no representan una camisa de fuerza para el administrador de red, su presencia, debe entenderse únicamente como un ejemplo de configuración.

Codigo del Script de Funcionamiento

```
#!/bin/sh
# Versión final

# Ubicación del archivo de configuración
[ -f /etc/qos.conf ] && . /etc/qos.conf

# Si no hay interfaz definida, use la interfaz WAN
[ "$QOS_IF" ] || QOS_IF=$(nvram get wan_ifname)

# Interfaz QoS activa
HOTPLUG_IF=$(nvram get ${INTERFACE}_ifname)

# Si la interfaz está activa se inicia el script de QoS
if [ "$ACTION" = "ifup" -a \( "$HOTPLUG_IF" = "$QOS_IF" -o
"$INTERFACE" = "qosstart" \) ]; then

  # Duración permitida para ráfagas de tráfico en ms
  DBURST_D=10
  DBURST_U=10

  MTU=1600

  # Mínima tasa de transferencia por clase en términos del porcentaje
de la capacidad total del enlace
  MIN_RATE=10

  # Paquetes necesarios para el funcionamiento del script:
  # kmod-sched
  # kmod-ipt-conntrack
  # iptables-mod-conntrack
  # kmod-ipt-ipopt
  # iptables-mod-ipopt
  # kmod-ipt-extra
  # iptables-mod-extra
  # ip
  # kmod-imq
  # iptables-mod-imq
  # kmod-ipt-filter
  # iptables-mod-filter

  # Carga de los módulos
  insmod cls_fw >&- 2>&-
  insmod sch_htb >&- 2>&-
  insmod sch_sfq >&- 2>&-
  insmod ipt_CONNMARK >&- 2>&-
```

```
  insmod ipt_length >&- 2>&-
  insmod imq numdevs=1 >&- 2>&-
  insmod ipt_IMQ >&- 2>&-
  insmod ipt_ipp2p >&- 2>&-
  insmod ipt_layer7 >&- 2>&-
  insmod ipt_tos >&- 2>&-
  insmod ipt_TOS >&- 2>&-
  insmod ipt_dscp >&- 2>&-
  insmod ipt_DSCP >&- 2>&-
  insmod sch_red >&- 2>&-

  iptables -t mangle -F
  iptables -t mangle -X

  # Inicializar cola IMQ (InterMediate Queuing device)
  ip link set imq0 up

  # Remover todas las disciplinas de encolamiento de las interfaces
  sed -n 's/ *\(.*\):.*/\1/p' /proc/net/dev | while read INTERFACE; do
    tc qdisc del dev $INTERFACE root >&- 2>&-
  done

  [ $UPLOAD -ne 0 ] && {
    # Cálculo del tamaño del buffer en bytes
    BURST_U=$(($DBURST_U*$UPLOAD/8))

    # Comparar tamaño del buffer con el MTU
    [ $BURST_U -lt $((1*$MTU)) ] && BURST_U=$((1*$MTU))

    # Cálculo del factor r2q para la disciplina HTB
    RTOQ_U=$(($MIN_RATE*$UPLOAD*10/(8*$MTU)))
    [ $RTOQ_U -gt 20 ] && RTOQ_U=20
    [ $RTOQ_U -eq 0 ] && RTOQ_U=1

    # Creación de las colas de salida en la interfaz QoS
    tc qdisc add dev $QOS_IF root handle 1: htb default 40 r2q $RTOQ_U
    tc class add dev $QOS_IF parent 1: classid 1:1 htb rate
${UPLOAD}kbit ceil ${UPLOAD}kbit burst $(($BURST_U*3)) cburst
$(($BURST_U*3)) mtu $MTU
    tc class add dev $QOS_IF parent 1:1 classid 1:10 htb rate
$(($UPLOAD*5/10))kbit ceil ${UPLOAD}kbit burst $(($BURST_U*1)) cburst
$(($BURST_U*1)) prio 1 mtu $MTU
    tc class add dev $QOS_IF parent 1:1 classid 1:20 htb rate
$(($UPLOAD*2/10))kbit ceil ${UPLOAD}kbit burst $(($BURST_U*1)) cburst
$(($BURST_U*1)) prio 2 mtu $MTU
    tc class add dev $QOS_IF parent 1:1 classid 1:30 htb rate
$(($UPLOAD*2/10))kbit ceil ${UPLOAD}kbit burst $(($BURST_U*1)) cburst
$(($BURST_U*1)) prio 3 mtu $MTU
    tc class add dev $QOS_IF parent 1:1 classid 1:40 htb rate
$(($UPLOAD*1/10))kbit ceil ${UPLOAD}kbit burst $(($BURST_U*1)) cburst
$(($BURST_U*1)) prio 4 mtu $MTU

    tc qdisc add dev $QOS_IF parent 1:10 sfq quantum $MTU perturb 10
    tc qdisc add dev $QOS_IF parent 1:20 sfq quantum $MTU perturb 10
    tc qdisc add dev $QOS_IF parent 1:30 sfq quantum $MTU perturb 10
    tc qdisc add dev $QOS_IF parent 1:40 sfq quantum $MTU perturb 10

    tc filter add dev $QOS_IF parent 1: prio 1 protocol ip handle 1 fw
flowid 1:10
```

```
    tc filter add dev $QOS_IF parent 1: prio 2 protocol ip handle 2 fw
flowid 1:20
    tc filter add dev $QOS_IF parent 1: prio 3 protocol ip handle 3 fw
flowid 1:30
    tc filter add dev $QOS_IF parent 1: prio 4 protocol ip handle 4 fw
flowid 1:40
  }

  [ $DOWNLOAD -ne 0 ] && {
    # Cálculo del tamaño del buffer en bytes
    BURST_D=$(($DBURST_D*$DOWNLOAD/8))

    # Comparar tamaño del buffer con el MTU
    [ $BURST_D -lt $((1*$MTU)) ] && BURST_D=$((1*$MTU))

    # Cálculo del factor r2q para la disciplina HTB
    RTOQ_D=$(($MIN_RATE*$DOWNLOAD*10/(8*$MTU)))
    [ $RTOQ_D -gt 20 ] && RTOQ_D=20
    [ $RTOQ_D -eq 0 ] && RTOQ_D=1

    # Creación de las colas de entrada en la interfaz IMQ
        tc qdisc add dev imq0 root handle 1: htb default 40 r2q
$RTOQ_D
    tc class add dev imq0 parent 1: classid 1:1 htb rate
${DOWNLOAD}kbit ceil ${DOWNLOAD}kbit burst $((BURST_D*3)) cburst
$((BURST_D*3)) mtu $MTU
    tc class add dev imq0 parent 1:1 classid 1:10 htb rate
$(($DOWNLOAD*5/10))kbit ceil ${DOWNLOAD}kbit burst $((BURST_D*1))
cburst $((BURST_D*1)) prio 1 mtu $MTU
    tc class add dev imq0 parent 1:1 classid 1:20 htb rate
$(($DOWNLOAD*2/10))kbit ceil ${DOWNLOAD}kbit burst $((BURST_D*1))
cburst $((BURST_D*1)) prio 2 mtu $MTU
    tc class add dev imq0 parent 1:1 classid 1:30 htb rate
$(($DOWNLOAD*2/10))kbit ceil ${DOWNLOAD}kbit burst $((BURST_D*1))
cburst $((BURST_D*1)) prio 3 mtu $MTU
    tc class add dev imq0 parent 1:1 classid 1:40 htb rate
$(($DOWNLOAD*1/10))kbit ceil $(($DOWNLOAD*3/4))kbit burst
$((BURST_D*1)) cburst $((BURST_D*1)) prio 4 mtu $MTU

    tc qdisc add dev imq0 parent 1:10 red limit $((40*$MTU)) min
$((5*$MTU)) max $((20*$MTU)) avpkt $(($MTU*6/10)) burst 16 probability
0.015
    tc qdisc add dev imq0 parent 1:20 red limit $((40*$MTU)) min
$((5*$MTU)) max $((20*$MTU)) avpkt $(($MTU*6/10)) burst 16 probability
0.015
    tc qdisc add dev imq0 parent 1:30 red limit $((40*$MTU)) min
$((5*$MTU)) max $((20*$MTU)) avpkt $(($MTU*6/10)) burst 16 probability
0.015
    tc qdisc add dev imq0 parent 1:40 red limit $((40*$MTU)) min
$((5*$MTU)) max $((20*$MTU)) avpkt $(($MTU*6/10)) burst 16 probability
0.015

    tc filter add dev imq0 parent 1: prio 1 protocol ip handle 1 fw
flowid 1:10
    tc filter add dev imq0 parent 1: prio 2 protocol ip handle 2 fw
flowid 1:20
    tc filter add dev imq0 parent 1: prio 3 protocol ip handle 3 fw
flowid 1:30
```

```
    tc filter add dev imq0 parent 1: prio 4 protocol ip handle 4 fw
flowid 1:40
  }

  iptables -t mangle -N mark_chain
  iptables -t mangle -N egress_chain
  iptables -t mangle -N ingress_chain

  # Inicialización de la cadena de marcado en la cola de salida
  iptables -t mangle -A POSTROUTING -o $QOS_IF -j egress_chain

  # Marca de ingreso en cadenas "FORWARD" e "INPUT"
  # Marca de ingreso en cadena "FORWARD" de entrada para que el
tráfico LAN pase por la cola IMQ
  iptables -t mangle -A FORWARD -i $QOS_IF -j ingress_chain
  iptables -t mangle -A FORWARD -i $QOS_IF -j IMQ --todev 0

  # Marca de ingreso en cadena "INPUT" de entrada para el que el
tráfico que tiene como destino el router pase por la cola IMQ
  iptables -t mangle -A INPUT -i $QOS_IF -j ingress_chain
  iptables -t mangle -A INPUT -i $QOS_IF -j IMQ --todev 0

  ################### Definición de Funciones ####################
  mark_addr_in()
    {
     # Inicialización de las reglas de ingreso basadas en las
direcciones IP[:puerto[:rango]]
     # $1 es la lista de elementos "ip:puerto"
     # $2 es la prioridad

     for ADDR in $1; do
       IP_PART=`echo $ADDR | sed -n 's/\([^:]*\):.*/\1/p'`
       if [ "$IP_PART" ]; then
         PORT_PART=`echo $ADDR | sed -n 's/[^:]*:\(.*\)/\1/p'`
         iptables -t mangle -A ingress_chain -d $IP_PART -p tcp --
dport $PORT_PART -j MARK --set-mark $2
         iptables -t mangle -A ingress_chain -d $IP_PART -p udp --
dport $PORT_PART -j MARK --set-mark $2
       else
         iptables -t mangle -A ingress_chain -d $ADDR -j MARK --set-
mark $2
       fi
     done
    }

  mark_addr_out()
    {
     # Inicialización de las reglas de salida basadas en las
direcciones IP[:puerto[:rango]]
     # $1 es la lista de elementos "ip:puerto"
     # $2 es la prioridad

     for ADDR in $1; do
       IP_PART=`echo $ADDR | sed -n 's/\([^:]*\):.*/\1/p'`
       if [ "$IP_PART" ]; then
         PORT_PART=`echo $ADDR | sed -n 's/[^:]*:\(.*\)/\1/p'`
         iptables -t mangle -A egress_chain -s $IP_PART -p tcp --dport
$PORT_PART -j MARK --set-mark $2
```

```
        iptables -t mangle -A egress_chain -s $IP_PART -p udp --dport
$PORT_PART -j MARK --set-mark $2
      else
        iptables -t mangle -A egress_chain -s $ADDR -j MARK --set-
mark $2
      fi
     done
    }

################### Cadena de Marcado ############################
  # Restablecer marca si la conexión aún no ha sido marcada
  iptables -t mangle -A mark_chain -m mark --mark 0 -j CONNMARK --
restore-mark

  # Marcar paquetes de máxima prioridad con base en el número del
puerto y protocolo
  for PORT in $UDP_EXPR; do
    iptables -t mangle -A mark_chain -m mark --mark 0 -p udp --dport
$PORT -j MARK --set-mark 1
  done
  for PORT in $TCP_EXPR; do
    iptables -t mangle -A mark_chain -m mark --mark 0 -p tcp --dport
$PORT -j MARK --set-mark 1
  done

  # Marcar paquetes de prioridad con base en el número del puerto y
protocolo
  for PORT in $UDP_PRIO; do
    iptables -t mangle -A mark_chain -m mark --mark 0 -p udp --dport
$PORT -j MARK --set-mark 2
  done
  for PORT in $TCP_PRIO; do
    iptables -t mangle -A mark_chain -m mark --mark 0 -p tcp --dport
$PORT -j MARK --set-mark 2
  done

  # Marcar paquetes de tráfico masivo con base en el número del puerto
y protocolo
  for PORT in $UDP_BULK; do
    iptables -t mangle -A mark_chain -m mark --mark 0 -p udp --dport
$PORT -j MARK --set-mark 4
  done
  for PORT in $TCP_BULK; do
    iptables -t mangle -A mark_chain -m mark --mark 0 -p tcp --dport
$PORT -j MARK --set-mark 4
  done

  # Marcar paquetes de máxima prioridad con base en las coincidencias
ipp2p
  ALL_PROTOS=""
  for PROTO in $IPP2P_EXPR; do
    ALL_PROTOS="${ALL_PROTOS}--$PROTO "
  done
  [ "$ALL_PROTOS" ] && iptables -t mangle -A mark_chain -m mark --mark
0 -m ipp2p $ALL_PROTOS -j MARK --set-mark 1

  # Marcar paquetes de prioridad con base en las coincidencias ipp2p
  ALL_PROTOS=""
  for PROTO in $IPP2P_PRIO; do
```

```
    ALL_PROTOS="${ALL_PROTOS}--$PROTO "
  done
  [ "$ALL_PROTOS" ] && iptables -t mangle -A mark_chain -m mark --mark
0 -m ipp2p $ALL_PROTOS -j MARK --set-mark 2

  # Marcar paquetes de tráfico masivo con base en las coincidencias
ipp2p
  ALL_PROTOS=""
  for PROTO in $IPP2P_BULK; do
    ALL_PROTOS="${ALL_PROTOS}--$PROTO "
  done
  [ "$ALL_PROTOS" ] && iptables -t mangle -A mark_chain -m mark --mark
0 -m ipp2p $ALL_PROTOS -j MARK --set-mark 4

  # Marcar paquetes de máxima prioridad con base en las coincidencias
en los filtros Capa 7
  for PROTO in $L7_EXPR; do
    iptables -t mangle -A mark_chain -m mark --mark 0 -m layer7 --
l7proto $PROTO -j MARK --set-mark 1
  done

  # Marcar paquetes de prioridad con base en las coincidencias en los
filtros Capa 7
  for PROTO in $L7_PRIO; do
    iptables -t mangle -A mark_chain -m mark --mark 0 -m layer7 --
l7proto $PROTO -j MARK --set-mark 2
  done

  # Marcar paquetes de tráfico masivo con base en las coincidencias en
los filtros Capa 7
  for PROTO in $L7_BULK; do
    iptables -t mangle -A mark_chain -m mark --mark 0 -m layer7 --
l7proto $PROTO -j MARK --set-mark 4
  done

  # La prioridad por defecto es la normal (para asegurarse de que todo
paquete pasando por la interfaz se marque)
  iptables -t mangle -A mark_chain -m mark --mark 0 -j MARK --set-mark
3

  # Guardar marca en la conexión
  iptables -t mangle -A mark_chain -j CONNMARK --save-mark

  # Dar máxima prioridad al tráfico ICMP (para reportar latencias
libres de interferencia)
  iptables -t mangle -A mark_chain -p icmp -j MARK --set-mark 1
  iptables -t mangle -A mark_chain -p ipv6-icmp -j MARK --set-mark 1

  # Dar máxima prioridad a paquetes UDP pequeños (mejora la
experiencia en juegos por internet)
  [ "$UDP_LENGTH" -gt 0 ] && iptables -t mangle -A mark_chain -p udp -
m length --length :$UDP_LENGTH -j MARK --set-mark 1

#################### Cola de Ingreso #################
  # Marcar paquetes de tráfico masivo con base en la dirección IP y
puerto destino en la LAN
  mark_addr_in "$IP_BULK" 4
```

```
  # Marcar paquetes de prioridad con base en la dirección IP y puerto
destino en la LAN
  mark_addr_in "$IP_PRIO" 2

  # Marcar paquetes de máxima prioridad con base en la dirección IP y
puerto destino en la LAN
  mark_addr_in "$IP_EXPR" 1

  # LLamar función "mark_chain"
  iptables -t mangle -A ingress_chain -j mark_chain


  ######################## Cola de Salida #######################
  # Marcar paquetes de tráfico masivo con base en las coincidencias en
el campo TOS
  for PROTO in $TOS_BULK; do
    iptables -t mangle -A egress_chain -m tos --tos $PROTO -j MARK --
set-mark 4
  done

  # Marcar paquetes de tráfico de prioridad con base en las
coincidencias en el campo TOS
  for PROTO in $TOS_PRIO; do
    iptables -t mangle -A egress_chain -m tos --tos $PROTO -j MARK --
set-mark 2
  done

  # Marcar paquetes de tráfico de máxima prioridad con base en las
coincidencias en el campo TOS
  for PROTO in $TOS_EXPR; do
    iptables -t mangle -A egress_chain -m tos --tos $PROTO -j MARK --
set-mark 1
  done

  # Marcar paquetes de tráfico masivo con base en las coincidencias en
DSCP
  for PROTO in $DSCP_BULK; do
    iptables -t mangle -A egress_chain -m dscp --dscp $PROTO -j MARK -
-set-mark 4
  done

  # Marcar paquetes de tráfico de prioridad con base en las
coincidencias en DSCP
  for PROTO in $DSCP_PRIO; do
    iptables -t mangle -A egress_chain -m dscp --dscp $PROTO -j MARK -
-set-mark 2
  done

  # Marcar paquetes de tráfico de prioridad máxima con base en las
coincidencias en DSCP
  for PROTO in $DSCP_EXPR; do
    iptables -t mangle -A egress_chain -m dscp --dscp $PROTO -j MARK -
-set-mark 1
  done

  # Marcar paquetes de tráfico masivo con base en la dirección IP y
puerto destino en la LAN
  mark_addr_out "$IP_BULK" 4
```

```
  # Marcar paquetes de prioridad con base en la dirección IP y puerto
destino en la LAN
  mark_addr_out "$IP_PRIO" 2

  # Marcar paquetes de máxima prioridad con base en la dirección IP y
puerto destino en la LAN
  mark_addr_out "$IP_EXPR" 1

  # LLamar función "mark_chain"
  iptables -t mangle -A egress_chain -j mark_chain

  # Asegurarse de que los paquetes ACK tengan máxima prioridad (evita
que la velocidad de subida limite la volcidad de bajada)
  iptables -t mangle -A egress_chain -p tcp -m length --length :128 --
tcp-flags SYN,RST,ACK ACK -j MARK --set-mark 1

fi
exit 0
```

Archivo de Configuración (qos.conf)

```
# Velocidad del canal Downstream en kilobits por segundo
#       MUY IMPORTANTE:
# Poner de 5% a 10% más baja que la velocidad medida en la línea (0
para dehabilitar)
DOWNLOAD=100

# Velocidad del canal Upstream en kilobits por segundo
# Poner de 5% a 10% más baja que la velocidad medida en la línea (0
para dehabilitar)
UPLOAD=50

# Filtro capa 7 (nombre del protocolo)
L7_BULK=""
L7_PRIO=""
L7_EXPR=""

# Filtro IPP2P (nombre del protocolo)
# Por defecto "ipp2p" incluye a los protocolos peer-to-peer más
conocidos
IPP2P_BULK="ipp2p"
IPP2P_PRIO=""
IPP2P_EXPR=""

# Puertos clasificados como de tráfico masivo
TCP_BULK="1024: 21"
UDP_BULK="1024:"

# Puertos clasificados como de tráfico de prioridad
TCP_PRIO="22 23"
UDP_PRIO=""

# Puertos clasificados como de tráfico de máxima prioridad
TCP_EXPR="53"
UDP_EXPR="53"
```

```
# Filtro ToS (Type of Service) (solo cola de salida)
#TOS_BULK="0x02"
#TOS_PRIO=""
#TOS_EXPR="0x10"

# DSCP (Differentiated Services Code Point) (solo cola de salida)
DSCP_BULK=""
DSCP_PRIO=""
DSCP_EXPR=""

# Direcciones IP para tráfico "masivo", "de prioridad" y "prioridad
máxima"
# La dirección IP puede incluir un puerto o un rango de puertos (ej:
192.168.0.1:80 ó
# 192.168.0.1:5900:5910). Para incluir todos los puertos solo se
especifica la dirección IP.
IP_BULK=""
IP_PRIO=""
IP_EXPR=""

# Interfaz QoS. Por defecto toma la interfaz WAN.
#QOS_IF=eth1

# Habilitar prioridad para pequeños paquetes UDP.
# Tamáño máximo para los paquetes UDP de prioridad.
#UDP_LENGTH=256
```

Script de Inicio del Planificador de Tráfico HTBi

```
#!/bin/sh

# Activar QoS

env ACTION=ifup INTERFACE=qosstart /etc/hotplug.d/iface/20-qos
```

Script de Desplieque del Estado del Planificador de Tráfico HTBi

```
#!/bin/sh

# Estado de las colas por interfaz

# Configuración de Usuario
[ -f /etc/qos.conf ] && . /etc/qos.conf

# Si no hay interfaz definida, usar la interfaz WAN
[ "$QOS_IF" ] || QOS_IF=$(nvram get wan_ifname)

echo Estado de la cola de Salida
tc class show dev $QOS_IF

echo Estado de la cola de entrada
tc class show dev imq0
```

Script para Deshabilitar el Planificador de Tráfico HTBi

```
#!/bin/sh
```

```
# Desactivar QoS

iptables -t mangle -F
iptables -t mangle -X

sed -n 's/ *\(.*\):.*/\1/p' /proc/net/dev | while read INTERFACE; do
  tc qdisc del dev $INTERFACE root >&- 2>&-
done

ip link set imq0 down
```