UNIVERSIDAD TECNOLOGICA DE LA MIXTECA

"ESTUDIO E IMPLEMENTACION DE QoS MEDIANTE LAS HERRAMIENTAS IPROUTE2 Y NETFILTER DE LINUX"

T E S I S

PARA OBTENER EL TÍTULO PROFESIONAL DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:
MIREYA MAGDALENA SILVA JIMÉNEZ

ASESOR:
L.I. MÓNICA EDITH GARCÍA GARCÍA

HUAJUAPAN DE LEON, OAXACA., OCTUBRE 2008

Índice

Lista de figuras

Lista de tablas

Capítulo 1

1.1 Introducción

El surgimiento de nuevas aplicaciones en la red (como: bases de datos compartidas, comercio electrónico, transmisión de voz y video, telefonía), ha hecho necesario mantener la calidad y acceso a los servicios que proporciona. Los administradores de red requieren tener un control sobre el tráfico entrante y saliente de la red, de tal manera que se puedan aplicar prioridades a cada uno de estos servicios. A este proceso se le conoce como QoS (*Quality of Service*, Calidad de Servicio), y dado que el servicio de Internet ofrece cada vez mayores aplicaciones, resulta importante optimizar los recursos de la red, lográndolo mediante QoS, ya que su función principal es medir el comportamiento de una red a fin de optimizar su desempeño.

Inicialmente, QoS se basó únicamente en la integridad de los datos, es decir, sólo importaba que no hubiera pérdida del contenido ni secuencia de los datos. Actualmente QoS ofrece nuevas alternativas dependiendo del tipo de usuario, tales como: enrutamiento selectivo, clasificación y control de tráfico, administración del ancho de banda, filtrado y jerarquización de paquetes, acceso remoto a redes y balanceo de carga. La manera en la que se puede aplicar QoS en una red, es a través de los campos de los encabezados de los Protocolos de Internet (IP, *Protocol Internet*). Actualmente, existen dos versiones de IP que son IPv4 e IPv6. El IPv4 ofrece QoS a través del campo ToS

(*Type of Service*, Tipo de Servicio), y el IPv6 a través de los campos TC (*Traffic Class*, Clase de tráfico) y Etiqueta de Flujo (*Flow Label*). Cabe señalar que tanto el campo del IPv4 y del IPv6 están comprendidos dentro de los Servicios Diferenciales.

La IETF (*Internet Engineering Task Force*, Grupo de Trabajo en Ingeniería de Internet) propone varios modelos y mecanismos que permiten la demanda de QoS, los más destacados o conocidos son: Int-Serv (*Integrated Services*, Servicios Integrales), Diff-Serv (*Differentiated Services*, Servicios Diferenciales) y MPLS *(Multi Protocol Label Switching*, Conmutación de Etiquetas Multiprotocolo), para profundizar sobre estos temas se puede consultar [1].

Los Int-Serv, son aquellos que utilizan el RSVP (*Resource ReSerVation Protocol*, Protocolo de Reservación de Recursos) [2], este protocolo se lleva a cabo entre los usuarios y la red el cual consiste en hacer reservas de recursos en cada nodo que pasa, para cada paquete de información, lo que origina un intercambio de mensajes RSVP entre la red y los usuarios, y en cada nodo se mantienen los estados de reserva, generando demasiado tráfico lo cual congestiona la red [URL, 1].

El modelo MPLS[1] es una tecnología de conmutación que proporciona circuitos virtuales en redes IP. Además MPLS, es un mecanismo de enrutamiento flexible que está basado en la asignación de flujos en rutas punto–punto.

[1] La manera en que funciona es anexando un encabezado a cada paquete, y el encabezado contiene uno o más etiquetas.

En MPLS la asignación de un determinado paquete a un FEC[2] (*Forwarding Equivalence Class*, Clase Equivalente de Envío), se hace una sola vez y está codificado por un valor conocido precisamente como etiqueta, en este modelo los paquetes son etiquetados antes de enviarse al siguiente salto [3].

Los Diff-Serv se caracterizan por marcar los paquetes IP y son tratados por la red tomando en cuenta su prioridad. Una red de nodos de este tipo de servicios no mantiene estados de las conexiones por flujo de paquetes, es aplicable a grandes entornos como Internet a diferencia de los servicios integrales y se basa en la separación de conceptos básicos de operación como son el control y los enrutadores de reenvío (forwading) conocidos comúnmente estos últimos como enrutamiento (routing) o ruteo [4].

En la presente tesis se estudiarán los Diff-Serv debido a las ventajas que presenta, así mismo se aplicará QoS en Linux para optimizar el tráfico de una red IP con el apoyo de sus herramientas IProute2[3] y Netfilter[4]. IProute2 es la herramienta que permite configurar y gestionar el ancho de banda de una red, por medio del sistema tc (*Traffic Control*, Control de Tráfico) de Linux, Por su parte, Netfilter es una herramienta que permite filtrar paquetes o cambiar sus cabeceras, una capacidad especial es que se puede marcar un paquete con un número. El marcado se hace con la herramienta IPtables y posteriormente los paquetes son filtrados según los valores con los que se marcan, para así determinar si pasa, se rechaza o se pone en cola [URL, 2].

[2] Conjunto de paquetes que comparten las mismas características para su transporte.
[3] http://devresources.linux-foundation.org/dev/iproute2/download/ - Último acceso: Septiembre 2008.
[4] http://www.netfilter.org/ - Último acceso: Septiembre 2008.

Se trabajará con Linux por ser un Sistema Operativo que además de ofrecer QoS, a partir de su versión 2.4 del Kernel[5], el cual incluye un subsistema de red y proporciona herramientas como el routing, firewalls y código de clasificación de tráfico, además de ser un sistema operativo basado en Unix, por lo que se considera también como multiusuario, multitarea, multiplataforma, multiprocesador, es robusto y estable. Existe documentación libre para su conocimiento, cuenta con licencia GPL (*General Public License*, Licencia Pública General), se pueden instalar y desinstalar programas de aplicaciones para los usuarios finales sin tener que reiniciar el equipo, también tiene la capacidad de establecer mecanismos para que los usuarios comuniquen bugs, publicarlos, corregirlos y poner los parches correspondientes.

Al final del trabajo se hará un análisis del comportamiento de la red antes y después de implementar dichas herramientas para determinar si efectivamente QoS logra un buen funcionamiento en la asignación del ancho de banda.

Este trabajo de tesis se divide en cinco capítulos. En este primer capítulo se presenta el planteamiento del problema, la justificación para la realización de este trabajo y objetivos del mismo. Como primer tema, en el capítulo dos, se tiene la descripción general de Qos, protocolos y mecanismos, así como la relación del tráfico en Linux para la manipulación de ancho de banda. En el capítulo tres se describe a IProute2, las disciplinas de colas con las que trabaja y se presenta como enrutar los diferentes tipos de tráfico en las conexiones y así, maximizar el uso del ancho de banda disponible. El capítulo cuatro trata sobre la herramienta Netfilter para filtrar paquetes y se muestran los

[5] http://www.kernel.org -Último acceso: Septiembre 2008.

comandos e instrucciones para lograr tal objetivo. La configuración e implementación para aplicar QoS se mostrará en el capitulo cinco, donde, se crearan las clases y la asignación de prioridades, tanto en Linux como en OpenBSD[6], también se muestra los resultados en una tabla y gráfica comparativas del tráfico antes y después de haber usado las herramientas Iprotue2, Netfilter y PF. Finalmente se dan a conocer las conclusiones de esta tesis, así como el trabajo a futuro.

1.2 Planteamiento del problema

El mal uso de los recursos de una red provoca la saturación del ancho de banda, provocando los llamados cuellos de botella, lo cual hace que sea imposible la salida de cualquier tipo de datos, afectando las actividades de los usuarios de la red a la que pertenecen. Entre los usos inadecuados se pueden mencionar algunos como:

- Utilización de canales de chat.
- Transmisión y recepción (Tx/Rx) de audio y video en tiempo real.
- Envío masivo de correo y con archivos atados de gran tamaño.
- Descargas de archivos con extensión mp3, mpeg, mpg, avi, wma, ram, entre otros.

Ante este problema, se buscan medidas que ofrezcan un mejoramiento en la optimización de los servicios de una red, es por ello que actualmente la administración de una red resulta de gran importancia para las organizaciones a fin de hacer un buen uso de

[6] http://www.openbsd-org - Último acceso: Septiembre 2008.

su ancho de banda, lo cual contribuya a efectuar con mayor rapidez sus actividades. Para ello, los recursos de una red deben ser asignados a los usuarios considerando previamente sus necesidades.

Una de las acciones encaminadas a la optimización de los recursos de una red es incrementar el ancho de banda, aunque esto implica un mayor costo, pero no es una buena solución ya que a mayor ancho de banda, es mayor la demanda por parte de los usuarios. Otra alternativa es utilizar los recursos o herramientas que actualmente ofrece el sistema operativo Linux [5], el cual permite implementar medidas robustas como:

- Proteger una red de ataques DoS (*Deny of Service*, Denegación de Servicio).
- Regular el ancho de banda de una interfaz de red.
- Repartir el ancho de banda, tomando en cuenta multitud de criterios de acuerdo a las funciones y necesidades.
- Hacer el enrutado de acuerdo a los siguientes criterios: usuario, dirección MAC, tipo de servicio, hora del día, etc.
- Implementar facilidades como el balanceo de carga o disponibilidad mejorada en varios servidores.

Las anteriores medidas son posibles gracias al nuevo código del Kernel de Linux, que proporciona enrutado, filtrado y clasificación, ofreciendo así, más posibilidades y facilidades que muchos otros routers, firewalls dedicados y productos de control de tráfico que existen [URL, 2]. A través de dicho código es posible implementar QoS en Linux

mediante sus herramientas IProute2 utilizando las tablas de enrutamiento y Netfilter con IPtables, a fin de optimizar los recursos de una red mediante un adecuado control de tráfico.

Para llevar a cabo el control de tráfico, se deben conocer los conceptos de filtros y colas. Los filtros ponen los paquetes en las colas, y las colas deciden que paquetes mandar antes que otros, las colas determinan el orden en que se mandan los paquetes.

1.3 Justificación

Debido al creciente avance en Internet y al surgimiento de nuevas necesidades por parte de los usuarios, hoy en día se buscan nuevas herramientas que contribuyan a evitar un desperdicio de ancho de banda y elevar la calidad del servicio de una red. Actualmente existen diferentes herramientas con las cuales es posible lograr una adecuada distribución del tráfico para optimizar los recursos de una red. Las herramientas existentes para lograr dicha distribución se dividen como se ve en siguiente tabla:

Tabla 1.1. Formas de aplicar QoS

Por hardware	Por software
Routers	Utilizando Linux son: IProute2 apoyándose con Netfilter
Gateways	En Solaris con Solaris Bandwidth Manager
Switchings	En OpenBSD con AltQ y recientemente con PF (PacketFilter, Filtrado de paquetes), por mencionar algunos.

Para efectos de la presente tesis, se desarrollará una investigación encaminada a optimizar el rendimiento de la red implementando QoS, a través de las herramientas de Linux (IProute2 y Netfilter). Se ha elegido Linux, debido a que es un sistema de libre distribución, lo cual representa una de las grandes ventajas comparada con otros sistemas tales como Microsoft Windows Server, Unix, NetWare de Novell. Además Linux es un sistema estable, versátil y potente. [URL, 3] Y también porque es el sistema que actualmente está implementado en la red en la cuál se aplicará QoS.

Las herramientas de Linux (IProute2 y Netfilter) tienen como función llevar a cabo el control de tráfico y filtración de paquetes. Por lo que, al implementar Qos en Linux se conocerá el funcionamiento de cada una de dichas herramientas y se demostrará cual es el comportamiento de una red IP después de aplicar QoS, con el propósito de optimizar los recursos de una red y ofrecer un mejor servicio a los usuarios.

1.4 Objetivos

OBJETIVO GENERAL

Analizar las herramientas IProute2 y Netfilter para marcar la prioridad de servicio en los paquetes de información trasmitidos en sistemas Linux.

OBJETIVOS ESPECIFICOS

- Definir y explicar el funcionamiento de la QoS en el sistema operativo Linux.

- Listar y mostrar las características generales de algunos modelos para satisfacer la QoS en redes.
- Mostrar las características principales de los Servicios Diferenciales en Internet.
- Mostrar las características principales de los Servicios Integrales en Internet.
- Describir el funcionamiento de las herramientas IProute2 y Netfilter.
- Mostrar las diferencias y ventajas de las tablas de enrutamiento de IPtables.
- Comparar IProute2 y Netfilter para verificar si trabajan independientemente o se complementan al aplicar QoS.
- Configurar IProute2 y Netfilter y realizar pruebas para verificar su buen funcionamiento dentro de la red.
- Comparar las diferencias encontradas en los resultados en cuanto al rendimiento de la red aplicando las herramientas mencionadas en el objetivo anterior y en la herramienta PF del OpenBSD.

Capítulo 2

2.1 Introducción

Debido al crecimiento y avance tecnológico en el mundo de Internet, los sistemas informáticos ya no solo se basan en una red de datos, sino que ahora también se hace uso de voz y video para su Tx/Rx en tiempo real, por lo que es necesario la implementación de QoS, cuya tarea primordial es asegurar determinadas características como la calidad y confiabilidad en la transmisión de la información evitando el congestionamiento de la red.

QoS se refiere a funciones aplicadas para proveer un tratamiento preferencial para determinados servicios, dichas funciones se esfuerzan por controlar el uso adecuado de dichos servicios y el ancho de banda.

Actualmente existen varios sistemas operativos, cada uno de ellos se utiliza tomando en cuenta las características (soporte y robustez) del equipo donde se va a ejecutar, ya que el sistema operativo es el que se encarga de controlar los recursos y procesos de hardware y software de una computadora.

Como se mencionó anteriormente se trabajará con el sistema operativo Linux, se nombraron algunas características pero cabe resaltar que por su gran capacidad para

trabajar en red y que está bajo la licencia Pública General de GNU, que es la licencia usada por el software desarrollado por la fundación de software libre que permite que cualquier persona modifique y redistribuya dicho software, es importante señalar que se le llama software libre no por el costo sino por que ofrece su código para mejorarlo y/o modificarlo de acuerdo a las necesidades de los usuarios [6].

Hoy en día existen varias distribuciones de Linux, en sí una distribución es el agrupamiento del núcleo del sistema operativo y otros programas de utilidad, entre las distribuciones más conocidas están: Red-Hat, Debian, Slackware, Suse, Ubuntu. Por tal motivo el sistema operativo usado en el desarrollo de esta tesis es Linux, utilizando la distribución Slackware[7] 10.2 con su Kernel 2.6.13, la distribución se elige tomando en cuenta su flexibilidad para enrutar paquetes a redes diferentes y por ser la distribución en la que se tiene mayor experiencia para su instalación y configuración.

En este capítulo se abordará sobre el funcionamiento de QoS en Linux, así como la identificación de los diferentes mecanismos y herramientas que hay para implementar QoS en la administración de los servicios en las redes, al final del capítulo se espera conocer cada uno de estos conceptos con los cuales se tendrá mayor conocimiento sobre el sistema operativo Linux identificando las ventajas que se tienen al explotar y utilizar en forma correcta sus diferentes herramientas con que cuenta.

[7] http://www.slackware.org -Último acceso: Septiembre 2008.

2.2 Descripción general de QoS

El término QoS ha sido definido por distintos autores, los cuales coinciden en que se trata de una relación entre usuarios y las diferentes capacidades de servicios en una red, algunas de estas definiciones son:

QoS: “Es la capacidad de la red de proporcionar un mejor servicio al tráfico seleccionado” [URL, 4].

QoS: “Es la medida de la bondad del comportamiento de la red con respecto a ciertas características de servicios definidos” [URL, 5].

QoS: “Es la capacidad que tiene un sistema de asegurar con un grado de fiabilidad preestablecido que se cumplan los requisitos de tráfico en términos de perfil y ancho de banda para un flujo de información dado” [URL, 6].

La definición que da el ISO (*The Organization International for Standardization*, La Organización Internacional de Estándares) es: “QoS es el conjunto de especificaciones o cualidades relacionadas en la provisión de un servicio” [7].

Tomando en cuenta estas definiciones se puede decir que:

QoS es la capacidad de la red que permite regular el tráfico tanto entrante como saliente, para poder ofrecer ciertos servicios a los usuarios, logrando así un mejor rendimiento de la misma red.

Al aplicar QoS en una red, se tienen las siguientes ventajas:

- Asignación de ancho de banda.
- Evitar y/o administrar la congestión de la red.
- Manejo de prioridades a través de la red.
- Modelado del tráfico en la red.

En Linux la manera de ofrecer QoS se hace a través de su Kernel, y el procedimiento inicia cuando los paquetes llegan al demultiplexor el cual se encarga de enviar el paquete a la capa superior en la pila de protocolos, siempre y cuando detecte que los paquetes son destinados al nodo local, en caso contrario el paquete se envía al bloque de reenvío, el cual también puede recibir paquetes de la capa superior, los paquetes que son reenviados son encolados en un dispositivo de colas, la cola es seleccionada por el control de tráfico (tc) basándose en los requerimientos de QoS. Tal como se muestra en la siguiente figura [URL, 7].

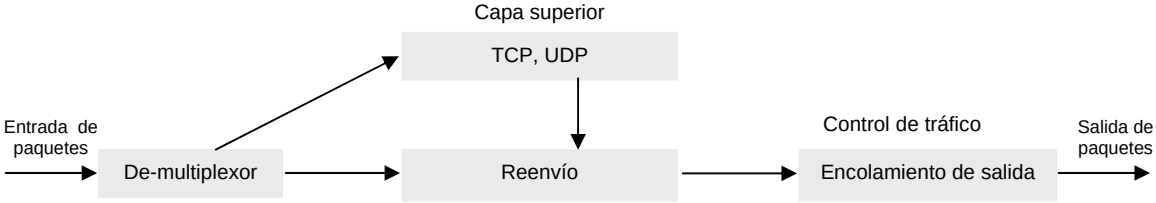


Figura 2.1. Funcionamiento de QoS en Linux

Para poder implementar QoS en una red es importante conocer el tipo de usuario que se tiene, y saber cuáles serán las aplicaciones y/o servicios necesarios para realizar el trabajo de dichos usuarios, esto es con el fin de poder hacer una diferenciación de tráfico y así establecer prioridades. Posteriormente se debe valorar el sistema operativo sobre el que se va a trabajar, tomando en cuenta sus ventajas y desventajas tanto en funcionalidad como en su desempeño, actualmente uno de los sistemas operativos más usado es Linux por sus cualidades administrables que tiene.

2.2.1 Protocolos de QoS

Los protocolos facilitan el establecimiento de una comunicación proporcionando a los miembros un lenguaje común [8].

Existen varios protocolos para cubrir las diferentes necesidades de QoS, algunos de ellos son:

- **RSVP,** el cual permite a las aplicaciones solicitar QoS y consiste en hacer reserva de recursos en los nodos por donde pasa cada flujo de información.
- **Diff-Serv,** se basa en el marcado de paquetes para ser diferenciados priorizando así el tráfico y tratar a cada paquete tomando en cuenta su prioridad.

- **MPLS,** se agrega una etiqueta en la cabecera de los paquetes y efectúa la conmutación en base a ella, otra característica de este protocolo es que permite controlar el ancho de banda asignado a una sesión.

- **SMB** (*Subset Bandwidth Manager*, Gestor de ancho de banda de subred)**,** este es necesario para mantener la QoS en los enlaces IP sobre Ethernet o Token Ring compartido o conmutados [URL, 8].

Todos estos protocolos son de alguna manera complementarios, existen varias arquitecturas donde dichos protocolos suministran QoS de manera conjunta, y depende de las necesidades que se quieran cubrir, en una administración de red se podrán usar un protocolo o la combinación de ellos para solucionar el problema.

2.2.2 Mecanismos de QoS

Los mecanismos de QoS sirven para la asignación de recursos entre los usuarios, además de que son necesarios para cubrir las necesidades de QoS de las diferentes conexiones, por que permiten establecer las estrategias de administración de cola que implementan prioridades para diferentes clasificaciones de datos de aplicación, si no se cuenta con una implementación correcta de los mecanismos de QoS, los paquetes de datos se descartan sin tomar en cuenta las características de la aplicación ni la prioridad, a continuación se explicarán algunos mecanismos de QoS.

- **Mecanismos de prioridad,** se refieren a la capacidad de los diferentes tratamientos al retardo, es decir los paquetes de mayor prioridad son los primeros en liberarse antes de los de menor prioridad.

- **Mecanismos de gestión,** se aplican en los nodos para asegurar que las conexiones obtengan los recursos prometidos tanto en procesamiento como en ancho de banda.

- **Mecanismos de identificación de tráfico,** se aplica antes de hacer el marcado de paquetes y habitualmente esta identificación se hace mediante la dirección origen y dirección destino, número de puerto, tipos de protocolos TCP, UDP.

- **Mecanismos de marcado de paquetes,** este tipo de mecanismos se refiere a diferenciar los tipos de tráfico usando una etiqueta para marcar, las diversas formas de marcado se hacen según se vayan a realizar sobre la cabecera de Nivel 2 o Nivel 3, documentado en [9] y [10].

- **Mecanismos de ordenamiento de paquetes,** selecciona un paquete para su transmisión desde la cola de paquetes, estos mecanismos deciden que paquete y desde que cola y estación están ordenados para su transmisión en un cierto periodo de tiempo, además controlan el ancho de banda asignado a estaciones, clases y aplicaciones.

- **Mecanismos de control de tráfico,** controlan las sesiones de tráfico admitidos para que no violen las reglas de QoS, asegura que todo el tráfico que pase a través de este siga las reglas conforme a los parámetros del tráfico, este mecanismo se complementa con el manejo del Mecanismo Leaky Bucket; el cual

es un mecanismo para modelar el tráfico, consiste en no enviar directamente los paquetes sino que los encola y los envía a una tasa constante, utiliza un algoritmo el cual funciona como un regulador, cada “n” segundos, el recipiente o balde libera un paquete, pero si hay un rebase del límite en el recipiente se descarta el paquete entrante [1].

2.2.3 Herramientas de QoS

Actualmente las herramientas disponibles para ofrecer QoS son los sistemas de colas y priorización de datos, los cuales tienen como objetivo hacer una distribución del tráfico para incrementar la eficiencia de la red. Las herramientas mas conocidas para llevar a cabo esto son las siguientes:

FIFO (*First In, First Out*, Primero que entra, Primero que sale), los paquetes son enviados en el mismo orden en el que llegan a la interfaz; y es válida en redes con poca congestión, es el mecanismo con que se trabaja por default en una red IP.

PQ (*Priority Queuing*, Cola con prioridad), da prioridad estricta al tráfico importante, se basa en la prioridad del tráfico de varios niveles el cual puede aportar el encabezado del paquete IP (ToS), estos niveles pueden ser: alto, medio, normal y bajo. Los paquetes con una prioridad alta son transmitidos primero que los que tienen una prioridad baja.

CQ (*Custom Queuing*, Encolamiento personalizado), garantiza el ancho de banda mediante una cola de espera programada, reserva un porcentaje del ancho de banda disponible para cada tipo de tráfico seleccionado y si un determinado tipo de tráfico no está utilizando su ancho de banda este puede ser usado por otro tráfico, su configuración es complicada.

WFQ (*Weighted fair queuing*, Cola ecuánime ponderada o por peso), asigna una prioridad a cada flujo de manera que determina el orden de tráfico en la cola de paquetes, la ponderación menor es la que despacha primero; dicha ponderación se lleva a cabo utilizando la dirección de origen y destino, tipo de protocolo en IP, puerto y por el ToS en el protocolo IP.

RED (*Random Early Detection*, Gestión de la congestión), evita congestión, se encarga de monitorear el tráfico al azar, si la congestión aumenta elimina paquetes, en otras palabras al darse cuenta del aumento de tráfico, disminuye la velocidad de transmisión de paquetes. Este mecanismo es útil cuando se almacena tráfico a ráfagas por que previene la eliminación o descarte de los últimos paquetes que entran en una cola llena.

WRED (W*eighted Random Early Detection*, Gestión de la congestión mejorada), es una combinación de RED con *IP Precendence*, descarta paquetes de forma aleatoria basándose en el valor de *IP Precedente*, los paquetes con mas bajo valor de precedencia

son los descartados, es mas avanzado que el RED y depende del mecanismo de clasificación para marcar los paquetes con diferentes prioridades de descarte.

GTS (*Generic Traffic Shaping,* Regulación del ancho de banda), retrasa el exceso de tráfico al usar buffers para modelar el flujo, con lo que reduce el tráfico saliente limitando el ancho de banda de cada tráfico específico y si se excede el ancho de banda establecido es enviado a una cola de espera.

La implementación de QoS en Linux se hace mediante las herramientas IProute2 y Netfilter, las cuales tienen que trabajar de manera conjunta; ambas herramientas se complementan, a continuación se explica la manera en que trabaja cada una de ellas.

El IProute2 tiene las herramientas *ip* y *tc* que permiten controlar el tráfico mediante disciplinas de colas (*Queueing Disciplines* o *qdisc*), clases y filtros. Por su parte la herramienta Netfilter utiliza un sistema para filtrar, marcar y seleccionar los paquetes, empleando tablas, este sistema es lo que se conoce como IPtables.

A continuación se mostrarán las diferentes disciplinas de colas, las cuáles se describirán en el siguiente capítulo.

Disciplinas de colas sin clase

- Pfifo-fast.
- TBF(Token Bucket Filter).
- SFQ(Stochastic Fairness Queueing).

Disciplinas de colas con clase

- Qdisc PRIO (Cola con prioridad).
- CBQ (Class Based Queueing).
- HTB (Hierarchical Token Bucket).

2.3 Redes en Linux

Linux desde un principio fue dotado de capacidad de red, los cambios en las implementaciones de red para su Kernel o núcleo son rápidos y continuos, en un principio el Kernel estándar requería de parches, para así poder dar soporte a redes, hoy esto forma ya parte del mismo Kernel, debido a que es el flujo principal del desarrollo de Linux.

Se puede encontrar en Internet una infinidad de sitios para poder obtener las distintas versiones del Kernel, las versiones a partir de la 2.4.x son las que ya cuentan con herramientas y características para poder llevar a cabo una buena administración de una red, controlando su ancho de banda y su tráfico, el cual se implementa tanto en redes

LAN como en redes WAN. A continuación se explican brevemente los componentes que intervienen en el intercambio de datos entre los equipos que conforman una red.

- **Protocolos,** son reglas de red que describen como se comunican entre si las computadoras.

- **Software de red,** son aplicaciones que implementan protocolos y se encarga de comunicar al hardware de red con otras aplicaciones.

- **Hardware de red,** son dispositivos que envían y reciben datos entre computadoras y redes.

Una de las características mas importantes para que haya comunicación entre las computadoras es el protocolo de red y el más popular actualmente es el TCP/IP. Linux soporta el TCP/IP en su Kernel y su distribución cuenta con software necesario para configuración, prueba y monitoreo de conexiones TCP/IP, así como muchas aplicaciones que usan este protocolo (navegadores Web, mail, FTP, etc), además cuenta con soporte en hardware, por ejemplo para dispositivos de red (tarjetas de red, modems, adaptadores ISDN).

El TCP/IP se basó en el modelo OSI (*Open System Interconnection* Interconexión de Sistemas Abiertos), con la diferencia de que nunca trató de ser un estándar

internacional, las capas del TCP/IP se conceptualizan como similares a las capas del OSI debido a que muchas de sus funciones son parecidas [4].

2.4 Manipulación de ancho de banda

Para manipular el ancho de banda en una red se usa el Traffic Shaper, que viene en Linux y que sirve para restringir el ancho de banda en nuestras máquinas, el cual es un regulador que viene contenido en el Kernel a partir de la versión 2.2 esta herramienta se debe compilar como módulo para ponerlo a funcionar.

Para configurarlo se necesita del programa shapecfg el cual es también una herramienta de Linux, este es simple pero provee un control preciso sobre el ancho de banda del tráfico de salida, el problema es que solo puede configurar una interfaz a limitar y no se podrá limitar por IP, debido a esta deficiencia se debe hacer uso de otras herramientas en este caso IProute2 y Netfilter.

Capítulo 3

3.1 Introducción

En este capítulo se hablará sobre la herramienta IProute2 que permite llevar un mejor control y distribución del tráfico en una red, para tal efecto se debe conocer como funciona el campo DS (*Diferencial Service,* Servicio Diferencial), su estructura y la forma en que trata los paquetes en la red. Cabe mencionar que también se verá de manera detallada las diferentes disciplinas de colas mencionadas en el capítulo anterior explicando algunas de las herramientas para clasificar los paquetes en las redes IP.

El sistema operativo Linux, en sus recientes versiones ha incorporado el conjunto de mecanismos de control de tráfico que han sido definidos para Diff-Serv. Tales como disciplinas de servicio y clases, medidores o estimadores, filtros, verificadores de conformidad (policing), y conformadores (shapers).

Cada uno de estos elementos tiene un papel importante a la hora de tratar los paquetes en la red y juntos forman lo que se conoce como el condicionamiento de tráfico, el cual se describirá en el desarrollo de este capítulo. Y para conocer las diferentes formas en las que se pueden manipular los paquetes se necesita saber cuales son las diferentes disciplinas de colas.

Al final del capítulo se conocerá sobre el funcionamiento de las herramientas que utiliza el IProute2 para aplicar QoS, se identificará la diferencia entre el campo ToS y el campo DS, así como también se tendrá un mejor conocimiento sobre la importancia del valor DSCP (*Differentiated Services Codepoint*, Punto de Codificación de Servicios Diferenciados) y su relación con los paquetes a la hora de hacer la clasificación de tráfico.

3.2 IProute2

IProute2 es un conjunto de comandos a través de los cuales se manipula la estructura del Kernel para llevar a cabo la distribución de tráfico de manera que se garantice QoS en una red.

Lo que se pretende hacer con el IProute2 es mantener la configuración de red y rutear diferentes tipos de tráfico en las conexiones para maximizar el uso del ancho de banda disponible, para poder hacer esto el IProute2 se ayuda de sus herramientas que son ip (Internet Protocol, Protocolo de Internet) y tc, ip se encarga de monitorear las direcciones, interfaces, gestión de tablas ARP (*Address Resolution Protocol*, Procolo de resolución de direcciones), uso de múltiples tablas de enrutamiento y la creación de túneles IP. El tc, se encarga de llevar a cabo el control y manipulación del ancho de banda y prioridad, asi como, de realizar toda la configuración del Kernel requerida para soportar el control de tráfico [URL, 9].

En resumen el tc se usa para hacer la configuración de las colas y clases, estos componentes se combinan para controlar el flujo de paquetes entrantes y salientes en una

interfaz de red. Para hacer uso de estas herramientas es necesario tener activadas algunas opciones del Kernel, debe tener compiladas las opciones de *IP: advanced router* e *IP: policy routing* que se encuentra en la sección de: *Networking options*, como se muestra en las Figuras 3.1 y 3.2.

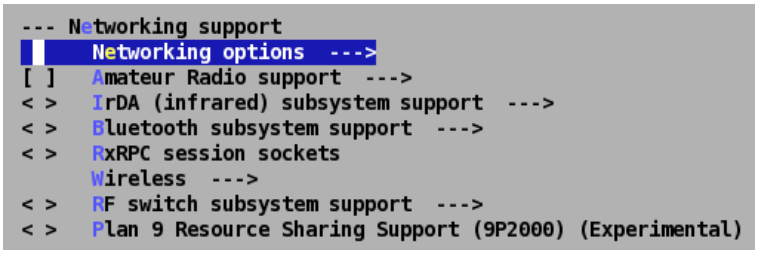


Figura 3.1. Configuración del Kernel - Networking options

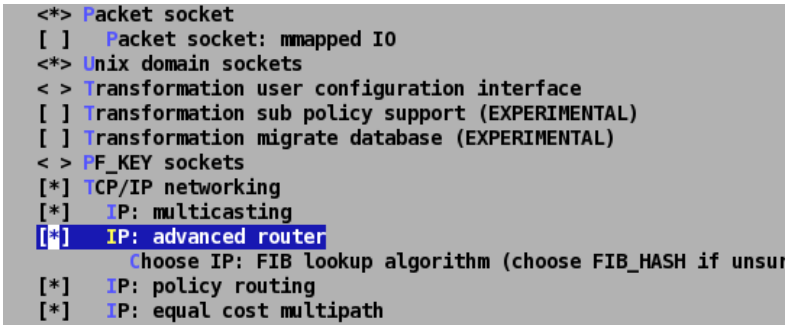


Figura 3.2. Activación de Advanced router y policy routing en el Kernel

3.3 Campo DS

La arquitectura de Diff-Serv se desarrolló en respuesta a la necesidad de ofrecer los métodos de niveles de servicio para el tráfico de Internet, para soportar varios tipos de aplicaciones y requerimientos en particular.

Los Diff-Serv difieren de los Int-Serv en la manera de tratar el tráfico; los Int-Serv asignan recursos para un flujo individual y los Diff-Serv dividen el tráfico dentro de un pequeño número de clases y asigna recursos basándose en clases.

El modelo de Diff-Serv consiste en definir clases de servicio, cada flujo particular de datos se agrupa en un tipo de clase donde son tratados de la misma forma en cada nodo, a este proceso se le conoce como PHB (*Per Hop Behavior*, comportamiento por saltos), en donde la diferenciación de servicios se hace tomando en cuenta las políticas y grupos de comportamiento específicos para cada clase de tráfico [URL, 10].

Este comportamiento está marcado en la cabecera del paquete y su valor determina el tipo de tratamiento que se le dará en cada nodo, su estructura es una reorganización del byte ToS, para poder hacer la diferenciación de servicios se define un campo que sustituye lo que sería el campo ToS en IPv4 y el TC del IPv6, y este es conocido como DS [11]. La descripción de cada subcampo del ToS se ve en la tabla 3.1.

Tabla 3.1. Descripción de los subcampos del campo ToS

Bit	Descripción
0-2	Nivel de precedencia
3	Diferenciación de tráfico, retardo, rendimiento y fiabilidad
4	
5	
6	Costo
7	Reservado (sin uso)

A continuación se hace una comparación del campo ToS en la Figura 3.3 y del campo DS en la Figura 3.4.

Estructura del campo ToS (RFC 1349)

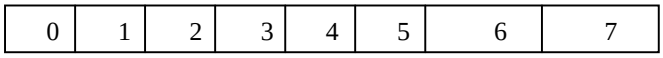


Figura 3.3. Byte ToS

Estructura del campo DS (RFC 2474)

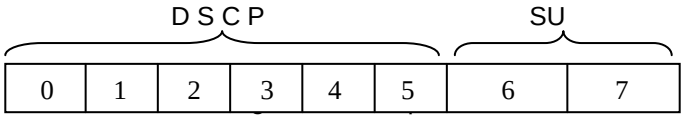


Figura 3.4. Campo DS

Donde DSCP son los primeros 6 bits más significativos del campo ToS, y los dos últimos bits están reservados para su uso.

Cada PHB está representado por un valor de 6 bits llamado DSCP, el bit menos significativo es el 0 y el más significativo es el 5. El campo DS puede tener hasta 64 distintos valores. El IETF propone hasta 32 valores como clases de servicio global y las otras 32 se dejan abiertas para una definición específica en la red. Existen dos tipos de PHBs. que son:

- EF PHB (*Expedited Forwarding,* Encaminamiento Expedito), minimiza el retardo en la variación del mismo entre paquetes [12], y se usa para tratar el tráfico de tiempo real.

- AF PHB (*Assured Forwarding,* Encaminamiento Asegurado), se refiere al tráfico HTTP, FTP, TELNET, SMTP, el cual se distribuye en 4 clases diferentes que son AF1, AF2, AF3 y AF4 con 3 prioridades en cada uno [13].

El valor DSCP otorga a determinados paquetes cierta prioridad sobre otros en cada nodo, reduciendo así la latencia e incrementando el paso de paquetes válidos para flujos de tráfico específicos.

Para representar los 6 bits del valor de un DSCP, se usa la notación xxxxxx, donde los tres primeros bits garantizan la coexistencia del valor de Precedencia de IPv4, pudiendo ser 001, 010, 011 ó 100, y se clasifican en 3 grupos según su uso [9] como se muestra en la tabla 3.2.

Tabla 3.2. Asignación de codepoints

Codepoint	USO
xxxxx0	Estándar
xxxx11	Local / Experimental
xxxx01	Reservado

Todos los paquetes con el mismo valor DSCP son enviados como BA (*Behavior Aggregate*, comportamiento agregado) y reciben el mismo tratamiento. Existen tres formas de usar el campo DSCP que son el clasificador, el marcador y el medidor, a su vez estas formas corresponden a lo que es la clasificación y condicionamiento de tráfico dentro de los Diff-Serv [1], como se ve en la Figura 3.5.

1. **Clasificador,** se encarga de seleccionar los paquetes, tomando en cuenta el contenido de la cabecera del paquete y le aplica PHB basándose en las características de servicio definidas por el valor DSCP.

   Existen dos clasificadores que son:

- **BA,** que ya se describió anteriormente y clasifica en función del DSCP.

- **MF** (*Multi-Field*, Multi-Campo), la selección de los paquetes los realiza basándose en la combinación de uno o mas campos como: dirección fuente, dirección destino, puerto fuente, puerto destino, identificador de protocolos, etc.

2. **Marcador,** es un conjunto del campo DSCP basado en el traffic profile (Perfil de tráfico). El traffic profile, especifica las propiedades del tráfico seleccionado por el clasificador y provee reglas para determinar si un paquete se ajusta o no a ellas.

3. **Medidor,** mide el tráfico usando el shaper (conformador de tráfico) y el dropper (descarte).

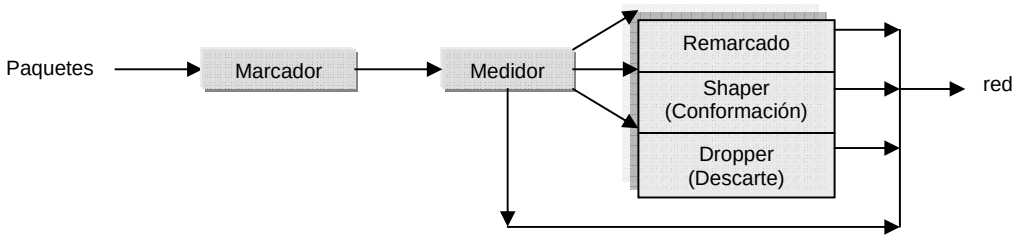


Figura 3.5. Clasificación y condicionamiento de tráfico

Algunas de las ventajas más importantes que se tienen al utilizar el modelo Diff-Serv con respecto a los Int-Serv son:

- Los enrutadores solo se limitan a la clasificación y el encolado.
- Se minimiza el tráfico de señalización y almacenamiento (hay menos carga), y
- No hay reservas sino prioridades.

3.4 Disciplinas de colas (qdisc)

Las colas determinan el orden en que se envían los paquetes basándose en ciertas características, el algoritmo de qdisc gestiona el encolamiento de paquetes en la tarjeta de red, tanto entrantes (*ingress*) como salientes (*egress*).

El encolamiento por defecto cuando se envían los paquetes a través de la red es una FIFO, pero este tipo de encolamiento no es recomendable cuando se tiene demasiado tráfico ya que puede haber paquetes de alta prioridad después de paquetes de baja prioridad o no tan urgentes de entregar y hay que esperar el turno de salir, provocando una latencia muy elevada. Las disciplinas de colas se dividen en:

Disciplinas de colas sin clase o simple. En las disciplinas de colas sin clase sólo se pueden aceptar, reordenar, retrazar y descartar los datos, no acepta una subdivisión interna. Los principales tipos de colas que existen se definieron en el capítulo dos y en esta sección se describe cada una de ellas.

- **pfifo_fast,** consta de tres bandas de prioridades en las cuales los paquetes son procesados tomando en cuenta el ToS para determinar así en que banda va cada paquete, cabe mencionar que no se puede enviar ningún paquete de otra banda si todavía hay paquetes por enviar en cualquiera de las otras dos restantes. Las prioridades no se pueden modificar ya vienen por default y por esto se les considera sin clase y están definidas como; prioridad 0, prioridad 1 y prioridad 2.

- **TBF,** deja pasar paquetes que no se excedan de una tasa impuesta por el administrador, solo limita un ancho de banda concreto permitiendo ráfagas pequeñas a mayor velocidad.

- **SFQ,** esta qdisc envía datos de manera equitativa entre todas las conexiones evitando que alguno de los usuarios acapare todo el ancho de banda. El tráfico se divide en un gran número de colas FIFO una por cada sesión TCP o un flujo UDP, dando a cada una la oportunidad de enviar sus datos o paquetes por turnos.

A esta disciplina, se le llama estocástica por que no crea una cola para cada sesión sino divide el tráfico en un número limitado de colas usando un algoritmo de hash[8].

Disciplinas de colas con clase. Este tipo de disciplinas contiene múltiples clases internas las cuales algunas pueden tener otra qdisc que pueden ser con clase o simple, este tipo de disciplinas son útiles cuando se tienen diferentes tipos de tráfico y se les quiere dar un tratamiento por separado. Esta qdisc utiliza un clasificador para determinar

[8] Divide en partes o trozos el tráfico y se utiliza para generar un valor de hash para cada parte.

a qué clase enviará cada paquete que llega, por medio de filtros que contienen los requisitos o condiciones que deben cumplir los paquetes y así saber a qué clase están destinados.

Una clase puede tener como padre una qdisc u otra clase, las clases que no tienen clases hijas son llamadas clase terminal o clase hoja (*leaf class*), cuando se crea una clase se le adjunta una qdisc FIFO, siempre y cuando no tenga clases hijas, esto se puede apreciar en la Figura 3.6.

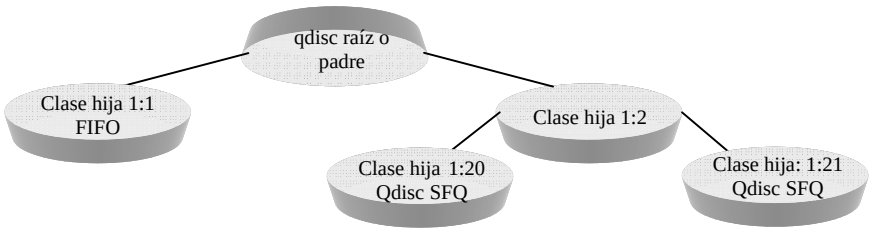


Figura 3.6. Disciplinas de colas con clase

Las principales qdisc con clase son:

- **PRIO,** esta qdisc subdivide el tráfico pero no hace ajustes, por defecto se crean tres clases, las cuales contienen qdisc FIFO sin estructuras internas por lo que se considera a esta disciplina como una pfifo_fast, por tener el mismo comportamiento con la diferencia de que cada banda es una clase separada y se pueden definir los filtros pertinentes de tal manera que la clasificación no se base únicamente en el campo ToS, y se puedan cambiar las disciplinas con política FIFO que se tienen por defecto. A diferencia de las pfifo_fast aquí si se pueden

cambiar por otras qdisc en lugar de FIFO. Esta qdisc se usa cuando se necesita dar prioridad a cierto tráfico ayudándose de los filtros de tc. Esta qdisc al no hacer ajuste se usa de igual forma que la SFQ, solo cuando el enlace físico está realmente lleno.

- **CBQ,** es una qdisc que se basa en prioridades para enviar los paquetes. Es la disciplina más conocida pero también es la más compleja y difícil de configurar debido a que no se ajusta del todo a la forma de trabajar con Linux, esta qdisc trabaja con los tiempos de retardo en los paquetes, como ejemplo tenemos, que si se quisiera reducir una conexión de determinada velocidad, el enlace tendría que estar inactivo el 90% del tiempo pero esto es difícil de conseguir ya que pudiera ser que la verdadera velocidad del enlace no transmitiera realmente el total de datos; por lo que nunca se alcanzaría la velocidad otorgada, es aquí donde no se puede calcular el tiempo [URL, 11].

  El CBQ es útil para enviar paquetes a las clases hijas, en esta qdisc se pueden usar clases y subclases, se puede configurar el ancho de banda que se tiene disponible, dividiéndolo en forma jerárquica para los diferentes servicios ofrecidos, otorgándoles a cada uno de estos un ancho de banda garantizado, es decir se puede regular o ajustar el ancho de banda, como se muestra en la Figura 3.7, la asignación del ancho de banda que se hace se basa en clases, la cual propone la división y el compartimiento de forma jerárquica [URL, 7].

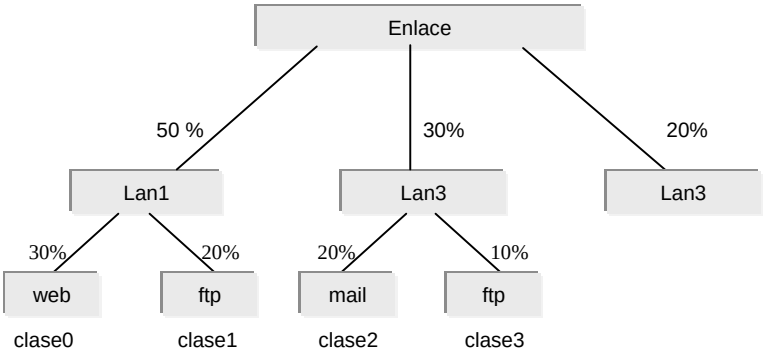


Figura 3.7. Compartimiento de ancho de banda en CBQ

Para utilizar la disciplina CBQ en Linux es por medio de la siguiente sintaxis.

DEVICE=ifname, bandwidth ,weight

Donde:

Ifname, es la interfaz de red donde se va a controlar el tráfico

Bandwidth, es el ancho de banda física del dispositivo o interfaz, p.e.:

Ethernet 10Mbit o 100Mbit, y

Weight, es el parámetro de sintonía que debe ser proporcional al ancho de banda.

$$\text{weight} = \frac{\text{bandwidth}}{10}$$

Por ejemplo; DEVICE=eth0,10Mbit,1Mbit

- **HTB,** esta qdisc es la mas sencilla de manejar, es ideal para cuando se necesita dividir un ancho de banda fijo en diferentes bandas, su función es básicamente igual a la de CBQ, pero es mas fácil de configurar debido a que no recurre a

cálculos de tiempo ocioso para ajustar la velocidad del tráfico. Esta qdisc tiene la característica de configurar una cola para tomar prestado el ancho de banda de la cola padre que la origina en caso de que esta no la esté usando totalmente, las colas tienen una prioridad de tal manera que las que tengan tráfico de mayor prioridad se atiendan antes que las de menor prioridad y las colas que tengan la misma prioridad se procesan usando el algoritmo Round Robin[9].

3.5 Clasificación de paquetes

La clasificación de paquetes se basa en reglas especificas o bien definidas, examinando los campos en las cabeceras de los paquetes como dirección origen, dirección destino, puerto destino, puerto origen, una vez que se identifica el paquete se utiliza un descriptor de tráfico cuya función es dividir por categorías los paquetes por medio de filtros, con el fin de determinar en que clase debe ir cada paquete.

Existen algunos filtros que son específicamente para determinadas disciplinas de colas, los filtros más conocidos por su uso son: **u32, fwmark** y **route**. La estructura para configurar un filtro consta de 3 partes; la primera de ellas es, especificación del filtro, después selector y por último la acción. La especificación del filtro se hace de la siguiente manera:

tc filter add dev IF [protocol PROTO]
[(preference|priority) PRIO]
[parent CBQ]

[9] Asigna a cada clase por turnos la oportunidad de usar el mismo ancho de banda disponible.

Dónde:

protocol, es el protocolo al que se aplicará el filtro.

preference/priority, establece la prioridad del filtro elegido.

parent, define la cima del árbol CBQ, es decir, define al nodo padre.

Esta sintaxis se usa en todos los filtros siguientes en los apartados 3.4.1, 3.4.2 y 3.4.3 por ser la manera general de configuración de un filtro.

3.5.1 Filtro u32

El filtro o clasificador u32 se basa en la cabecera de los paquetes, permitiendo muchos criterios para realizar el filtrado, y usando el algoritmo hash. Este filtro es una lista de registros formada por los campos: selector y acción.

El selector es el que se encarga de definir que bits se tienen que comparar en la cabecera del paquete, comparando patrones con el paquete procesado y en base a la primera coincidencia ejecutar una acción, como por ejemplo, enviarlo a una clase CBQ definida [URL, 2].

Un ejemplo real queda de la siguiente manera: [URL, 12].

```
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 match ip src 192.168.2.1/24 match ip dport 80 0xffff flowid 10:1
```

Esto quiere decir que el filtro seleccionará a los paquetes ip que tengan como dirección origen 192.168.2.1 y como puerto destino el 80.

3.5.2 Filtro fwmark

Este filtro se usa mediante Netfilter, y los paquetes son marcados con IPtables. Cuando se usa como cortafuego el Netfilter se puede marcar un determinado grupo de paquetes con IPtables con un número y después utilizar el filtro para clasificar los paquetes tomando en cuenta el número de marca. La sintaxis se muestra por medio del siguiente ejemplo:

```
# tc filter add dev eth1 protocol ip parent 1:0 prio 1 handle 6 fw flowid 1:1
```

El ejemplo anterior sirve para ajustar el tráfico de eth1 siempre y cuando venga de eth0.

3.5.3 Filtro route

Este tipo de filtro se basa en el resultado que se tiene al pasar los paquetes ip por la tabla de rutas, un ejemplo de cómo se puede usar este filtro es:

```
# tc filter add dev eth1 parent 1:0 protocol ip prio 100  route to 10 class id 1:10
```

En el ejemplo anterior todos los paquetes que vayan a la red destino eth1 coinciden con la clase id 1:10 y la red eth1 la definimos como 192.168.2.0 con un número de dominio 10, esto queda de la siguiente manera:

```
# ip route add 192.168.2.0/24 via 192.168.10.1 dev eth1 realm 10
```

Donde cualquier paquete que vaya a la red 192.168.2.0 es marcado con real 10 si son consultados por un filtro route.

Capítulo 4

4.1 Introducción

En el capítulo 4, se expone de manera detallada la herramienta Netfilter con sus componentes, así como también la herramienta mas usada y conocida que es el IPtables, la cual es de gran apoyo para analizar los paquetes que circulan por la red, y de esta forma saber que tratamiento deben tener los diferentes tipos de tráfico para mejorar el comportamiento de su desempeño. Además se menciona el uso de Netfilter enfocado a la creación de Firewalls o cortafuegos para la protección y control de la red tanto con el exterior como de manera local.

Es importante hacer notar la diferencia entre Netfilter e IPtables, porque muchas veces se puede confundir con respecto a lo que se puede hacer con cada una de estas herramientas, Netfilter se refiere al código de Kernel y se utiliza para hacer el filtrado de paquetes, administrar el estado de las conexiones, NAT por mencionar algunas de sus funcionalidades. Por su parte IPtables es una herramienta entorno al usuario con la cual se permite manipular el código del Kernel para configurar lo que es el filtrado.

Finalmente, se conocerá a detalle el funcionamiento de la herramienta Netfilter, exponiéndose los comandos más utilizados por ésta, que sirven para aplicar las reglas de

filtrado, así también se mostrará el comportamiento de los paquetes en la red, junto con los diferentes protocolos de enrutamiento a fin de conocer la relación que existe con la comunicación entre los enrutadores.

4.2 Definición

Netfilter es una herramienta que viene integrada en el Kernel de Linux a partir de la versión 2.4, se encarga de aplicar las reglas de filtrado, manipula los encabezados de los paquetes, su principal característica consiste en marcar paquetes con un número usando el siguiente parámetro:

--set-mark

Dentro de los comandos que tiene el Netfilter, existen los básicos que son usados por la herramienta IPtables y son los que se muestran en la tabla 4.1.

Tabla 4.1. Comandos básicos del Netfilter

Comando	Acción
iptables –N cadena	crea una cadena
iptables –A cadena	añade una regla a la cadena
–t tabla	para usar tabla (mangle, nat, filter)
–p protocolo	tcp,udp,icmp,etc.
–s dirección-fuente [/mask] --sport puerto [:port]	puerto de origen si –p es tcp o udp
-d dirección-destino[/mask] --dport puerto[:port]	puerto de destino si –p es tcp o udp
-j objetivo	qué hacer si coincide
-i nombre-interfaz-entrada -o nombre-interfaz-salida	para INPUT,FORWARD,PREROUTING para FORWARD,OUTPUT,POSTROUTING

Aplicando algunos de estos comandos en un ejemplo, sería de la siguiente manera: [URL, 2]

```
# iptables –A PREROUTING –i eth0 –t mangle –p tcp --deport 22 –j MARK --set-mark 1
```

Donde se indica el marcado de todos los paquetes que tienen como destino el puerto 22.

Netfilter se representa como una serie de cinco ganchos o cadenas en donde se insertar los módulos de tratamiento de los paquetes, es decir forman lo que sería el recorrido de los paquetes (ver Figura 4.1). Estos ganchos son:

1. NF_IP_PREROUTING
2. NF_IP_LOCAL_IN
3. NF_IP_FORWARD
4. NF_IP_POSTROUTING
5. NF_IP_LOCAL_OUT

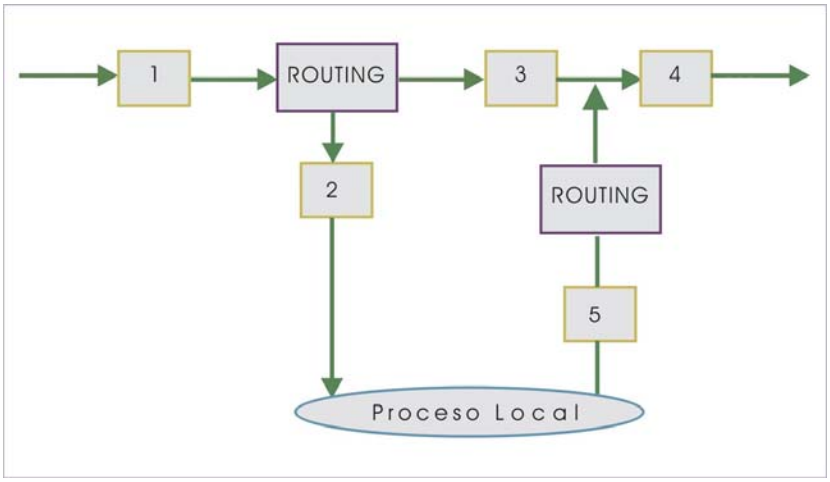


Figura 4.1. Recorrido de los paquetes en el Netfilter

En la descripción del recorrido, los paquetes pasan al gancho NF_IP_PREROUTING del sistema Netfilter, es en este punto donde pasan todos los paquetes que han sido validados, posteriormente en el código de ROUTING (enrutamiento), el cual decide si el paquete es destinado a otra interfaz o a un proceso local.

Si su destino es hacia otra interfaz se llama al sistema Netfilter para ejecutar el gancho que está etiquetado con el número 3, llamado también NF_IP_FORWARD, después todos los paquetes que salen del sistema pasan por un gancho final llamando NF_IP_POSTROUNTING.

En cambio si el paquete es destinado para un proceso local se llama al sistema NF_IP_LOCAL_IN, por otra parte para los paquetes creados de manera local se procesa

con NF_IN_LOCAL_OUT, el enrutamiento se hace después de la ejecución de este gancho [URL, 13].

Como complemento al Netfilter, se usa la herramienta IPtables la cual inserta y elimina reglas de la tabla de filtrado o de paquetes del Kernel, algunas de las principales ventajas que se tienen al utilizar el Netfilter junto con la herramienta IPtables son:

- Construir firewalls basados en filtrado de paquetes.
- Utilizar NAT y enmascaramiento con ip's privadas, para accesar a Internet en caso de no contar con suficientes direcciones ip reales o públicas.
- Usar NAT para la implementación de proxies.
- Ayuda a tc e IProute2 para la aplicación de QoS.
- Manipulación de paquetes.

Las cadenas que Netfilter tiene predefinidas son:

- INPUT, FORWARD que están relacionadas con el filtrado de paquetes.
- PREROUTING y POSTROTING para la NAT y la manipulación de paquetes.
- OUTPUT, está disponible para todas las actividades.

4.3 Descripción de IPtables

Es la herramienta que configura las reglas del cortafuegos IP del Kernel, además de ser considerada un sistema de selección de paquetes, que consiste en filtrar dichos

paquetes, traducir direcciones de red y manipular los paquetes antes de enrutarlos. IPtables es una estructura de tabla genérica para la definición de un conjunto de reglas, en cada una de estas reglas se especifican ciertas características que deben cumplir los paquetes y para cada regla se asocia una acción o target, dichas reglas son las encargadas de decidir que hacer con el paquete [URL, 2].

Algunas de las facilidades que se tienen al usar el IPtables son:

- Especificación de puertos de origen y destino.
- Soporte para protocolos TCP/UDP/ICMP.
- Soporte para interfaces de paquetes origen y destino.
- Permite un número ilimitado de reglas.
- Redireccionamiento de puertos.
- Muy rápido estable y seguro.
- Enmascaramiento.

Las targets o acciones mas usadas son:

- ACCEPT, deja pasar el paquete.
- DROP, descarta el paquete.
- QUEUE, pone un paquete en cola.
- RETURN, regresa el paquete hasta encontrar o cumplir con la regla comparada.

Cada target hace uso de 3 tablas que se describen a continuación [14]:

1. TABLA FILTER.- Es la tabla por defecto, se encarga de filtrar como lo dice su nombre únicamente los paquetes y contiene las cadenas: INPUT, OUTPUT y FORWARD, (de entrada, salida y reenvió). Estas cadenas son listas de reglas que sirven para marcar un conjunto de paquetes y cada una de ellas tiene una función específica, tal como se muestra a continuación.

- **INPUT,** decide el destino de los paquetes entrantes localmente en el host.

- **OUTPUT,** se usa para filtrar paquetes que son generados localmente en el host con destinos externos, permite modificar el destino de los paquetes.

- **FORWARD,** se usa para decidir que hacer con los paquetes que llegan a una interfaz y tienen como destino otra.

2. TABLA NAT.- Esta tabla se refiere a los paquetes enrutados en un sistema de enmascaramiento, trabaja de la siguiente manera; cuando un flujo de paquetes atraviesa la tabla el primer paquete es admitido y el resto se identifica automáticamente como parte del flujo. Esta tabla contiene las siguientes cadenas.

- **PREROUTING,** altera los paquetes recibidos por medio de una interfaz de red al momento de llegar.

- **OUTPUT,** se encarga de alterar los paquetes generados localmente antes de ser enviados por una interfaz.

- **POSTROUTING,** esta cadena altera los paquetes antes de que sean enviados a través de una interfaz.

3. TABLA MANGLE.- Esta tabla marca los paquetes entrantes generados localmente, y esta marca permite un tratamiento específico para dichos paquetes, tiene la función de modificar los paquetes y sus cabeceras como el TTL y el ToS, contiene las cadenas:

- **PREROUTING,** que se encarga de alterar los paquetes entrantes antes de ser enviados o enrutados.

- **OUTPUT,** altera los paquetes generados localmente antes de ser enviados por medio de una interfaz.

4.4 Firewalls

Un firewall o cortafuegos se define como un organizador cuyo propósito o función principal es proteger la red, [URL, 14], un firewall restringe ciertos tipos de tráfico desde Internet a la red local y viceversa, esto quiere decir, que restringe el flujo de información entre dos redes.

Para determinar qué tipo de tráfico se permitirá se utiliza el firewall usando un conjunto de reglas que son predefinidas por el administrador, tomando en cuenta sus necesidades y la de sus usuarios. El tráfico de red está constituido por paquetes IP, los cuales son datos que viajan en flujos desde un lugar origen hasta un lugar destino, y dichos paquetes tiene cabeceras que contienen información sobre el paquete fuente, el paquete destino, tipos de protocolos, etc., esto sirve para que el firewall con las reglas compruebe las cabeceras y determine así que paquetes se aceptan (ACCEPT), cuales se niegan (REJECT) o se rechazan (DROP).

El firewall aparte de proteger y controlar el acceso puede hacer otras cosas como por ejemplo:

- Impedir el acceso a determinados usuarios para ciertos servidores o servicios.
- Bloquear el acceso a sitios en específico en Internet.
- Sirve para vigilar todas las comunicaciones entre las redes, tanto internas como externas.

4.5 Tipos de Firewalls

Actualmente la política que se ha adoptado para implementar un firewall es la de negar todo y después ir abriendo los puertos de cada uno de los servicios necesarios, lo cual hace lo hace más seguro. Un firewall puede ser implementado por software o por medio de hardware.

Por hardware, es un dispositivo que se coloca entre el servidor y la conexión física al Internet [URL, 15], cabe señalar que estos dispositivos tiene a su vez reglas que se encargan de llevar a cabo la protección de la redes, lo que hace que sean equipos dedicados exclusivamente a realizar esta función específica y dedicada de firewall.

Por software, es un conjunto de reglas instaladas en un servidor que tiene acceso a Internet, la mayor desventaja que se puede tener al instalar un firewall de este tipo es que se llegan a consumir recursos de la computadora y en determinadas ocasiones pueden provocar un error de compatibilidad con otro software instalado, sin embargo la ventaja que se puede tener, es que resulta más económico y tanto su instalación como su actualización es más sencilla [15].

Para los firewalls por software existen varias técnicas o modelos para ser implementadas, las más comunes son:

- **Nivel de aplicación,** analiza el contenido del paquete para tomar su decisión de filtrado, este tipo de firewall tienen un alto grado de intrusión y permiten un control relacionado con el contenido del tráfico, algunos firewalls de este tipo combinan recursos básicos existentes. Esta técnica checa la información de todos los paquetes y mantiene el estado de conexión y secuencia de la información, examina el contenido del nivel 7 y ofrece servicios proxy.

Un proxy aplica mecanismos de seguridad a determinadas aplicaciones como p.e. FTP, HTTP, TELNET, SMTP, una de las ventajas que se tiene al usar un proxy es la capacidad de ocultar la red interna desde el exterior ya que todos los paquetes que atraviesan el proxy no aparecen con su dirección IP verdadera sino con la IP asignada al proxy [URL, 16].

- **Nivel de paquetes,** este tipo de firewall toma decisiones basados en los parámetros del paquete como puerto de origen/destino, estado de conexión. Los paquetes que pasan por él, pueden ser o no aceptados tomando en cuenta las reglas que se tienen predefinidas, es decir:

  a) El paquete se descarta en caso de no encontrar alguna regla que se le aplique o también al no coincidir con ninguna de las condiciones de las reglas.
  b) Si existe una regla y el paquete cumple con ella, se le permite el paso.
  c) Si hay una regla que se pueda aplicar al paquete y en ella se especifica la negación de estos paquetes, se rechazan.

- **Nivel de Circuito,** trabajan a nivel de la capa 4 (Transporte)[10], se basan en el control de la conexión TCP, por un lado reciben las peticiones de conexiones a un puerto TCP y por otro establecen la conexión con el destinatario deseado [URL, 17].

[10] Capa del Modelo de Referencia OSI que es una norma universal para protocolos de comunicación el cual está dividido en 7 capas.

Una de las ventajas de este tipo de técnica es que permite controlar las conexiones por puertos pero la desventaja es que no verifica el contenido de aplicación de la comunicación.

4.6 Enrutamiento

4.6.1 Definición y componentes

El enrutamiento es el proceso de selección de un camino por el cual se mandarán paquetes [11], también se le conoce con el nombre de ruteo o encaminamiento, en general se le conoce como el proceso de dirigir y transferir un paquete por medio de la red.

El enrutamiento consta de dos características que son:

Determinación de trayectorias y conmutación, la conmutación es cuando los paquetes se distribuyen por la red pero no existe ninguna trayectoria definida, y la determinación de trayectorias se refiere a lo que es la métrica, longitud de trayectoria, algoritmos de enrutamiento y tablas de enrutamiento, la métrica es un estándar de medición en cuanto a la longitud, para determinarse cual es la mejor ruta.

Algunas de las métricas más utilizadas son las siguientes:

- **Contabilidad,** se refiere a la dependencia descrita en tasas de errores de cada enlace.
- **Retardo,** se refiere al tiempo que tarde en moverse la información por el enlace.
- **Ancho de banda,** capacidad de transmisión de un enlace.
- **Carga,** es la cantidad de tráfico que hay en un enlace.
- **Costos de comunicación,** se refiere al valor obtenido con cualquier ponderación de las métricas usadas.
- **Longitud de trayectoria,** es la más común de las métricas, es la suma de los costos asociados con cada uno de los enlaces por los que se pasa.

4.6.2 Protocolos de enrutamiento

Los protocolos de enrutamiento son el lenguaje por el cual los demonios[11] de enrutamiento intercambian información acerca de la red [URL, 18], un protocolo de enrutamiento es el que se encarga de recopilar información de los enrutadores de la red y se dividen en dos tipos: IGP (*Interior Gateway Protocols*, Protocolos de Enrutamiento Interior) y EGP (*Exterior Gateway Protocols*, Protocolos de Enrutamiento Exterior).

Existen tres clases de protocolos de enrutamiento:

• **Vector distancia,** se encarga de determinar la dirección y distancia.

• **Estado del enlace,** conoce la topología completa de la red.

[11] Un demonio o daemon es un proceso que se ejecuta en segundo plano en lugar de ser controlado directamente por el administrador, se ejecuta de manera continúa.

• **Híbridos equilibrados,** es una combinación de los protocolos vector distancia y los protocolos de estado del enlace.

El objetivo de los IGP es llegar por el mejor camino a todos los destinos de los SA (*Autonomous System*, Sistemas Autónomos), son llamados así por ser zonas administradas por una autoridad común, mientras los EGP, hacen enrutamiento por políticas, con este protocolo se pueden impedir ciertos tráficos y también se encargan de la comunicación entre enrutadores de redes diferentes [URL, 18].

Los protocolos más conocidos se muestran en la siguiente tabla:

Tabla 4.2. Protocolos IGP y EGP

IGP	EGP
RIP: (*Routing Information Protocol*, Protocolo de Información de Enrutamiento).	Está el estándar **BGP-4** (*Border Gateway Protocol*, Protocolo de Enrutamiento Exterior Versión 4,)
IGRP: (*Interior Gateway Routing Protocol*, Protocolo de enrutamiento de gateway interior).	
EIGRP:(Protocolo de Enrutamiento interior de gateway mejorado).	
OSPF: (*Open Shortest Path First*, Abrir Primero la Trayectoria más corta).	

A continuación se da una descripción de cada uno de estos protocolos.

RIP, es un protocolo de enrutamiento por vector de distancia basado en el algoritmo Bellman-Ford, para redes LAN, que calcula las distancias hacia la computadora destino

en función de cuántos enrutadores debe pasar un paquete para llegar a su destino, utiliza UDP como protocolo de transporte, con el número de puerto 520 como puerto de destino, el RIP actualiza las tablas de enrutamiento cada 30 segundos, la desventaja de este tipo de protocolo es que necesita que constantemente se conecten los enrutadores vecinos lo que hace que genere una gran cantidad de tráfico.

IGRP, es un protocolo de enrutamiento por vector de distancia, el cual usa una métrica compuesta basada en variables diferentes de la red tales como el ancho de banda y contabilidad, solo por mencionar algunas, las actualizaciones de las tablas de enrutamiento se hacen cada 90 segundos.

EIGRP, es un protocolo basado en el IGRP es un protocolo mixto, por basarse en la métrica de vector distancia y en el estado de enlace, este un protocolo de enrutamiento sin clase a diferencia del IGRP, su tabla de enrutamiento contiene las mejores rutas hacia un destino.

OSFP, este protocolo se convirtió en estándar en 1990, es el sucesor del RIP, se basa en el algoritmo de Dijkstra, creando un árbol con el nodo que lo calcula como raíz y los caminos más cortos a cada enrutador dentro de la red, garantizando con esto, rutas sin ciclos (loops), pero por tal motivo se requiere mucha más memoria y procesamiento que en el RIP, una de las ventajas o beneficios al usar este protocolo es que se pueden calcular varias rutas para un mismo destino. Este tipo de protocolo soporta enrutamiento

manejando tipo de servicio y balanceo de carga, permite importar datos de los protocolos RIP y EGP.

BGP-4, usa el TCP como protocolo de transporte, con el número de puerto 179, este protocolo intercambia copias completas de sus tablas de enrutamiento, las cuales pueden ser bastante grandes.

Una de sus funciones más importantes del BGP (*Border Gateway Protoco*l, Protocolo de Enrutamiento Exterior) es la detección de ciclos en un SA, usando los atributos de rutas de un SA, y para intercambiar información de enrutamiento entre enrutadores externos.

Este tipo de protocolo tiene 3 características que son:

- Soporta un protocolo NAP (*Neighbor Acquisition Protocol*), se considera como vecinos a dos enrutadores si están conectados por una red que es transparente para ambos.

- Soporta un protocolo NR (*Neighbor Reachability*), el enrutador lo usa para mantener información en tiempo real sobre la accesibilidad de sus vecinos, para la cual usa dos mensajes el *Hello* y *I Hear You* que es la respuesta a *Hello.*

- Soporta mensajes de actualización (NR) que llevan información de enrutamiento [URL, 19].

Para seleccionar el tipo de protocolo que se debe usar, se necesita tomar en cuenta las características de la red donde serán implementados, así como también conocer los servicios de las aplicaciones, como por ejemplo:

- Topología de la red.
- Convergencia.
- Escalabilidad de la red.
- Selección de rutas.
- Seguridad.

4.6.2.1. Algoritmos y tablas de enrutamiento

Un algoritmo de enrutamiento es quien decide el mejor camino que deben tomar los paquetes para determinado destino y se hace tomando en cuenta la información recopilada por el protocolo.

Una definición mas formal de lo que es un algoritmo de enrutamiento la da CISCO, de la siguiente manera, “es un sistema de reglas que controlan un comportamiento de red de tal manera que la adaptan a las diferentes circunstancias dentro de una topología de red” [URL, 20].

Las tablas de enrutamiento se alimentan de información variada otorgada por el algoritmo de enrutamiento, en sí la tablas de enrutamiento contienen información que se utiliza por el software para seleccionar la mejor ruta y también poseen los datos acerca de la conveniencia de una trayectoria, los enrutadores comparan medidas para determinar las rutas óptimas, esto difiere en función del diseño del algoritmo de enrutamiento que se utilice.

Las tablas de enrutamiento tienen la siguiente apariencia, usando la utilería ip route [URL, 2].

```
# ip route show
212.64.94.1   dev   ppp0  proto kernel scope link src 212.64.94.251
10.0.0.0/8    dev   eth0  proto kernel scope link src 10.0.0.1
127.0.0.0/8   dev   lo    scope link
default via  212.64.94.1   dev ppp0
```

Lo que anteriormente se mostraba con el route como sigue:

```
# route –n

Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
212.64.94.1	0.0.0.0	255.255.255.255	UH	0	0	0	ppp0
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	212.64.94.1	0.0.0.0	UG	0	0	0	ppp0

Donde:

U ruta utilizable

G ruta indirecta (pasarela)

H la entrada es una dirección de máquina

Capítulo 5

5.1 Introducción

Un Administrador de red debe asegurarse que ésta funcione de manera confiable, rápida y que sus recursos sean utilizados de manera eficiente, por tal motivo es indispensable hacer un monitoreo de la misma. Otro de los beneficios de hacer un monitoreo es para determinar a través de éste, si existen virus o uso de aplicaciones p2p (*peer to peer*). En el presente trabajo de investigación se utilizaron las herramientas Ntop y Ethereal para realizar un análisis, el cual ayuda a determinar el estado de la red antes de implementar QoS, esto es con el propósito de saber de que manera se pueden gestionar adecuadamente los recursos de la red.

El Ntop permite monitorear una red en tiempo real, mostrando una lista de los dispositivos que se están usando así como información del IP. Su interfaz para llevar a cabo el monitoreo es administrable por vía Web, así también es importante mencionar que se encarga de monitorear protocolos, tales como, TCP/UDP/ICMP, ARP, NetBios, IPX, TCP/UDP FTP, HTTP, DNS, TELNET, SMTP/POP/IMAP, SNMP, X11 [URL, 21].

El Ethereal también es un analizador de protocolos, éste se encarga de leer paquetes y los decodifica para su comprensión, utiliza las librerías Libpcap y Winpcap que se usan para la captura de paquetes a nivel usuario y son requeridas para las plataformas Linux y Windows respectivamente [URL, 22]. En este caso se usa para identificar a los

usuarios y ver el tipo de tráfico que generan. El Ethereal se utiliza como apoyo para obtener los datos de las cabeceras IP, y del segmento TCP, donde aparece el tipo de protocolo, el origen y destino del paquete que se captura.

En este capítulo se mostrarán las reglas que conforman la QoS tomando en cuenta el diseño de la red y las necesidades de los usuarios, para poder determinar o identificar dichas necesidades se hará un análisis del estado de la red. En dicho análisis lo que se pretenderá es conocer que aplicaciones utilizan más ancho de banda. Una vez que se tenga implementada la QoS, se harán pruebas tanto en la descarga de archivos desde Internet como en la transferencia de archivos por FTP para poder determinar el desempeño de la red. El escenario de trabajo donde se aplicara QoS, es una red Ethernet con aproximadamente 100 usuarios y un enlace de 2048 kbps de bajada y 512 kbps de subida.

Al final de esta sección, se conocerán los beneficios de aplicar QoS en una red usando las herramientas de Linux y la herramienta PF de OpenBSD con el fin de conocer la efectividad de cada una de ellas.

5.2 Antecedentes del rendimiento de la red sin aplicar QoS

Para efectos del presente trabajo, es necesario recordar que se está trabajando bajo el sistema operativo Linux, con la distribución Slackware 10.2 y con la versión del Kernel 2.6.13.

Antes de aplicar QoS se hace un análisis del comportamiento de la red, para detectar cuales son los recursos que más se utilizan, así como el consumo de ancho de banda que hace cada usuario.

Para llevar a cabo dicho análisis se hace uso de la herramienta Ntop ver. 3.2 que como se mencionó anteriormente, ayuda a detectar el estado de la red a través de una página Web, el uso de esta herramienta hace posible lo siguiente [URL, 21]:

- Medición del tráfico.
- Caracterización y monitorización del tráfico.
- Detección de posibles violaciones de seguridad en la red.
- Optimización y planeación de la red.

El Ntop consta de un menú de opciones que contiene toda la información antes mencionada y las secciones que permiten verificar esto, son:

- Summary.
- All Protocols.
- IP.

Las opciones ofrecidas por el Ntop permiten obtener:

- Total de paquetes que están siendo procesados.

- Distribución de protocolos.
- Consumo de ancho de banda por cada usuario.
- Puertos más utilizados.
- Un resumen general de tráfico.

Resumiendo las opciones anteriores, se puede decir que el Ntop sirve para detectar el mal uso del ancho de banda, a partir de estos datos es posible implementar las políticas de QoS para hacer eficiente el uso de la red.

La Figura 5.1 explica los datos registrados durante el análisis, en ella se muestra el total de paquetes recibidos y procesados en el momento del monitoreo, clasificándose por su tamaño medido en bytes así mismo su porcentaje correspondiente.

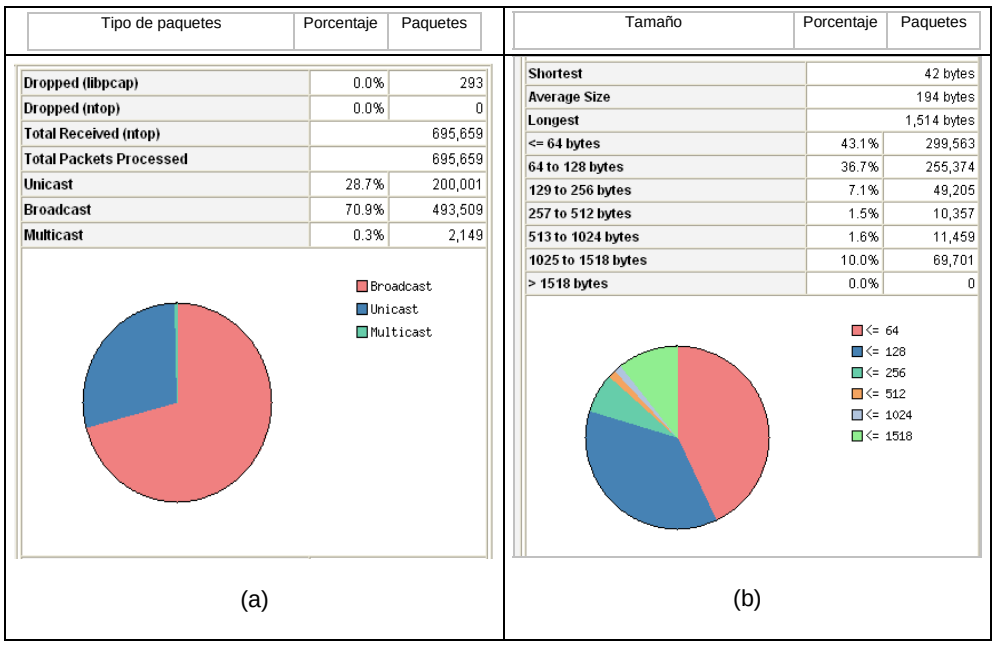
Gráficamente se notan los 3 tipos de tráfico existentes en la red, que son:

Unicast, con un 28.7%, esto debido a que es el trafico mas común en Internet, los datos se envían solo donde haya usuarios interesados en recibirlos es decir, el destinatario de la información es un único host con una IP concreta, un ejemplo de este tipo de tráfico es cuando se hace uso de las aplicaciones tales como FTP, TELNET, WWW, la desventaja de este tipo de tráfico es que envía una copia para cada receptor.

Broadcast, con un 70.9%, aquí los datos se envían en todos los segmentos de la red lo que genera demasiado tráfico esto debido a que el broadcast es un componente

natural de las redes TCP/IP y particularmente las redes Ethernet, la cual es el tipo de red que se tiene implementada en este caso de estudio.

Multicast, con un 0.3%, el tráfico aquí no es mucho debido a que los datos se envían una vez y solo van a dirigidos a los usuarios interesados.



Tipo de paquetes	Porcentaje	Paquetes
Dropped (libpcap)	0.0%	293
Dropped (ntop)	0.0%	0
Total Received (ntop)		695,659
Total Packets Processed		695,659
Unicast	28.7%	200,001
Broadcast	70.9%	493,509
Multicast	0.3%	2,149

(a)

Tamaño	Porcentaje	Paquetes
Shortest		42 bytes
Average Size		194 bytes
Longest		1,514 bytes
<= 64 bytes	43.1%	299,563
64 to 128 bytes	36.7%	255,374
129 to 256 bytes	7.1%	49,205
257 to 512 bytes	1.5%	10,357
513 to 1024 bytes	1.6%	11,459
1025 to 1518 bytes	10.0%	69,701
> 1518 bytes	0.0%	0

(b)

Figura 5.1. Clasificación de paquetes. (a) De acuerdo al tipo de tráfico, (b) De acuerdo a su tamaño

En la Figura 5.2 se indica en porcentaje el tipo de protocolo mas usado, a fin de obtener una estadística y saber cuál es la cantidad del uso de cada uno de ellos en la interfaz monitoreada.

Global Protocol Distribution

Protocol	Data	Percentage
IP	136.0 MB	85.8%
(R)ARP	11.3 MB	8.3%
IPX	6.2 MB	3.9%
NetBios	733.5 KB	0%
IPv6	72.1 KB	0%
STP	756.0 KB	0%
Other	9.7 KB	0%

TCP	104.7 MB	77.0%
UDP	31.2 MB	23.0%
ICMP	65.3 KB	0%
ICMPv6	7.7 KB	0%
IGMP	42.7 KB	0%
Other IP	10.5 KB	0%

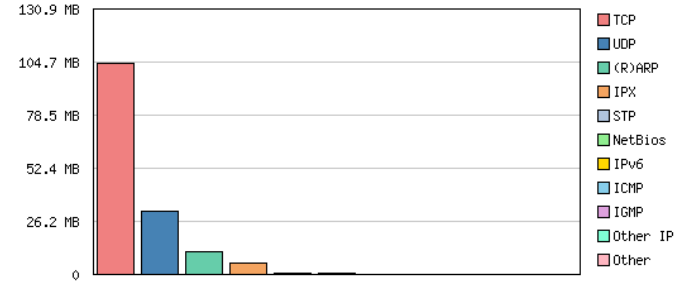


Figura 5.2. Distribución global de protocolos

En esta distribución global el tráfico se clasifica de acuerdo al protocolo de red (IP, IPX, NetBios, etc.). Se observa que los protocolos de mayor consumo son el TCP que ocupa el 77.0% y el UDP que ocupa un 23.0%. En la siguiente figura se observa de manera detalla el por que el TCP tiene un mayor porcentaje comparado con los demás protocolos.

Global TCP/UDP Protocol Distribution

TCP/UDP Protocol	Data	Flows	Accumulated Percentage / Historical Protocol View
HTTP	86.3 MB	8,017	63.5%
DNS	4.3 MB	11,921	3.2%
NBios-IP	18.7 MB	170,854	13.7%
Mail	18.3 MB	245	13.4%
DHCP-BOOTP	4.6 MB	10,158	3.4%
NFS/AFS	2.7 KB	1	0%
VoIP	0.1 KB	0	0%
SSH	168.0 KB	58	0%
Gnutella	2.8 KB	37	0%
Messenger	0.1 KB	0	0%
Other TCP/UDP-based Protocols	3.7 MB	13,497	2.8%

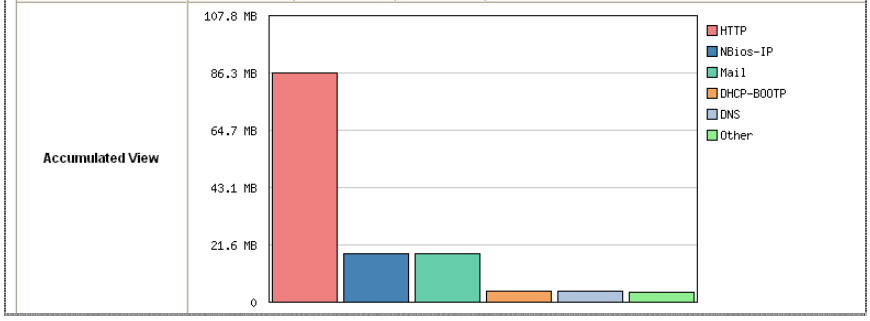


Figura 5.3. Distribución global de los protocolos TCP/UDP

La clasificación del tráfico se hace de acuerdo al protocolo IP (HTTP, FTP, etc.) como puede verse en la Figura 5.3. Esa gráfica ayuda a detectar el consumo del ancho de banda, siguiendo los valores del tráfico para ciertos protocolos, como en el caso del TCP/80 correspondiente al HTTP, se tiene mayor consumo de ancho de banda, esto se debe en mayor parte por la demanda que tienen los usuarios en la aplicaciones por la consulta de páginas Web, messenger, correo.

En la Figura 5.4 se indica el tráfico total por puerto, el cual es la suma del tráfico enviado y recibido.

Detallando la figura se tiene que el puerto TCP/137 corresponde al NetBios, el cual está generado un total de 46.6 KB, en el momento de ejecutar el monitoreo, esta prueba

se realizó en un horario de trabajo de 09:30 a 19:00 hrs. Aproximadamente durante un año y los resultados siempre fueron similares a los que se muestran en la figura siguiente.

TCP/UDP Traffic Port Distribution:
Last Minute View

TCP/UDP Port		Total	Sent	Rcvd
netbios-ns	137	46.6 KB	23.3 KB	23.3 KB
netbios-dgm	138	18.2 KB	9.1 KB	9.1 KB
6646	6646	13.5 KB	6.8 KB	6.8 KB
bootpc	68	7.9 KB	7.9 KB	0
bootps	67	7.9 KB	0	7.9 KB
domain	53	3.7 KB	2.9 KB	787
1027	1027	1.5 KB	267	1.3 KB
3128	3128	793	793	0
1590	1590	793	0	793
1029	1029	750	228	522
1890	1890	548	76	472
blackjack	1025	484	76	408
1037	1037	448	140	308
1211	1211	351	0	351
1063	1063	351	351	0

Figura 5.4. Distribución por puerto del tráfico TCP/UDP

En la Figura 5.5 se nota la distribución de tráfico de acuerdo a los protocolos de origen y destino, el cual se divide en:

- Tráfico Local (TCP vs. UDP).
- Tráfico remoto a local (TCP vs. UDP).
- Tráfico remoto (TCP vs. UDP).
- Tráfico local a remoto (TCP vs. UDP).

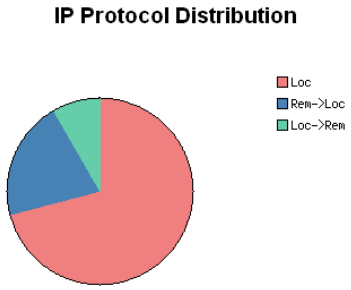


Figura 5.5. Distribución del protocolo IP

La distribución de la Figura 5.5 se obtuvo de la siguiente tabla, donde se detalla el tráfico con respecto al protocolo y origen (local vs. remoto), en donde se puede ver que el mayor tráfico es generado de manera local y que se identifica en la tabla con un 78.9 MB en datos, el tráfico remoto-local con un 23.3 MB y el tráfico local-remoto con un 8.8 MB.

Tabla 5.1. Distribución del tráfico de acuerdo al protocolo.

	Tráfico local		**Tráfico remoto-local**		**Tráfico remoto**		**Tráfico local- remoto**	
Protocol IP	Datos	Porcentaje	Datos	Porcentaje	Datos	Porcentaje	Datos	Porcentaje
TCP vs UDP	78.9 MB	TCP 98.5 % UDP 1.5 %	23.3 MB	TCP 0 % UDP 100 %	998.2 KB	TCP 85.2 % UDP 14.8 %	8.8 MB	TCP 0 % UDP 100 %
Protocol TCP/UDP								
FTP	18.8 KB	0%	0.1 KB	0 %			1.2 KB	0 %
http	64.5 KB	81.7 %	0.2 KB	0%	838.0 KB	83.9 %	0.1 KB	0 %
DNS	940.1 KB	1.2 %	535.7 KB	2.2 %	19.1 KB	1.9 %	8.8 MB	99.9 %
NBios-IP	114.5 KB	0 %	16.6 MB	71.4 %	1.2 KB	0%		
Mail	13.2 MB	16.8 %	0.2 KB	0 %	2.7 KB	0 %		
DHCP-BOOTP			3.8 KB	16.1 %				
NFS/AFS					1.3 KB	0 %		
VoIP			0.1 KB	0 %	0.1 KB	0 %	2.4 KB	0 %
SSH			4.3 KB	0 %	0.2 KB	0 %	6.8 KB	0 %
Kazaa					1.7 KB	0 %		
Messenger					5.3 KB	0 %		
Other TCP/UDP-based Protocols	144.6 KB	0%	2.4 MB	10.1 %	128.8 KB	12.9 %	2.7 KB	0 %

Con estos resultados se puede determinar cuales son las aplicaciones de mayor demanda, los puertos más utilizados y que usuarios consumen más ancho de banda, para determinar cuales son los servicios y aplicaciones que requieren de políticas de QoS, en este caso se puede ver en los datos de la tabla 5.1 que en los 4 tipos de tráfico, el protocolo de mayor consumo de ancho de banda es el HTTP, por lo que se le pondrá mayor atención a la hora de crear las reglas que conformarán la QoS para otorgarle una prioridad apropiada con un ancho de banda adecuado que asegure un buen desempeño en el servicio.

5.3 Recompilación del Kernel

Antes de iniciar con la creación de políticas se deben cumplir ciertos requisitos que son:

1. Verificar que el Kernel sea superior a la versión 2.6.x en caso que sea de la serie 2.4.x, se recomienda descargar la versión actual del Kernel, el cual contiene todas las opciones necesarias para activar los módulos que sirven para utilizar las herramientas Netfilter, IProute2, y las distintas clases y colas, una vez que se ha descargado la versión fuente se descomprime y recompila, activando las opciones (ver Anexo) que permiten la implementación de QoS.

2. Compilar e instalar las versiones más actuales tanto del lProute2 como del IPtables, al momento de realizar este trabajo, las versiones que estaban disponibles para cada herramienta son 2-2.6.20 y 1.3.8 respectivamente. En el IProute2 se utilizará el comando tc que sirve para manipular el control de tráfico, en combinación con las opciones [URL, 23]:

- **tc qdisc,** permite establecer la disciplina de colas.
- **tc class,** permite establecer clases basadas en la planificación de las disciplinas de colas.
- **tc estimator,** permite hacer una estimación del flujo de red.
- **tc filter,** permite establecer el filtrado de paquetes QoS/CoS (*Class of service*, Clase de servicio).
- **tc policy,** permite a los usuarios establecer las políticas QoS/CoS.

Por otra parte, IPtables se usa para establecer, mantener e inspeccionar las tablas de las políticas de filtrado de paquetes IP en el Kernel Linux, lo cual se hace a través del marcado de paquetes ya sea para aceptarlos, redirigirlos o rechazarlos. Aunando a esta característica importante, esta herramienta también es fundamental para implementar cortafuegos, lo cual da mayor robustez y seguridad a una red.

5.4 Creación y aplicación de reglas

Una vez que se cumplen los requisitos anteriores, se procede a realizar las reglas que conformaran la QoS, tomándose en cuenta las políticas que el administrador de red

haya planteado para la asignación de la prioridad en las diferentes aplicaciones que se utilizan en la red, haciéndolo en base al resultado del monitoreo y conociendo de antemano las aplicaciones más utilizadas así como las que más ancho de banda consumen, por lo tanto, con base en lo anteriormente expuesto, es conveniente para este caso de estudio considerar las siguientes políticas:

- Dar mayor prioridad al tráfico SSH, SYN, DNS, IPv4, IPv6 que al resto del tráfico, se toma para este tipo de tráfico una alta prioridad debido a que primero se tienen que enviar paquetes para comprobar si se puede o no establecer la comunicación, una vez que se comprueba se envía la información.

- El tráfico HTTP, HTTPS y FTP tendrá una prioridad mas baja que el SMTP, POP3, IMAP, los cuáles tendrán una prioridad media, por ser el que tiene mayor demanda, ya que la mayoría de los usuarios utilizan el correo como parte fundamental para mantener una comunicación constante y fluida. Esto se comprobó en el monitoreo según las gráficas presentadas anteriormente.

- El tráfico generado por aplicaciones chat, de intercambio de archivos, por ejemplo: como MSN, EMULE y cualquier otro tipo de tráfico no contemplado anteriormente, tendrá la prioridad más baja, por estar considerados dentro del tipo de servicios de ocio.

Cabe mencionar que la prioridad es independiente a la asignación de ancho de banda para cada clase, la cual se hace desde unos cuantos Kbytes hasta la capacidad máxima, del ancho de banda, menos el 20% del total, por perdidas tanto en la Tx/Rx de la señal.

Antes de aplicar las reglas de QoS, se comprueba que no exista ninguna regla ejecutándose en ese momento, es decir el servidor debe estar limpio de cualquier regla para partir de cero, si existe alguna regla de IPtables no se puede aplicar alguna otra, ya que debe haber un orden en la ejecución de reglas, lo anterior se verifica con la siguiente instrucción:

```
# tc qdisc show dev eth1
```

Donde:

tc qdisc, indica la disciplina de colas que existen.

show dev, es la instrucción para mostrar las colas que hay en el dispositivo.

eth1, es la interfaz o tarjeta de red que se tiene configurada.

En caso que el servidor no contenga alguna regla, mostrará una salida similar a lo siguiente:

```
qdisc pfifo_fast 0 bands 3 priomap 1222120011111111
```

Si por el contrario apareciera una salida diferente a lo anterior, significa que existen reglas del IPtables por lo tanto se procede a eliminarlas utilizando el siguiente comando:

```
# tc qdisc del dev eth1 root
```

En donde:

del dev, es la instrucción con la que se indica que se va a borrar la cola raíz en la tarjeta de red especificada por **eth1 root.**

Para implementar las reglas de QoS se realizan los siguientes puntos:

1.- Hacer un script el cual va a estar diseñado de acuerdo a las políticas mencionadas anteriormente. Para este caso, se divide todas aquellas aplicaciones a las que se le dará un trato o prioridad especial.

2.- Se generan 4 clases, esta distribución de clases se crean de acuerdo al criterio de cada administrador, en este caso los usuarios necesitan hacer uso del Internet para la búsqueda de información acerca de sus investigaciones y clases, hacen uso de la transferencia de archivos vía FTP y envío de correo ya que constantemente se comparten información con instituciones externas, así mismo en los departamentos administrativos y financieros se hace uso de transacciones vía Internet. Teniendo identificadas las clases anteriores, se especifican la aplicación, la asignación del ancho de banda y su prioridad, quedando como se muestra en la tabla 5.2.

Tabla 5.2. Distribución de ancho de banda

Clase	Aplicación	Ancho de banda asignado	Prioridad
10	SSH, SYN, DNS, IPv4, IPv6	350kbits	1
11	SMTP, POP3, IMAP	400kbits	2
12	HTTP, HTTPS, FTP	650kbits	3
13	EMULE, MSN, tráfico en general	200kbits	4

Las clases se forman de acuerdo a la prioridad que se va aplicar a cada una de ellas, cabe mencionar que mientras mayor sea la prioridad otorgada al servicio es menor el número que se asigna es decir, la prioridad 1 quiere decir que es la de mayor importancia. El total de ancho de banda que se tiene para repartir entre las diferentes aplicaciones es de 1800kbps.

Cada tarjeta de red tiene una qdisc raíz asociada a ella, y cuando se reciben los paquetes en la tarjeta, la qdisc recibe la petición de desencolar, tomando en cuenta las marcas en el paquete que se haya establecido con el Netfilter, es así como la qdisc lo enviará a alguna de las clases que contiene. Cada qdisc y cada clase tienen asignado un controlador el cual consta de 2 partes separadas por 2 puntos, un número mayor y un número menor, las clases deben tener el mismo número mayor que sus padres y el número menor debe ser único dentro de una qdisc y sus clases [16]. Para hacer más entendible lo anterior se genera un diagrama del diseño de las colas, como se ve en la Figura 5.6.

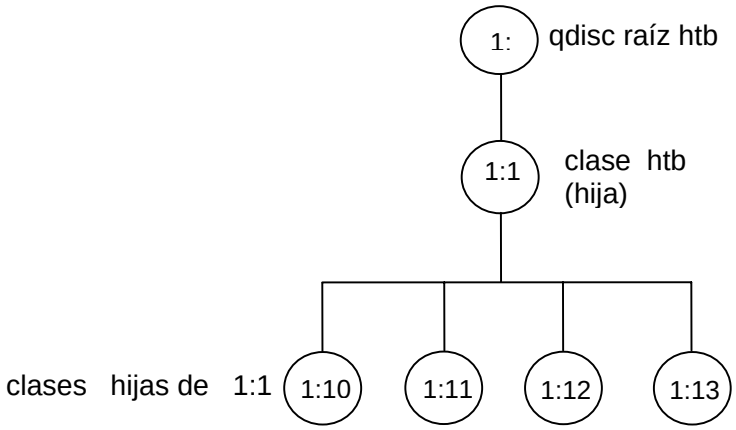


Figura 5.6. Árbol de encolado o de clases

La clase raíz 1: se define de la siguiente manera:

```
$TC qdisc add dev  $IF_EXT root handle 1: htb default 13
```

Donde:

$IF_EXT = es la tarjeta de red, eth1

TC permite añadir (**add**) a la tarjeta de red *(***dev $IF_EXT***)* una **qdisc** raíz (**root**) con disciplina **htb,** que llamaremos **1:** haciendo referencia a la clase **13** que es donde se irá el tráfico por **default**.

Como se menciona en el capítulo 3, dentro de la disciplina de colas con clase, se tienen a CBQ, PRIO, HTB, las cuales son útiles para cuando se tienen diferentes tipos de tráfico y se les quiere dar un tratamiento por separado, en este caso se usa la disciplina

HTB porque permite crear subcolas para reservar ancho de banda, esto se utiliza al momento de crear la clase 1:1, con el fin de organizar la llegada de los paquetes a la cola qdisc, como se ve en la siguiente instrucción:

```
$TC class add dev  $IF_EXT parent 1:  classid 1:1 htb  rate ${MAX}kbit ceil ${MAX}kbit
```

Donde **TC class add** se encarga de crear una clase dentro de la qdisc, donde la cola 1 es su padre, esto se indica con **parent 1:** y su nombre se define con **classid** y es 1:1 (significa que es la hija 1 de la raíz 1).

En esta clase ningún paquete pasa a más velocidad de lo que se ha definido como máximo, para este caso se maneja un **MAX** de 1800kbit, y donde **ceil** se refiere al máximo posible de transferencia y **rate** el mínimo garantizado.

Posteriormente se crean las clases denominadas 1:10, 1:11, 1:12, 1:13, indicando tanto el máximo de transferencia (**ceil**) como su mínimo (**rate**) y su prioridad, quedando de la siguiente manera:

```
$TC class add dev  $IF_EXT parent 1:1 classid 1:10 htb  rate 350kbit  ceil 600kbit          prio 1
$TC class add dev  $IF_EXT parent 1:1 classid 1:11 htb  rate 400kbit  ceil 700kbit          prio 2
$TC class add dev  $IF_EXT parent 1:1 classid 1:12 htb  rate 650kbit  ceil ${MAX}kbit      prio 3
$TC class add dev  $IF_EXT parent 1:1 classid 1:13 htb  rate 200kbit  ceil 400kbit          prio 4
```

Dónde:

Las clases ahora (**classid**) son hijas de 1:1 (**parent 1:1**), **htb** es la disciplina que se va a utilizar en cada una de las clases y con **prio** se indica la prioridad para cada una de las clases, esto significa que si alguna de las clases no usa todo su ancho de banda asignado y garantizado, este puede ser repartido entre las otras restantes de acuerdo a sus prioridades.

3.- Ahora se asigna una cola (**qdisc**) a cada clase, usando la disciplina **sfq**, para clasificar los paquetes por conexión, y que todos sean tratados de manera justa, se puede ver que cada qdisc 100:, 110:, 120: y 130: están asociadas con las clases 1:10, 1:11, 1:12 y 1:13.

```
$TC qdisc add dev  $IF_EXT parent 1:10  handle 100: sfq
$TC qdisc add dev  $IF_EXT parent 1:11  handle 110: sfq
$TC qdisc add dev  $IF_EXT parent 1:12  handle 120: sfq
$TC qdisc add dev  $IF_EXT parent 1:13  handle 130: sfq
```

Posteriormente, se añade a la interfaz de red (**$IF_EXT***)* un filtro (**filter**) para tráfico de tipo IP, el cual se asocia a la rama 1: y el cual se encargará de manejar los paquetes (**handle 1, handle 2, handle 3, handle 4**) y dirigirlos (**fw**) a la cola adecuada gestionada por la clase 1:1x (**classid 1:10, classid1:11, classid 1:12, classid 1:13**). Cabe mencionar que el **fw** clasifica en base a la marca del paquete que esté hecha con IPtables.

```
$TC filter add dev  $IF_EXT protocol ip parent 1:  handle  1  fw classid  1:10
$TC filter add dev  $IF_EXT protocol ip parent 1:  handle  2  fw classid  1:11
$TC filter add dev  $IF_EXT protocol ip parent 1:  handle  3  fw classid  1:12
$TC filter add dev  $IF_EXT protocol ip parent 1:  handle  4  fw classid  1:13
```

4.- Una vez hechas las clases y definiendo sus prioridades se procede al marcado de cada uno de los paquetes, para cada tipo de tráfico definido anteriormente, es aquí donde se hace uso del Netfilter al invocar el IPtables en cada una de las políticas.

Al terminar de crear las reglas se inicia el script con la siguiente instrucción, desde una terminal de Linux.

```
root@dante:/home/mireya/qos# ./qos2n
```

Donde root@dante:/home/mireya/qos muestra la ruta donde se encuentra el script dentro del servidor, llamado también prompt y qos2n es el nombre que se le da a dicho script.

Y con la siguiente instrucción: `tc –s class show dev eth1`

Se muestra (**show dev**) que realmente el tráfico esté en las clases que se definieron anteriormente, con la opción **-s** se pueden ver las estadísticas de una interfaz de red (**eth1**). En la Figura 5.7, se observa que las clases creadas contienen información sobre cuantos bytes y paquetes se están enviando en ese momento, en resumen el tráfico

que pasa en cada una de las clases es debido al trato que se le da en las prioridades de acuerdo a la marca que se le asigna a cada paquete.

```
root@dante:/home/mireya/qos# tc -s class show dev eth1
class htb 1:11 parent 1:1 leaf 110: prio 2 rate 400000bit ceil 700000bit burst 1799b cburst 1949b
 Sent 895 bytes 15 pkt (dropped 0, overlimits 0 requeues 0)
 rate 152bit 0pps backlog 0b 0p requeues 0
 lended: 15 borrowed: 0 giants: 0
 tokens: 33752 ctokens: 21042

class htb 1:1 root rate 1800Kbit ceil 1800Kbit burst 2499b cburst 2499b
 Sent 15632 bytes 169 pkt (dropped 0, overlimits 0 requeues 0)
 rate 352bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 11196 ctokens: 11196

class htb 1:10 parent 1:1 leaf 100: prio 1 rate 350000bit ceil 600000bit burst 1774b cburst 1899b
 Sent 2707 bytes 33 pkt (dropped 0, overlimits 0 requeues 0)
 rate 88bit 0pps backlog 0b 0p requeues 0
 lended: 33 borrowed: 0 giants: 0
 tokens: 39485 ctokens: 24739

class htb 1:13 parent 1:1 leaf 130: prio 4 rate 200000bit ceil 400000bit burst 1699b cburst 1799b
 Sent 12030 bytes 121 pkt (dropped 0, overlimits 0 requeues 0)
 rate 96bit 0pps backlog 0b 0p requeues 0
 lended: 121 borrowed: 0 giants: 0
 tokens: 67994 ctokens: 36045

class htb 1:12 parent 1:1 leaf 120: prio 3 rate 650000bit ceil 1800Kbit burst 1924b cburst 2499b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 24260 ctokens: 11377

root@dante:/home/mireya/qos#
```

Figura 5.7. Asignación del tráfico en las clases creadas

5.5 Análisis del comportamiento de la red aplicando QoS

En este punto se vuelve a hacer uso de la herramienta Ntop para monitorear el tráfico, esta vez teniendo implementadas las reglas de QoS. Comparando las Figuras 5.1 y 5.8 se nota una mejoría en el comportamiento de la red, debido a que los paquetes ya son tratados de acuerdo a las prioridades establecidas anteriormente.

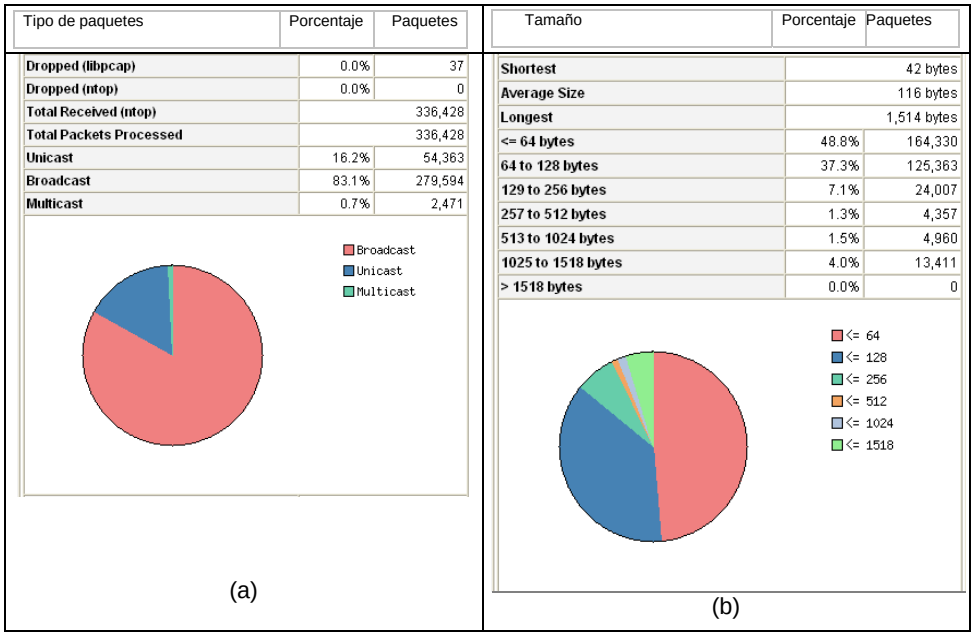


Figura 5.8. Clasificación de paquetes con QoS. (a) De acuerdo a su volumen, (b) De acuerdo a su tamaño.

En los resultados que muestra la Figura 5.8 se puede notar que se cuenta con una clasificación del tamaño de los paquetes en el cual hay un mejor control; así como también, en la tabla 5.3 se ve esta mejora en cuanto a los diferentes tipos de tráfico.

Tabla 5.3. Tipos de tráfico

Tipos de tráfico	Sin QoS	Con QoS
Unicast	28.7 %	16.2%
Broadcast	70.9%	83.1%
Multicast	0.3%	0.7%

En la distribución global de protocolos se puede ver también el mejoramiento de la red, la tabla 5.4 muestra el consumo de ancho de banda de los diferentes protocolos.

Tabla 5.4. Distribución global de protocolos

Protocolos		**Sin QoS**		**Con QoS**	
		Tamaño	**%**	**Tamaño**	**%**
IP	TCP	104.7 MB	77.0	21.1 MB	58.5
	UDP	31.2 MB	23.0	15.1 MB	41.7
	ICMP	65.3 KB	0	31.4 KB	0
	ICMPv6	7.7 KB	0	6.9 KB	0
	IGMP	42.7 KB	0	32.5 KB	0
	Other IP	10.5 KB	0	14.8 KB	0
(R)ARP		11.3 MB	8.3	6.5 MB	18.0
IPX		6.2 MB	3.9	3.5 MB	7.1
NetBios		733.5 KB	0	393.7 KB	0
IPv6		72.1 KB	0	148.6 KB	0
SMTP		756.0 KB	0	420.6 KB	0
Other		9.7 KB	0	18.0 KB	0

Como se puede apreciar en la tabla 5.4, la asignación de ancho de banda se hace de acuerdo a las necesidades de los usuarios al priorizar el tráfico, el cuál disminuye de manera considerable, incrementando el mejoramiento del comportamiento de la red lo cual se nota al abrir una ventana de cualquier navegador, por la pronta respuesta que se tiene al realizar la transferencia de archivos, utilizando el FTP y verificando la tasa de transferencia y el tiempo en realizar las descargas.

En la tabla 5.1 se observa como la distribución del tráfico se hacía de acuerdo al protocolo mas usado, en donde se aprecian los servicios que sobresalen más por su uso o demanda como el HTTP, FTP, DNS, MAIL, SSH. Una vez que se ha implementado QoS

se puede ver también el mejoramiento de estos servicios, pues se reduce de manera considerable el consumo de ancho de banda de los servicios, y este se puede ver resumido en la siguiente tabla:

Tabla 5.5. Comparación de la distribución de tráfico de acuerdo al tipo de protocolo

(a) Sin QoS	**Tráfico local**		**Tráfico remoto-local**		**Tráfico remoto**		**Tráfico local- remoto**	
Protocol IP	Datos	Porcentaje	Datos	Porcentaje	Datos	Porcentaje	Datos	Porcentaje
TCP vs UDP	78.9 MB	TCP 98.5 % UDP 1.5 %	23.3 MB	TCP 0 % UDP 100 %	998.2 KB	TCP 85.2 % UDP 14.8 %	8.8 MB	TCP 0 % UDP 100 %

(b) Con QoS	**Tráfico local**		**Tráfico remoto-local**		**Tráfico remoto**		**Tráfico local- remoto**	
Protocol IP	Datos	Porcentaje	Datos	Porcentaje	Datos	Porcentaje	Datos	Porcentaje
TCP vs UDP	21.8 MB	TCP 96.6 % UDP 3.4 %	14.8 MB	TCP 0 % UDP 100 %	474.7 KB	TCP 78.9 % UDP 21.1 %	52.8 KB	TCP 12.9 % UDP 87.1 %

Con los resultados de la tabla anterior se tiene la certeza de hacer un buen uso y control sobre los servicios ofrecidos a los usuarios en la red, dando a cada servicio el ancho de banda requerido y limitando aquellas tareas innecesarias que entorpecen y saturan la red.

En la Figura 5.9, extraída de la consola del servidor Linux, se nota el paso del tráfico a las distintas colas definidas en las reglas, de acuerdo a la priorización.

```
root@dante:/home/mireya/qos# tc -s class show dev eth1
class htb 1:11 parent 1:1 leaf 110: prio 2 rate 400000bit ceil 700000bit burst 1799b cburst 1949b
 Sent 10545715 bytes 9272 pkt (dropped 0, overlimits 0 requeues 0)
 rate 240bit 0pps backlog 0b 0p requeues 0
 lended: 6311 borrowed: 2961 giants: 0
 tokens: 33752 ctokens: 21042

class htb 1:1 root rate 1800Kbit ceil 1800Kbit burst 2499b cburst 2499b
 Sent 26906656 bytes 53985 pkt (dropped 0, overlimits 0 requeues 0)
 rate 4200bit 3pps backlog 0b 0p requeues 0
 lended: 7733 borrowed: 0 giants: 0
 tokens: 6902 ctokens: 6902

class htb 1:10 parent 1:1 leaf 100: prio 1 rate 350000bit ceil 600000bit burst 1774b cburst 1899b
 Sent 172795 bytes 2114 pkt (dropped 0, overlimits 0 requeues 0)
 rate 48bit 0pps backlog 0b 0p requeues 0
 lended: 2114 borrowed: 0 giants: 0
 tokens: 39860 ctokens: 24957

class htb 1:13 parent 1:1 leaf 130: prio 4 rate 200000bit ceil 400000bit burst 1699b cburst 1799b
 Sent 15065629 bytes 36143 pkt (dropped 0, overlimits 0 requeues 0)
 rate 4280bit 2pps backlog 0b 0p requeues 0
 lended: 32405 borrowed: 3738 giants: 0
 tokens: 27855 ctokens: 16714

class htb 1:12 parent 1:1 leaf 120: prio 3 rate 650000bit ceil 1800Kbit burst 1924b cburst 2499b
 Sent 1122517 bytes 6456 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 5422 borrowed: 1034 giants: 0
 tokens: 23656 ctokens: 11159
```

Figura 5.9. Resultado de la asignación de tráfico en las colas

5.6 Configuración de PF en el OpenBSD

En esta sección se hace la configuración de la herramienta PF del sistema operativo OpenBSD, versión 4.1, para aplicar QoS, los resultados obtenidos de esta manera serán comparados con los resultados de aplicar QoS con las herramientas de Linux IProute2 y Netfilter sobre la misma red, para tales efectos se da una breve explicación acerca de cuáles y para qué sirven tanto OpenBSD como PF.

OpenBSD es un sistema operativo libre muy parecido al Unix, multiplataforma, basado en 4.4 BSD, es gratis y funcional, se concentra en la portabilidad, cumplimiento de normas y regulaciones, corrección, seguridad y criptografía integrada, OpenBSD incluye emulación de binarios para la mayoría de los programas de los sistemas Solaris, FreeBSD, Linux, BSD/OS, SunOs y Hp-UX [17].

PF es una herramienta para el filtrado de paquetes que contiene el OpenBSD, con la cual se puede dar paso selectivo o bloquear los paquetes de datos que pasan a través de una red [URL, 24]. PF es la forma de aplicar QoS en OpenBSD, siendo una herramienta equivalente a las herramientas IProute2 y Netfilter de Linux.

La configuración del PF se hace en el archivo **pf.conf** en el cual se van a introducir las políticas que cumplen con las necesidades de la red que se está estudiando. En PF se utiliza el llamado regulador de caudal o **AltQ** (*Alternate Queueing*, Encolamiento Alterno), el cual es un conjunto de herramientas de QoS, que permiten establecer colas de tráfico, asignando caudales y prioridades [URL, 25].

En PF se hace la activación de colas (**queue**) en la interfaz de red (**on xl0**) que controla el tráfico hacia la red LAN, se hace uso del scheduler (planificador) **priq** para las prioridades, y se configura aquí mismo el ancho de banda (**bandwidth**) [URL, 26]. Lo anterior se realiza con la siguiente instrucción:

```
altq on xl0 priq bandwidth  1800 kb queue {std_out, ssh_im_out, dns_out, tcp_ack_out }
```

De manera general, lo que hace la regla anterior es activar las colas en la interfaz interna (**xl0**) para controlar el tráfico que sale hacia Internet.

Donde:

std_out, es la cola estándar, donde se irá todo el tráfico que no se especifique hacia que cola debe ir.

ssh_im_out, tráfico de SSH

dns_out, DNS

tcp_ack_out, paquetes TCP ACK

Después se asigna la prioridad (**prioq**) a cada cola (**queue** std_out):

```
queue std_out          prioq(default)
queue ssh_im_out    priority 3
queue dns_out          priority 4
queue tcp_ack_out   priority 5
```

Una vez realizado lo anterior, se hace la activación de colas (**queue**) en la interfaz interna (**xl1**), con el fin de controlar el tráfico que viene de Internet.

```
altq on xl1 cbq bandwidth  1.5 Mb queue {std_in, ssh_im_in, dns_in, red_in}
```

La regla anterior permite controlar el ancho de banda con el **cbq.** Posteriormente se hace el filtrado de las políticas para la interfaz xl1 entrante. Por razones de seguridad se recomienda negar por defecto todo y luego permitir selectivamente determinados tráficos, haciéndolo con la siguiente regla:

```
block in on xl0 all
```

Reglas de filtrado para la interfaz xl1 saliente:

```
block out on xl0 all
pass out on xl0  ext proto tcp from (xl1) to any flags S/SA keep state
queue(std_out,tcp_ack_out)
```

Con la regla anterior se deja pasar (**pass**) el tráfico de salida (**out**) por la interfaz **xl0** donde cada regla coincidirá con paquetes TCP (**proto tcp**) que vienen de la interfaz **xl1** con cualquiera de las banderas SYN o ACK (**S/SA**) y se usa **keep state** para especificar el estado de información [URL, 27].

De igual manera se especifica la regla para el tráfico ICMP y UDP

```
pass out on xl0  ext proto { udp icmp } from (xl1) to any keep state
```

Ahora se escriben las reglas de filtrado para la interfaz xl1 entrante, primero se bloquea todo el tráfico de entrada:

```
block in on xl1 all
```

Posteriormente se permite el paso del tráfico proveniente de la red local:

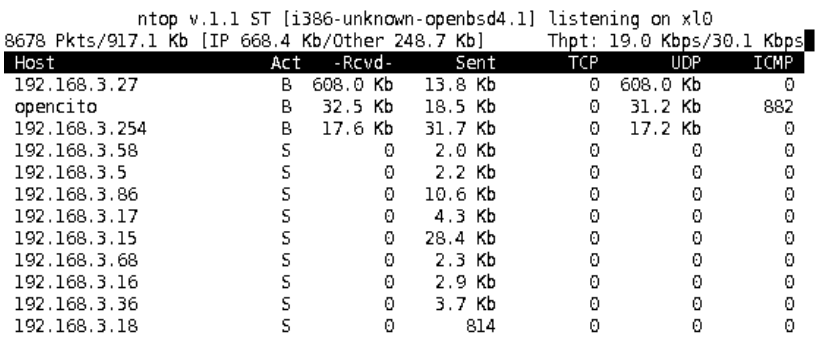
```
pass in on xl1 from red_local
```

Y una vez que se da paso a este tipo de tráfico se hace las reglas de filtrado (para el tráfico TCP y UDP) para la interfaz **xl1** saliente para cada una de las colas ya mencionadas anteriormente, cabe mencionar que red_local es una macro que define la red sobre la cual se va a implementar el PF.

```
pass out on xl1 proto tcp  from  any port $ssh_ports to red_local queue (std_in, ssh_im_in)
block out on xl1 all
pass out  on xl1 from any to red_local
pass out on xl1 proto { tcp udp } from any port domain to red_local queue dns_in
pass out on xl1 proto tcp  from  any port $ssh_ports to red_local queue (std_in, ssh_im_in)
```

5.6.1 RESULTADOS DEL PF

Antes de configurar y activar la herramienta PF del OpenBSD se hace un monitoreo de la misma manera como se realizó con el sistema Linux, utilizando la herramienta Ntop (ver.1.1), la cual no cuenta con soporte gráfico para éste sistema operativo.



ntop v.1.1 ST [i386-unknown-openbsd4.1] listening on xl0
8678 Pkts/917.1 Kb [IP 668.4 Kb/Other 248.7 Kb] Thpt: 19.0 Kbps/30.1 Kbps

Host	Act	-Rcvd-	Sent	TCP	UDP	ICMP
192.168.3.27	B	608.0 Kb	13.8 Kb	0	608.0 Kb	0
opencito	B	32.5 Kb	18.5 Kb	0	31.2 Kb	882
192.168.3.254	B	17.6 Kb	31.7 Kb	0	17.2 Kb	0
192.168.3.58	S	0	2.0 Kb	0	0	0
192.168.3.5	S	0	2.2 Kb	0	0	0
192.168.3.86	S	0	10.6 Kb	0	0	0
192.168.3.17	S	0	4.3 Kb	0	0	0
192.168.3.15	S	0	28.4 Kb	0	0	0
192.168.3.68	S	0	2.3 Kb	0	0	0
192.168.3.16	S	0	2.9 Kb	0	0	0
192.168.3.36	S	0	3.7 Kb	0	0	0
192.168.3.18	S	0	814	0	0	0

Figura 5.10. Monitoreo antes de configurar el PF

En la Figura 5.10, se puede ver que el monitoreo arroja datos en la transmisión de los paquetes con poca información ya que solo muestra los paquetes enviados desde los equipos de la red, pero no captura los paquetes que recibe, ni la clasificación de UDP o TCP, esto debido a que aún no se cuenta con la configuración correspondiente en el archivo PF.

Una vez que se configura el PF con la clasificación correspondiente del tráfico, el resultado del monitoreo es el siguiente:

ntop v.1.1 ST [i386-unknown-openbsd4.1] listening on xl0
5093033 Pkts/507.1 MB [IP 299.8 MB/Other 207.3 MB] Thpt: 15.0 Kbps/201.5 Kbps

Host	Act	-Rcvd-	Sent	TCP	UDP	ICMP
192.168.3.27	B	218.7 MB	1.1 MB	180	218.7 MB	4.0 Kb
192.168.3.85	S	9.1 MB	429.0 Kb	9.1 MB	786	0
192.168.3.86	S	2.6 MB	1.8 MB	2.6 MB	2.6 Kb	0
192.168.3.16	S	996.1 Kb	457.2 Kb	992.9 Kb	2.9 Kb	0
192.168.3.4	S	474.3 Kb	346.5 Kb	465.0 Kb	8.4 Kb	0
opencito	B	431.2 Kb	920.3 Kb	146.9 Kb	256.9 Kb	0
192.168.3.36	S	303.1 Kb	533.3 Kb	295.6 Kb	6.8 Kb	0
192.168.3.14	S	278.1 Kb	458.7 Kb	277.9 Kb	0	0
192.168.3.70	S	196.9 Kb	245.5 Kb	196.2 Kb	0	0
192.168.3.7	S	183.2 Kb	234.6 Kb	180.8 Kb	92	0
192.168.3.254	B	151.5 Kb	35.8 MB	387	116.0 Kb	980
192.168.3.68	S	144.3 Kb	771.2 Kb	141.7 Kb	0	0
192.168.3.40	S	127.8 Kb	427.0 Kb	119.0 Kb	2.0 Kb	0
192.168.3.48	S	51.4 Kb	260.1 Kb	47.6 Kb	450	0
192.168.3.15	S	47.9 Kb	4.2 MB	40.5 Kb	4.5 Kb	0

Figura 5.11. Monitoreo después de configurar el PF

En la Figura 5.11 se observa una clasificación del tráfico, en donde se muestran los paquetes que están circulando en la red, tanto los enviados como los recibidos, la cantidad de paquetes TCP y UDP. Se nota que existen 2 equipos que en ese momento hacen un ping entre ellos generando tráfico ICMP.

5.7 Pruebas

Una vez implementadas las políticas de QoS en Linux se hace la prueba del comportamiento de la red, bajando archivos de Internet para verificar la tasa de transferencia con que se descargan, el archivo de prueba es de un tamaño de 4.48MB, cabe mencionar que es un tamaño promedio de los archivos descargados por los usuarios en esta hora pico que comprende el periodo de las 9:30 a.m. a las 13:00 hrs, junto con la descarga del archivo se realiza al mismo tiempo la transferencia de un archivo de 908 bytes por FTP al servidor Web local, el cual tardó 0.10 segundos, y el uso del HTTP, cabe mencionar que antes de tener QoS en la red, el tiempo de descarga y transferencia del archivo de 4.48 MB, es de 2 minutos con 50 segundos y 27 KB/seg respectivamente.

Teniendo QoS en la red, la tasa de transferencia es de 44.1 KB/seg y el tiempo de descarga es de 1 minuto con 44 segundos. La descarga de páginas Web se hace de manera completa y sin interrupción, en cuanto a la transferencia local de archivos vía FTP se hace con pequeñas interrupciones, la prueba se realizó al subir un archivo de 908 bytes en un tiempo de 0.49 segundos, lo importante de aplicar QoS es que la navegación en Internet sea constante y completa, al final se hace la entrega de los archivos FTP, se descarga el archivo y se cumple el objetivo de navegar sin ningún problema de interrupción.

Otra de las pruebas que se realizaron fue descargar el mismo archivo pero sin hacer transferencia FTP y la descarga se realizó en 21.0 segundos con una tasa de transferencia de 218 KB/seg.

También se hizo la prueba de TTL (*Time To Live*, Tiempo de vida o latencia), en Linux con y sin aplicar QoS y en el sistema OpenBSD antes y después de activar la herramienta PF, los resultados se ven en la tabla 5.6. El número de paquetes que se captura es indistinto, ya que se puede probar con cinco, diez o más, lo importante es hacer notar la pérdida que existen sin aplicar QoS, se sugiere probar a partir de 5 paquetes pues con este número ya se puede apreciar una diferencia.

Tabla 5.6. Comparación antes y después de aplicar QoS

	Linux	**Descripción**
Sin QoS	6 packets transmitted, 5 received, 16% packet loss, time 5021ms rtt min/avg/max/mdev = 400.740/652.492/1189.412/276.142 ms	Se transmiten 6 paquetes y solo se reciben 5 de los cuales un 16% se pierden, en un tiempo de 5021 ms.
Con QoS	5 packets transmitted, 5 received, 0% packet loss, time 4025ms rtt min/avg/max/mdev = 245.267/288.929/360.313/38.117 ms	Se transmiten 5 paquetes y se reciben los 5 con un 0% de perdidos, en un tiempo de 4025 ms.
	Open BSD	**Descripción**
Sin PF	9 packets transmitted, 8 received, 11.16% packet loss round-trip min/avg/max/std-dev = 659.721/938.874/1250.865/163.50 ms	Se puede notar al hacer un ping cuantos paquetes se pierden y el tiempo de respuesta hacia nuestro servidor, con un ttl de 250 en un tiempo promedio de 938.874 ms.
Con PF	8 packets transmitted, 8 received, 0% packet loss round-trip min/avg/max/std-dev = 580.137/760.069/836.112/78.069 ms	Se transmiten 8 paquetes y se reciben 8 con un 0% de perdidos, en un tiempo promedio de760.069 ms.

En resumen, se determina que el comportamiento de la red después de implementar QoS es favorable y las tareas cotidianas se llevan a cabo sin interrupciones, y sin desperdicio de ancho de banda, la pérdida de paquetes es nula y se hacen la entrega de los mismos en un tiempo mínimo.

De la misma manera los resultados de aplicar PF en OpenBSD se observan también en la tabla 5.6, donde el total de paquetes transmitidos es el mismo número de paquetes recibidos. En ambos resultados se cumple con el objetivo de garantizar un ancho de banda para cada aplicación de las cuales se hace uso en la red.

5.8 Análisis de resultados de la aplicación de QoS y PF

Una vez que se obtienen los resultados de las dos maneras de implementar QoS en una red, se hace un análisis del comportamiento de ésta, para saber con que método se trabaja mejor o con cual de ellos se tienen mejores resultados. Para este caso de estudio y tomando en cuenta la configuración que se tiene en la red local, se pueden detallar más los resultados al aplicar QoS con Linux y sus herramientas.

El proceso de clasificación de tráfico y de asignar prioridades es mucho más flexible, la configuración se hace desde cero y se comprende mejor la manera de cómo trabaja cada herramienta de Linux, además en Linux el reenvío de paquetes se hace en menos tiempo que en el OpenBSD, y el filtrado de paquetes es más rápido. Muchas de las ventajas mencionadas son logradas debido a que el propio administrador de red hace la clasificación del tráfico y la asignación de prioridades.

Se determina que tanto Netfilter como IProute2 son herramientas que se complementan para poder realizar QoS. Por otra parte PF de OpenBSD es más cómodo para aplicar QoS ya que no es necesario recompilar el Kernel ni aplicar ningún parche de actualización y sólo en un archivo del sistema (el archivo pf.conf) se escriben las reglas

de filtrado, pero se tiene la desventaja de no poder conocer a detalle cada una de las partes de los paquetes que hacen funcionar la QoS, porque las versiones de las aplicaciones para monitorear el tráfico están muy atrasadas y no van a la par del sistema Linux. En el OpenBSD la razón por la que no se cuenta con suficiente soporte para entorno gráfico es por seguridad ya que no se recomienda trabajar bajo este entorno, y el sistema se maneja en línea de comandos.

A diferencia de Linux en el OpenBSD si un paquete coincide con una regla que contenga la opción quick, se considera como la regla final y se tomará la acción que esté especificada sin tener que pasar por el resto de las reglas. Como lo importante es lograr la implementación de QoS queda a consideración de cada administrador elegir la implementación de una de las dos formas de aplicar QoS, aquí presentadas.

Una es utilizando la herramienta PF del OpenBSD, si lo que se requiere es rapidez, eficacia y comodidad, y otra es utilizando IProute2 y Netfilter de Linux, si además lo que se desea es conocer y aprender más sobre reconfiguración del Kernel, parches y realizar paso a paso cada regla para tener el control de su propio script de QoS. Se recomienda ésta última forma, si además, debido al crecimiento en un futuro que se pudiera dar en su red o en el trascurso de su administración, se busca hacer las modificaciones que desee, como por ejemplo, actualizar algún paquete como es Netfilter o IProute2.

Capítulo 6

6.1 Conclusiones

Las herramientas que permiten hacer distribución del ancho de banda para optimizar los recursos de una red y ofrecer QoS se dividen en hardware y software.

Las herramientas en hardware tienen un costo, lo cual representa una desventaja con respecto a las herramientas de software, puesto que muchas de ellas son de libre distribución, además, permiten adaptarse a las necesidad del usuario proporcionando enrutamiento selectivo, clasificación y control de tráfico, acceso remoto a redes y balanceo de carga.

En este trabajo se tuvo como objetivo general analizar las herramientas IProute2 y Netfilter para marcar la prioridad de servicio en los paquetes de información trasmitidos en sistemas Linux.

Para poder cumplir con este objetivo se realizaron las siguientes tareas:

- Definir y explicar el funcionamiento de la QoS en el sistema operativo Linux.

- Listar y mostrar las características generales de algunos modelos para satisfacer la QoS en redes, tales como los Servicios Diferenciales y los Servicios Integrales en Internet.
- Describir el funcionamiento de las herramientas IProute2 y sus tablas de enrutamiento así como, para conocer el funcionamiento IPtables.
- Determinar que IProute2 y Netfilter trabajan de manera complementaria.
- Configurar el PF de OpenBSD para comparar con las dos herramientas comprobando que su implementación es más sencilla.
- Configurar IProute2 y Netfilter usando los conceptos de filtro y colas para priorizar las reglas adecuadas a la red.
- Realizar pruebas verificando su buen funcionamiento dentro de la red.

Al hacer la comparativa de las dos maneras de aplicar QoS, se comprobó que si se garantizan la QoS pero en la implementación se realiza de diferente manera y queda a consideración del propio administrador elegir cual ocupará en base a su criterio y experiencia.

Se recomienda el uso de IProute2 y Netfilter, ya que a pesar que su implementación es poco sencilla es necesario conocer más del sistema y esto permite una mejor forma de controlar el tráfico y la asignación de prioridades. La mejora de tener la red con QoS, fue aproximadamente de un 80% con respecto a no contar con ella, y las tareas cotidianas de los usuarios no se vieron interrumpidas durante la jornada laboral.

La implementación de QoS en una red no se considera una receta a seguir, ya que los escenarios de trabajo varían dada las circunstancias de cada diseño e implementación de una red, considerando muchas veces también la ubicación geográfica, puesto que la infraestructura es diferente. Para condiciones de trabajo similares se pueden seguir los pasos descritos en el Anexo2.

Cabe mencionar que este trabajo de tesis se contempla únicamente la implementación de QoS para tener el control de tráfico de salida, ya que para hacer el control de tráfico de entrada se necesita hacer uso del IMQ[12] (Intermediate Queueing Device, Dispositivo intermedio de encolado), lo cual lleva mas tiempo en cuanto a la implementación, pues se trata de controlar información externa hacia la red que se va a aplicar QoS, y es más fácil controlar lo que los usuarios pueden enviar hacia fuera por que se tiene el control de manera local, y el objetivo principal de este trabajo es garantizar un buen funcionamiento de la red el cual se cumple con la aplicación de QoS al tráfico de salida o subida.

Existen diversos trabajos [16] y [URL,8] donde se han utilizado como base los principios de asignación de prioridades y creación de clases para ciertos tipos de aplicación o tráfico, aparentemente se pueden ver como iguales dichas implementaciones, sin embargo, esto no es así de sencillo, ya que se trata de una constante manipulación de los parámetros en la asignación del ancho de banda, en la creación de reglas y hacer el típico prueba y error hasta coincidir con los valores que garanticen una buena asignación,

[12] Dispositivo que sirve para controlar el tráfico de bajada o de entrada

en este trabajo de tesis, se aporta una investigación, implementación y comprobación de las herramientas que ofrece Linux para aplicar QoS.

6.2 Trabajos futuros

Debido al extenso campo sobre el tema de QoS, este trabajo puede ser una base para algunos trabajos a futuro como:

- La creación de una herramienta que permita monitorear la QoS en una red donde se pueda observar gráficamente cada una de las aplicaciones a las cuales se les está marcando una prioridad.

- Creación de una herramienta similar al NTOP que sea compatible con el sistema operativo OpenBSD, que permita monitorear el estado de una red.

- Aplicar QoS para poder controlar el tráfico de entrada ya que en este trabajo solo se está haciendo QoS para el tráfico de salida.

- Crear un sistema que permita de forma automatizada el análisis de tráfico de red y pueda crear sus propias reglas y aplicarlas para lograr la QoS más adecuada a la red.

- Modelar el proceso de asignación de Qos de IProute2 y Netfilter con una herramienta de análisis y diseño como lo es UML.

ACRÓNIMOS

A

AF Assured Forwarding. Encaminamiento Asegurado.

ALTQ Alternate Queueing, Encolamiento Alterno

ARP Address Resolution Protocol. Procolo de resolución de direcciones.

B

BA Behavior Aggregate. Comportamiento Agregado.

BANDWIDH Ancho de banda.

BGP Border Gateway Protocol. Protocolo de Enrutamiento Exterior.

BGP-4 Border Gateway Protocol. Protocolo de Enrutamiento Exterior Versión 4.

C

CBQ Class Based Queueing. Disciplina de colas con clase. Limita el ancho de banda máximo.

CoS Class of service, Clase de servicio.

CQ Custom Queuing. Encolamiento personalizado.

D

DIFF-SERV Differentiated Services. Servicios Diferenciales.

DoS Deny of Service. Denegación de Servicio.

DS Diferential Service. Servicio Diferencial –Conocido como campo DS definido para los Servicios Diferenciales.

DSCP Differentiated Services Codepoint. Punto de Codificación de Servicios Diferenciados.

E

EF Expedited Forwarding. Encaminamiento Expedito.

EGP Exterior Gateway Protocols. Protocolos de Enrutamiento Exterior.

EIGRP Protocolo de Enrutamiento interior de gateway mejorado.

F

FEC Forwarding Equivalence Class. Clase Equivalente de Envío.

FIFO First In, First Out,. Primero que entra, Primero que sale. Disciplina de encolamiento por defecto.

FIFO_FAST Disciplina de colas que sigue la política de FIFO, Pfifo_fast está formada por tres bandas.

G

GTS Generic Traffic Shaping. Regulación del ancho de banda.

H

HTB Tiene la misma función del CBQ, solo que esta disciplina imita el ancho de banda mínimo disponible.

I

IETF Internet Engineering Task Force. Grupo de Trabajo en Ingeniería de Internet.

IGP Interior Gateway Protocols. Protocolos de Enrutamiento Interior.

IGRP Interior Gateway Routing Protocol. Protocolo de enrutamiento de gateway interior.

INT-SERV Integrated Services. Servicios Integrales.

ISO The Organization International for Standardization. Organización Internacional de Estandares.

M

MF Multi-Field, Multi-Campo.

MPLS Multi Protocol Label Switching. Conmutación de Etiquetas Multiprotocolo.

N

NAP Neighbor Acquisition Protocol. Protocolo de Adquisición al Vecino.

NR Neighbor Reachability. Accesibilidad de Vecino, usado en el BGP.

O

OSI Open System Interconnection. Interconexión de Sistemas Abiertos.

OSPF Open Shortest Path First. Abrir Primero la Trayectoria más corta.

P

PF PacketFilter. Filtrado de paquetes. Herramienta del OpenBSD.

PHB Per Hop Behavior. comportamiento por saltos.

PQ Priority Queuing. Cola con prioridad.

Q

QdiscPRIO Cola con prioridad. Esta cola divide el tráfico pero no hace ajustes.

QoS Quality of Service. Calidad de Servicio.

R

RED Random Early Detection. Gestión de la congestión.

RIP Routing Information Protocol. Protocolo de Información de Enrutamiento.

RSVP Resource ReSerVation Protocol. El Protocolo de Reservación de Recursos.

S

SA Autonomous System. Sistemas Autónomos.

SFQ Stochastic Fairness Queueing. Disciplina de Colas Justa. El tráfico se envía de una forma parecida a Round Robin.

SMB Subset Bandwidth Manager. Gestor de ancho de banda de subred.

T

TBF Disciplina de colas sin clase, se usa cuando se quiere controlar la tasa máxima de paquetes que se envía desde una cola.

tc Traffic Control. Control de Tráfico.

TC Traffic Class. Clase de tráfico.

ToS Type of Service. Tipo de Servicio.

TTL Time To Live. Tiempo de vida o latencia.

W

WFQ Weighted fair queuing. Cola ecuánime ponderada o por peso.

WRED Weighted Random Early Detection. Gestión de la congestión mejorada.

REFERENCIAS

[1] Braden, R., Clark, D., and S. Shenker, Integrated Services in the Internet Architecture, RFC 1633, Junio 1994.

[2] E. Rosen, Multiprotocol Label Switching Architecture, Enero 2001.

[3] Sedano, F. C. Enrique “QoS y mecanismos de transición IPv4/IPv6”, Area de Ingeniería Telemática Universidad CARLOS III de Madrid, 2002.

[4] Matthew G. Marsh, Encaminamiento regulado con Linux. Pearson Educación, S.A. Madrid, 2001.

[5] Sierver Ellen, Figgins Stephen, Weber E. Aaron, Linux In a Nutshell 4th Edition, Paperback, Ed. O’really, 2005.

[6] ISO, International Organization for Standardization. Quality of Service – Framework, ISO/IEC DIS 13236, 1996.

[7] Gallo, Michael A., Hancock William M. Comunicación entre computadoras y tecnologías de redes. Ed. Thomson, 20028

[8] Darpa Internet Program, Internet Potocol, RFC [791], Septiembre 1981.

[9] K. Nichols, Definition of the Differentiated Services Field (DS Field) in the IPv4 and and IPv6 Headers, RFC 2474, Diciembre 1998.

[10] Wang Zheng, Internet QoS, Architectures and Mechanisms for Quality of Service Publisher MK. 2001.

[11] P. Almquist, Type of Service in the Internet Protocol Suite, RFC 3031, Julio 1992.

[12] V. Jacobson, An Expedited Forwarding PHB, RFC 2598, Junio 1999.

[13] J. Heinanen, Assured Forwarding PHB Group, RFC 2597, Junio 1999.

[14] Zwicky Elizabeth, D. Cooper Simon & Chapman D. Brent, Building Internet FIREWALLS, O’really, 2nd Edición, 2000.

[15] E. Comer Douglas. Redes globales de información con internet y TCP/IP, Prentice-Hall, 3ra. Edición, 1996.

[16] Fuster Monzó Joan, filter_qos Universidad Politécnica de Valencia, 2004.

[17] Lucas Michael W. Absolute Openbsd Unix For The Practical Paranoid, ISBN:1886411999, 2003.

URL'S

[URL, 1] M. Moreno M. Señalización para redes IP, http://www.aui.es/biblio/bolet/bole025/art_4.htm, Último acceso: Noviembre, 2006.

[URL, 2] Enrutamiento avanzado y control de tráfico en Linux. http://aria.eui.upm.es/~svinter/Enrutamiento-avanzado-y-control-de-trafico-en-Linux.pdf Último acceso: Noviembre 2006.

[URL, 3] Optimizando tráfico de Internet con Linux, http://cofradia.org/~aavelar/html/html/linux/servidor_gestor.htm, Último acceso: Diciembre 2006.

[URL, 4] Presentación de QoS en walc, www.walc2002.pucmm.edu.do/material/taller4/QoS.ppt , Último acceso: Diciembre 2006.

[URL, 5] Tutorial de QoS: http://long.ccaba.upc.es/long/050Dissemination_Activities/alberto_lopez_QoStutorial.pdf. Último acceso: Enero 2007.

[URL, 6] Artículo:QoS y mecanismos de transición IPv4/IPv6, http://www.it.uc3m.es/~fvalera/int_red/trabajos/IRSBAv1.pdf, Último acceso: Enero de 2007.

[URL, 7] Linux Traffic Control, http://www.opalsoft.net/qos/DS-21.htm, Último acceso: Marzo 2007.

[URL, 8] Tesis doctoral electrónica, http://it.aut.uah.es/josema/publicaciones/tesis.pdf Propuesta de optimización de la interconexión de redes con calidad de servicio para aplicaciones multimedia, José Manuel Arco Rodríguez. 2000. Último acceso: Enero 2007.

[URL, 9] Traffic Control HOWTO,http://www.tldp.org/HOWTO/Traffic-Control-HOWTO/software.html#s-IProute2 . Último acceso: Febrero 2007.

[URL, 10] Calidad de Servicio en Redes de Servicios Diferenciados, http://www.tid.es/presencia/publicaciones/comsid/esp/24/aart5.pdf. Último acceso: Febrero 2007.

[URL, 11] Linux Advanced Routing & Traffic Control HOWTO, http://lartc.org/howto/ Último acceso: Marzo 2007.

[URL, 12] NAT,http://www.monografias.com/trabajos16/red-local-internet/red-local-internet.shtml. Último acceso: Marzo 2007.

[URL, 13] Netfilter et Iptables Architecture, http://christian.caleca.free.fr/Netfilter/architecture.htm. Último acceso: Abril 2007.

[URL, 14] Firewalling with Netfilter/Iptables, http://www.knowplace.org/Netfilter/index.html. Último acceso: Mayo 2007.

[URL, 15] Configuración de un firewall en Linux,http://www.uazuay.edu.ec/estudios/sistemas/teleproceso/ciclo2/firewall.doc. Último acceso: Mayo 2007.

[URL, 16] Proyecto FIREWALL, http://computo.fisica.unam.mx/firewall.html. Último acceso: Junio 2007.

[URL, 17] Introducción a Firewalls,http://ww.interlan.com.co/firewall.htm. Último acceso: Julio 2007.

[URL, 18] TCP/IP y Enrutamiento,http://iie.fing.edu.uy/ense/asign/admunix/tcpip.htm. Último acceso: Agosto 2007.

[URL, 19] TCP/IP Tutorial and Technical Overview, http://ditec.um.es/laso/docs/tut-tcpip/3376c34.html. Último acceso: Noviembre 2007.

[URL, 20] Routing Basics, http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Routing-Basics.pdf. Último acceso: Agosto 2008.

[URL, 21] Effective Traffic Measurement Using ntop, IEEE Communications Magazine. May 2000. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=841838. Último acceso: Diciembre 2007.

[URL, 22] Manual de Ethereal, http://www.planeta-digital.com. Último acceso: Diciembre 2007.

[URL, 23] Instalación de IProute2, http://www.escomposlinux.org/lfs-es/lfs-es-SVN/chapter06/iproute2.html. Último acceso: Diciembre 2007.

[URL, 24] PF: Packet Filtering: http://www.openbsd.org/faq/pf/filter.html#intro. Último acceso: Febrero de 2008.

[URL, 25] OpenBSD 4.1 Installation, http://www.andsux.info/web/uploads/obsd.html. Último acceso: Marzo de 2008.

[URL, 26] QoS Con Pf, http://www.bsd.cl/wiki/index.php?title=QoS_Con_Pf. Último acceso: Abril de 2008.

[URL, 27] PF, ftp://ftp.openbsd.org/pub/OpenBSD/doc/pf-faq.pdf. Último acceso: Abril de 2008.

Anexo1

Activación de opciones en el Kernel, para implementar QoS.

En esta parte lo que se hace es activar las opciones que se necesitan para llevar a cabo QoS, las opciones pueden puede ser instaladas como módulo (M), o de manera permanente en el sistema (*), para este caso todas fueron instaladas de manera permanente, así en caso de reiniciar el sistema no se tienen que cargar nuevamente para ser utilizadas.

Networking →

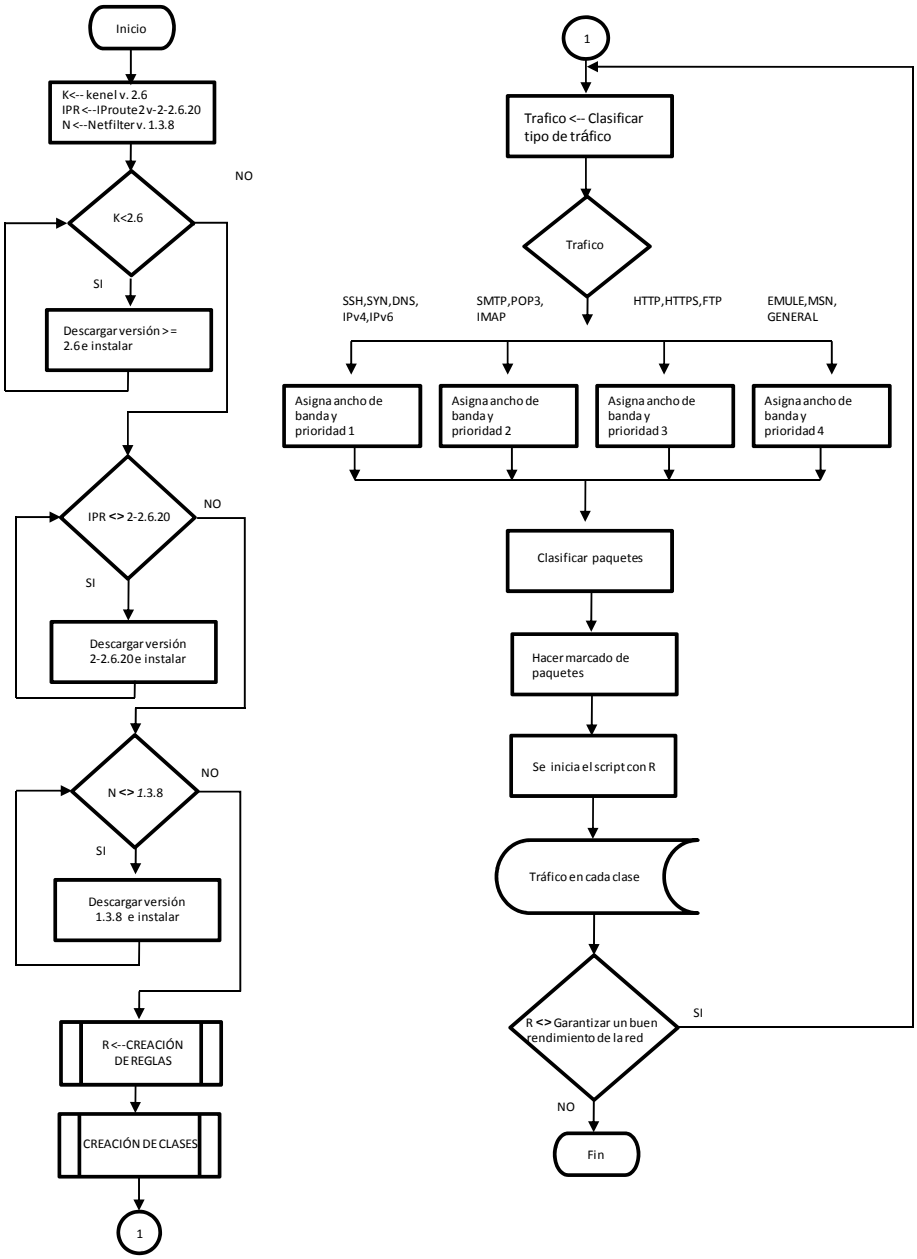
Networking Options →

```
 [ * ]  QoS and/or fair queueing
--- Queueing/Scheduling                  /* programación de colas */
< * > Class Based Queueing (CBQ)         /* permite hacer préstamo de ancho de banda  */
< * > Hierarchical Token Bucket (HTB)    /* similar a CBQ pero usa diferente algoritmo*/
< * > Hierarchical Fair Service Curve (HFSC)
--- Multi Band Priority Queueing (PRIO)          /* Consiste en dividir el tráfico en bandas    */
< * > Multi Band Round Robin Queuing (RR)
< * > Random Early Detection (RED)
< * > Stochastic Fairness Queueing (SFQ)       /*separa el tráfico con hash en  varias FIFO*/
< * > True Link Equalizer (TEQL)   /* combina varias rede físicas en una red virtual */
< * > Token Bucket Filter (TBF)                  /* limita la tasa de transmisión de paquetes */
< * > Generic Random Early Detection (GRED)
< * > Differentiated Services marker (DSMARK)        /* marcador de servicios diferenciados */
< * > Network emulator (NETEM)
< * > Ingress Qdisc
--- Classification
< * > Elementary classification (BASIC)
< * > Traffic-Control Index (TCINDEX)      /* es un clasificador especialmente para los DS * /
< * > Routing decision (ROUTE)
< * > Netfilter mark (FW)                                    /*   marcador fw */
< * > Universal 32bit comparisons w/ hashing (U32)         /* filtro U32 compara con hash */
 [ * ]    Performance counters support
 [ * ]    Netfilter mask support                    /* soporte de enmascaramiento con Netfilter */
< * > IPv4 Resource Reservation Protocol (RSVP)
< * > IPv6 Resource Reservation Protocol (RSVP6)
 [  ]  Extended Matches
 --- Actions
 --- Traffic Policing
 <  > Generic actions
```

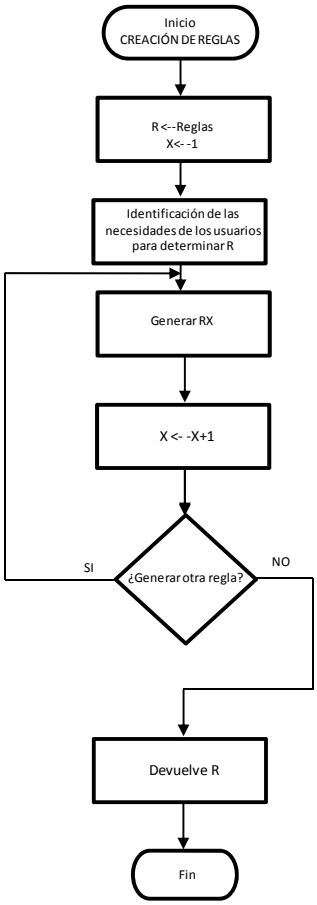
```
<   > Redirecting and Mirroring
< * > IPtables targets                                  /* etiquetas de iptables*/
< * > Packet Editing
<   > Simple Example (Debug)
 [ * ]   Traffic Policing (obsolete)
 [   ]     Incoming device classification
```

Anexo2

Diagrama de flujo para la implementación de QoS.



Módulo de creación de reglas



Módulo de creación de clases