UNIVERSIDAD NACIONAL DE INGENIERÍA
RECINTO UNIVERSITARIO SIMÓN BOLÍVAR
FACULTAD DE ELECTROTECNIA Y COMPUTACIÓN

Propuesta de Control de Tráfico de Redes Informáticas con el Sistema Operativo Linux.

Caso de Estudio: Red Interna UNI-RUSB

Proyecto de Tesis Monográfica para optar al título de Master en Redes de Computadoras

Autores:

María de Lourdes Avilés Narváez

Alfonso Martín Boza Tuckler

Tutor:
Ing. MSc. Janine Mairena Solórzano

Managua, 27 de Abril 2009

Dedicatoria

Dedico este trabajo a nuestro Señor Jesús, quien es el autor y consumador de nuestra fe y confianza, que nos a permitido culminar este trabajo con éxito.

Alfonso Boza

Dedico este trabajo a Dios, quien me ha dado la Luz, la fortaleza, los medios y la dirección para empezar y culminar este proyecto de investigación.

María de Lourdes Avilés

Resumen

El presente trabajo de tesis, *Propuesta de Control de Tráfico de Redes Informáticas con el Sistema Operativo Linux. Caso de Estudio: Red Interna UNI-RUSB,* proporciona una alternativa para realizar control de tráfico diferenciado en la red informática de la UNI.

La propuesta incluye tres partes importantes, que son las siguientes: la base teórica del control del tráfico diferenciado en Linux, un algoritmo que proporciona los pasos y parámetros necesarios a considerar en una red para realizar el control del tráfico diferenciado en Linux, y un caso de estudio particular, en este caso, la red informática UNI, el cual contiene un conjunto de reglas propuestas para la realización del control del tráfico diferenciado de la red.

El caso de estudio está basado en la teoría del control de tráfico diferenciado en Linux y en el algoritmo elaborado. Las reglas de control de tráfico, han sido escritas de manera flexible, tomando en cuenta la posibilidad de cambios topológicos en la red y cambios de decisión del administrador, en relación a restricciones de usuarios, prioridades en la ejecución de los servicios, uso de los servicios y recursos de la red.

INDICE

I. Introducción 7

II. Objetivos 8

1. General 8

2. Específicos 8

III. Antecedentes 9

IV. Justificación 10

V. Marco Teórico 11

Capítulo 1. Control de Tráfico en Linux 11

1. Calidad de Servicio 11

2. Disciplinas de Cola para Gestión de Ancho de Banda 16

2.1. Disciplinas de Cola Simples (sin clases) 16

- pfifo_fast | bfifo_fast 16
- Token Bucket Filter (TBF) 19
- Stochastic Fairness Queueing (SFQ) 21
- Extended Stochastic Fairness Queueing (ESFQ) 22
- Random Early Detection (RED) 23

2.2. Disciplinas de Cola con Clases 24

- Qdisc PRIO 25
- DSMARK 27
- Class Based Queueing (CBQ) 28
- Hierarchical Token Bucket (HTB) 29

2.3. Consejos sobre qué Disciplina de Cola usar en determinado momento 33

3. Técnica de Encolado 34

4. Herramientas de Control de Tráfico para Gestión de Ancho de Banda en Linux 34
4.1. Iptables 34
4.2. Trafic Control – TC 46
- Colas 46
- Clases 47
- Filtros o clasificadores 47
4.3. Ejemplo de Control de Tráfico con la Disciplina de Cola HTB 52
5. El Dispositivo Intermedio de Encolado IMQ 64
VI. Análisis y Presentación de Resultados 66
Capítulo 2. Algoritmo de Control de Tráfico en Linux 66
Capítulo 3. Control de Tráfico. Caso de Estudio: Red Informática UNI. 72
VII. Conclusiones 89
VIII. Recomendaciones 89
IX. Referencia Bibliográfica 90
X. Glosario de Términos 92
XI. Anexos 101
Anexo 1. Mapa Topológico de la red informática de la UNI 102
Anexo 2. Estadísticas de Red 103
Anexo 3. Distribución del Ancho de Banda 106
Anexo 4. Script para generación de las Reglas de Control de Tráfico Propuestas para el Gateway 8. 107
Anexo 5. Análisis de las Estadísticas del Tráfico. 115

I. Introducción

El análisis del tráfico de una red permite obtener datos importantes para gestiones administrativas. Al determinar el uso de la red, es posible obtener una estadística real de lo que ocurre, lo que facilita a los administradores de la red, darle seguimiento y mantenerla dentro de parámetros aceptables de buen desempeño, por ejemplo, priorización de tipos de tráfico y uso de ancho de banda. **[Ref[1]. u]**

Comúnmente, en las instituciones y empresas, los empleados usan los recursos de la red para realizar actividades que no son críticas para el negocio, tales como, descargas de audio / vídeo y el uso de programas P2P; éstas aplicaciones aumentan cada vez más la demanda de los recursos de red y hasta pueden colapsar el tráfico de la misma. En vista de este problema, el sistema operativo Linux, permite desarrollar distintos tipos de QoS (Quality of Service) para controlar el tráfico, a través de una variedad de funciones de control de tráfico en el Kernel y el espacio del usuario, con las cuales los administradores pueden establecer sus propios controles (reglas de control de tráfico), de forma que sea posible lograr una diferenciación del tráfico de la red.

La diferenciación del tráfico les facilitaría a los administradores asegurar prioridades por tipos de tráfico y realizar entre ellos, una distribución equitativa del ancho de banda disponible, por ejemplo, para descarga de archivos, navegación Web, correo electrónico, chat, transferencia de archivos, etc. **[Ref. o]**

En el caso particular del personal administrativo de la red interna del Recinto Universitario Simón Bolívar – RUSB en la Universidad Nacional de Ingeniería – UNI (la cual es administrada con el Sistema Operativo Debían GNU / Linux) requiere de un proceso definido que le oriente en el establecimiento de sus reglas internas de control de tráfico. Por tanto, en el presente documento, se plantea elaborar una propuesta alternativa de control de tráfico, para una muestra de la red UNI-RUSB, definiendo cada uno de los pasos y parámetros a tomar en cuenta para realizar esta tarea.

[1] Ref. hace alusión a Referencia Bibliográfica, que se encuentra al final de este documento

II. Objetivos

1. General

- Brindar una alternativa para el control diferenciado del tráfico en la red informática de la UNI, con el fin de establecer restricciones y políticas de acceso a la misma.

2. Específicos

- Analizar los elementos del control de tráfico del kernel y programas de usuario de Linux.

- Analizar la red interna de la UNI, tanto en los servicios que proporciona como en su topología de conexión.

- Proponer un conjunto de pasos definidos y reglas de control de tráfico diferenciado, para una muestra de la red de la UNI.

III. Antecedentes

La División de Informática y Tecnología de la Información DITI-UNI, ha sido creada por la Universidad Nacional de Ingeniería - UNI, para apoyar la planificación y desarrollo de alternativas de solución a las necesidades TIC de las entidades y comunidad Universitaria. La DITI está compuesta de una variedad de oficinas responsables de llevar a cabo diferentes tareas, entre ellas, garantizar a los usuarios la disponibilidad y uso adecuado de los servicios y recursos de la red.

La creciente demanda de recursos informáticos en la universidad, ha permitido una expansión de su red interna, lo que ha provocado un mayor consumo de ancho de banda. Como consecuencia, ha surgido la necesidad de establecer medidas de control de tráfico de la red.

Hasta el momento, la DITI-UNI, ya ha establecido algunos controles avanzados en su tráfico, con el sistema Netfilter / Linux, los que son de gran importancia y son los siguientes:

- Reglas para restricción de acceso a servicios, que ha estimado conveniente hacerlo. Por ejemplo, el bloqueo de Skype a través de la herramienta llamada Snort_inline.
- Habilitar NAT masquerading para los ordenadores, que es necesario para que éstos tengan acceso a internet.
- Habilitar el soporte para forwarding, lo que permite que los paquetes pasen “a través” del Gateway.
- Configurar las rutas adecuadamente, para que los paquetes vayan a la tarjeta de red correcta.

Sin embargo, no le ha sido posible aplicar una alternativa definida de control de tráfico para la red, pero es de su conocimiento que se requiere diferenciar el flujo total de la red, para poder aplicarle un tratamiento diferenciado, según su clasificación.

Con base en la teoría de los expertos, la solución al problema respecto al control del tráfico con sistemas Linux, es escribir reglas de control de tráfico que permitan al administrador establecer sus propias políticas de control, de acuerdo a los servicios que

proporciona la red y las necesidades de los usuarios, basándose en el escenario de servicio y conectividad de la misma. Como ya se ha mencionado, Linux está habilitado con las funciones necesarias, para realizar esta labor.

IV. Justificación

En la actualidad, la red interna UNI-RUSB y su conexión a Internet, tiene una operación satisfactoria, según la administración técnica actual, pero, ésta no dispone de un procedimiento definido que le permita tratar el tráfico por tipos, de diferentes maneras y realizar una distribución razonablemente equitativa del ancho de banda entre los servicios, según su importancia para los usuarios. En vista de que el ancho de banda es un recurso muy limitado, se necesita de tal procedimiento, tomando en cuenta los parámetros necesarios para optimizar su uso, al establecer límites de consumo y prioridades por tipo de tráfico en la red. Esto permitiría optimizar el desempeño de la red UNI.

V. Marco Teórico

Capítulo 1. Control de Tráfico en Linux

1. Calidad de Servicio

Calidad de Servicio o QoS (*Quality of Service),* es un conjunto de tecnologías que garantizan la entrega de datos a través de una red de computadoras en un momento dado, estableciendo prioridades y tratamiento diferenciado por tipos de tráfico. Aplicar QoS a la red, significa realizar un control del tráfico de la misma. [Ref. b]

El control del tráfico en Linux, consiste en la aplicación de una serie de tecnologías disponibles en su kernel y en las herramientas del espacio de usuario, para establecer controles en el uso de los recursos de la red.

Cuando el kernel del sistema operativo, tiene muchos paquetes para enviar sobre un dispositivo de red, éste tiene que decidir cuales enviar primero, cuales retardar y cuales descartar. Este trabajo es realizado por un conjunto de algoritmos, implementados en el kernel, que han sido propuestos para este fin, y administrados por herramientas de gestión en el espacio del usuario.

Algunos de los beneficios de implantar QoS son los siguientes: [Ref. b]

- ***Control sobre los recursos:*** es posible priorizar los servicios en ejecución. Por ejemplo, el tráfico P2P puede saturar la cola de envío de los paquetes, lo que impediría que los paquetes de navegación lleguen a su destino, y por lo tanto la navegación sería muy lenta. Para evitar esta situación, se deberá priorizar el tráfico web antes que el tráfico P2P.
- ***Uso más eficiente de los recursos de red:*** es obtenido como resultado del control sobre los recursos.
- ***Menor latencia:*** se logra al priorizar el tráfico interactivo, que requiere un tiempo de respuesta corto.

Es preciso conocer los siguientes términos de control de tráfico [Ref. a y b]:

- **Qdisc o disciplina de cola**: es el algoritmo que controla la cola de paquetes de un dispositivo y es el bloque principal de control de tráfico en Linux. Una qdisc es un scheduler, cada interfaz (tarjeta de red) de salida necesita un scheduler de algún tipo para atender sus paquetes, el scheduler por defecto es una cola FIFO.

- **Qdisc root o Qdisc raíz:** es la qdisc que se encuentra adjunta a la interfaz de red.

- **Qdisc classfull o Qdisc con clases:** es la qdisc que contiene múltiples clases. Este tipo de qdiscs, permiten realizar subdivisiones internas, que a su vez pueden ser otras qdiscs.

- **Qdisc classless o Qdisc sin clases:** es la qdisc en la que no se pueden configurar subdivisiones internas.

- **Class o clase:** es una organización o agrupación de paquetes, realizada a través de filtros. Solo existen dentro de las qdisc con clases. Las qdiscs organizan sus paquetes en clases para luego ser desencolados. Una qdisc con clases puede tener múltiples clases, internas todas a la qdisc. A su vez también, pueden añadirse clases a clases, siendo llamadas clases hijas o clases internas.

- **Clases terminales u hojas:** son las clases que no tienen clases “hijas” y tienen una qdisc adjunta. Una clase hoja es siempre una clase hija, pero no viceversa.

- **Clasificadores:** ordenan o separan el tráfico entre colas. Cada qdisc con clases necesita determinar a qué clase enviar un paquete, esto lo hace usando clasificadores. Clasificación es el mecanismo por el cual los paquetes son separados para tener diferente tratamiento. El más común es el clasificador **u32**,

que permite que el usuario seleccione paquetes, en función de los atributos de los mismos.

- **Filter o Filtro**: un filtro contiene las condiciones a ser cumplidas para que los paquetes sean clasificados.

- **Scheduling (ordenamiento):** es el mecanismo por el cual los paquetes son ordenados (o desordenados) entre la entrada y la salida de una cola. Una qdisc puede, con la ayuda de un clasificador, decidir que algunos paquetes necesitan salir antes que otros. Este proceso se denomina Scheduling.

- **Shaping (ajuste):** es el proceso de retrasar paquetes antes de que salgan, para hacer que el tráfico fluya conforme a una tasa máxima configurada, generalmente, con el fin de suavizar el tráfico de ráfagas. El Shaping se realiza durante la salida («egress»). Coloquialmente, también se le suele denominar Shaping, al descarte de paquetes para ralentizar el tráfico.

- **Policing:** es un mecanismo por el cual el tráfico es limitado. Un policer aceptará tráfico a una determinada velocidad, y luego realizará alguna acción si el tráfico excede la velocidad fijada. Una de las opciones puede ser que el tráfico sea descartado o puede ser reclasificado. Un policer es una pregunta tipo si / no acerca de qué hacer con el tráfico que ingresa en una cola. Un policer no tiene la capacidad de retardar el tráfico como un "shaper".

- **Dropping:** descartar un paquete, un flujo o una clasificación.

- **Marking o Marcado de paquetes:** es el mecanismo por el cual un paquete es alterado o marcado. Los mecanismos de marcado del control de tráfico, instalan una marca en el paquete mismo, la cual es usada y respetada por otros routers dentro de un mismo dominio administrativo, para diferenciarlo del resto de paquetes.

- **Handle**: es el identificador único para una *class* y una *qdisc* dentro de la estructura de control de tráfico y tiene dos miembros constitutivos, que son, un número **major** y un número **minor**. Estos números pueden ser asignados de manera arbitraria de acuerdo a las siguientes reglas:

  Major: el usuario puede utilizar cualquier número arbitrario, sin embargo, todos los objetos que tengan el mismo padre dentro de la estructura de control de tráfico deben tener el mismo número *major*.

  Minor: este número identifica a una qdisc si es igual a 0. Cualquier otro valor significa que se trata de una clase. Todas las clases que tienen el mismo padre deben tener un *minor* diferente.

  Los handles son locales a la interfaz en la que están definidos, por ejemplo, las interfaces eth0 y eth1, pueden tener cada una de ellas, clases con el mismo handle 1:1.

  El handle ffff:0 está reservado para la ingress qdisc.
  El handle es utilizado como destino (target) en los argumentos *classid* y *flowid* del comando *tc filter*, el cual se explicará más adelante.

- **Queueing o encolado**: ocurre cuando la qdisc de salida recibe el paquete.

- **Dequeueing o desencolado**: ocurre cuando el núcleo pide que el paquete, que está en la qdisc de salida esperando, sea retransmitido por la interfaz de salida.

El kernel del sistema operativo Linux, ofrece una amplia variedad de funciones de control de tráfico, que constituyen dos partes: las partes del kernel y varios programas del espacio de usuario.

El código para control de tráfico en el kernel, consiste en los siguientes componentes principales:

- Disciplinas de Cola (queueing disciplines)

- Clases (dentro de una disciplina de cola)
- Filtros (filters)
- Vigilancia (policing)

Cada dispositivo de red tiene una disciplina de cola asociada, que controla cómo serán tratados los paquetes encolados en dicho dispositivo.

Los programas del espacio de usuario en Linux, para control de tráfico son *iptables* y *tc,* esta segunda, se encuentra dentro del paquete iproute2.

Tanto las partes del kernel, como los programas de usuario, serán estudiados en detalle, más adelante en este capítulo.

Linux, desde su versión 2.2/2.4, está habilitado con las funciones de control de tráfico y las emplea de la siguiente manera:

a. El paquete llega al kernel de Linux.

b. Netfilter establece en el paquete una marca que lo identifica dentro del kernel, para posteriormente clasificarlo. La marca consiste en un valor entero de 32 bits y el marcado puede ser realizado en base al puerto, la dirección IP del paquete, etc.

c. El paquete es clasificado. Según su marca, el paquete es ubicado en un árbol de preferencias, donde se determina el envío de los datos mediante disciplinas de cola. El proceso es realizado de la siguiente manera: las disciplinas de cola son organizadas en una estructura jerárquica de clases en forma de árbol con dependencia padre-hijo entre sus miembros, esto es llamado árbol de preferencia. Cada interfaz tiene una qdisc raíz que está asociada a ella. Cuando se reciben paquetes en la interfaz de red, la qdisc raíz recibe la petición de desencolar. Entonces, en base a las marcas en el paquete, que se hayan

establecido con netfilter, la qdisc raíz lo enviará a alguna de las clases que ella contiene.

2. Disciplinas de Cola para Gestión de Ancho de Banda

[Ref. a, e, f]

Las disciplinas de cola son algoritmos del kernel de Linux, usados para administrar el proceso de encolado de paquetes en un dispositivo (interfaz de red). Esta gestión puede ser tanto en la cola de entrada (ingress), como en la cola de salida (egress), sin embargo, la cola ingress no está aún documentada y no se puede comentar mucho al respecto. A través de las disciplinas de cola para egress, es posible cambiar el modo en que se envían los datos a la red y ayudan a controlar el uso del ancho de banda saliente de un enlace dado.

2.1. Disciplinas de Cola Simples (sin clases)

Las disciplinas de cola sin clases son aquellas que, mayormente, aceptan datos y se limitan a reordenarlos, retrasarlos, o descartarlos. Esto se puede usar para ajustar el tráfico de una interfaz entera, sin subdivisiones.

- ***pfifo_fast* | *bfifo_fast*** (First In, First Out – El primero en entrar es el primero en salir)

Esta disciplina de cola atiende los paquetes en el orden de su llegada, por lo que ningún paquete recibe un tratamiento especial. Tiene tres «bandas», dentro de cada una se aplican las reglas FIFO. Sin embargo, no se procesará la banda 1 mientras haya paquetes esperando en la banda 0. Lo mismo se aplica para las bandas 1 y 2. Esta disciplina utiliza el campo TOS (Type Of Service del paquete IP, para más información consultar RFC 791 y RFC1349) y una máscara de asignación de prioridades (priomap) para seleccionar las bandas de prioridades.

Es la disciplina de cola activada por defecto en Linux y la que se aplica también por defecto al crear una clase, por tanto la más usada, y no puede ser configurada.

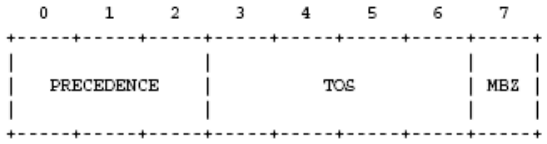
Sintaxis:

[p/b]fifo [*limit paq*|*bytes*]

limit
La longitud de la cola. Si es pfifo representa paquetes, si es bfifo representa bytes. Se obtiene de la configuración de la interfaz y puede ser visualizada y modificada únicamente con ifconfig o ip, pero no con la herramienta tc de control de tráfico. Por ejemplo: ifconfig eth0 txqueuelen 10. Esta instrucción establece la longitud de la cola a 10.

priomap
Este parámetro es configurado por defecto. Determina cómo se corresponden las prioridades de los paquetes, tal como las asigna el núcleo, a las bandas.

La correspondencia se basa en el octeto TOS del paquete, que es así:



Los cuatro bits TOS (el «campo TOS») se define como:

Binary	Decimcal	Meaning
1000	8	Minimizar retraso (md)
0100	4	Maximizar transferencia (mt)
0010	2	Maximizar fiabilidad (mr)
0001	1	Minimizar el coste monetario (mmc)
0000	0	Servicio normal

Tabla 1. Bits TOS

El número TC_PRIO.. se corresponde a bits, contando desde la derecha.

TC_PRIO..	Num	Corresponde al TOS
BESTEFFORT	0	Maximizar fiabilidad
FILLER	1	Minimizar coste
BULK	2	Maximizar transferencia (0x8)
INTERACTIVE_BULK	4	
INTERACTIVE	6	Minimizar retrasos (0x10)
CONTROL	7	

Tabla 2. TC_PRIO

Como hay 1 bit a la derecha de estos cuatro, el valor real del campo TOS es el doble del valor de sus bits. Tcpdump -v -v muestra el valor del campo TOS completo, no sólo sus cuatro bits. Este es el valor que ve en la primera columna de esta tabla:

TOS	Bits	Significa	Prioridad Linux	Banda
0x0	0	Servicio normal	0 Mejor esfuerzo	1
0x2	1	Minimizar coste monet.	1 Relleno	2
0x4	2	Maximizar fiabilidad	0 Mejor esfuerzo	1
0x6	3	mmc+mr	0 Mejor esfuerzo	1
0x8	4	Mazimizar transferencia	2 En masa	2
0xa	5	mmc+mt	2 En masa	2
0xc	6	mr+mt	2 En masa	2
0xe	7	mmc+mr+mt	2 En masa	2
0x10	8	Minimizar retrasos	6 Interactivo	0
0x12	9	mmc+md	6 Interactivo	0
0x14	10	mr+md	6 Interactivo	0
0x16	11	mmc+mr+md	6 Interactivo	0
0x18	12	mt+md	4 Int. en masa	1
0x1a	13	mmc+mt+md	4 Int. en masa	1
0x1c	14	mr+mt+md	4 Int. en masa	1
0x1e	15	mmc+mr+mt+md	4 Int. en masa	1

Tabla 3. PRIOMAP

La segunda columna contiene el valor de los cuatro bits TOS relevantes, seguidos por su significado traducido. Por ejemplo, el 15 significa que un paquete espera un Mínimo coste monetario, la Máxima fiabilidad, la Máxima transferencia y un Retraso mínimo. La cuarta columna indica la manera en que el núcleo Linux interpreta los bits del TOS, mostrando qué prioridad les asigna. La última columna indica el resultado del priomap por defecto. En la línea de órdenes, el priomap por defecto se parece a lo siguiente:

1, 2, 2, 2, 1, 2, 0, 0 , 1, 1, 1, 1, 1, 1, 1, 1

Esto significa que a la prioridad 4, por ejemplo, se asigna la banda número 1. El priomap también le permite listar prioridades mayores (> 7) que no se corresponden a asignaciones del TOS, sino que se configuran por otros medios.

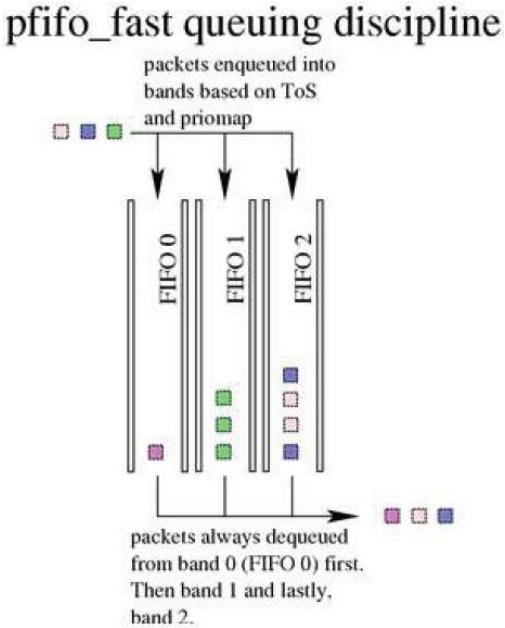


Figura 1. Disciplina de Cola PFIFO_FAST

- ***Token Bucket Filter (TBF)***

Consiste en un búfer (el bucket o balde), que se llena constantemente con piezas virtuales de información denominadas *tokens*, a una tasa específica (token rate). El parámetro más importante del *bucket* es su tamaño, que es el número de *tokens* que puede almacenar.

Esta disciplina de cola es sencilla y se limita a dejar pasar paquetes que lleguen a una tasa que no exceda una impuesta administrativamente, pero con la posibilidad de permitir ráfagas cortas que excedan esta tasa.

Cada *token* que llega toma un paquete de datos entrante de la cola de datos y se elimina del *bucket*. Asociar este algoritmo con los dos flujos (tokens y datos), presenta tres posibles situaciones:

- Los datos llegan a TBF a una tasa que es igual a la de *tokens* entrantes. En este caso, cada paquete entrante tiene su *token* correspondiente y pasa a la cola sin retrasos.

- Los datos llegan al TBF a una tasa menor a la de los *tokens*. Sólo una parte de los *tokens* se borran con la salida de cada paquete que se envía fuera de la cola, de manera que, los *tokens* se acumulan hasta llenar el *bucket*. Los *tokens* sin usar se pueden utilizar para enviar datos a velocidades mayores de la tasa de *tokens*, en cuyo caso se produce una corta ráfaga de datos.

- Los datos llegan al TBF a una tasa mayor a la de los *tokens*. Esto signigica que el *bucket* se quedará pronto sin *tokens*, lo que causará que TBF se acelere a sí mismo por un rato. Esto se llama «situación sobrelímite». Si siguen llegando paquetes, empezarán a ser descartados.

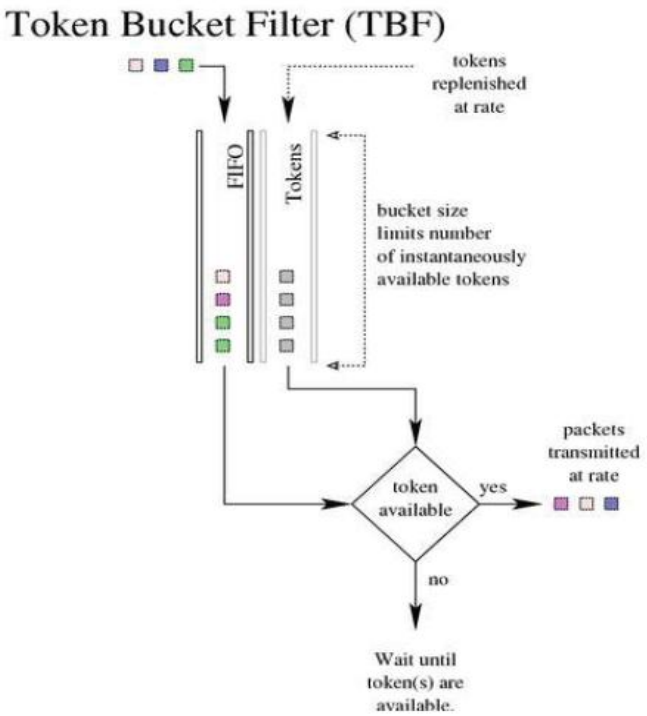


Figura 2. Disciplina de Cola Token Buket Filter

Sintaxis:

tbf limit BYTES burst BYTES[/BYTES] rate KBPS [mtuBYTES[/BYTES]] [peakrate KBPS] [latency TIME]

limit es el número de bytes a encolar antes que los paquetes sean eliminados.

burst es el tamaño del bucket en bytes.

rate permite especificar la velocidad para el scheduler.

latency indica el periodo máximo de tiempo que puede pasar un paquete en el TBF.

Ejemplo: tc qdisc add dev ppp0 root tbf rate 220kbit latency 50ms burst 1540

- ***Stochastic Fairness Queueing (SFQ)***

Este tipo de disciplina de colas, intenta distribuir el ancho de banda de un determinado interfaz de red de la forma más justa posible. Un flujo de comunicación será cualquier sesión TCP o flujo UDP, y de esta forma se consigue que ninguna comunicación impida al resto poder enviar parte de su información. Lógicamente esta disciplina de colas sólo tendrá sentido en aquellos interfaces que normalmente estén saturados y en los que no se quiere que una determinada comunicación oculte al resto.

SFQ encolará paquetes que corresponden a diferentes conexiones y tratará de permitir cada conexión para enviar la misma cantidad de paquetes, de manera que, diferentes flujos de información se ejecuten concurrentemente a la misma velocidad y no choquen entre sí. SFQ retarda paquetes e introduce retardos para tratar todas las conexiones igualmente, de manera que, hace un gran trabajo balanceando el flujo sin configuración adicional, es especialmente cómoda para conexiones P2P, pero no para tráfico interactivo (como SSH, Telnet, IPTV o Voice over IP) porque trabaja con una sola cola, lo que significa que los paquetes no son enviados inmediatamente, sino que son procesados en turnos.

El tráfico se divide en un número bastante grande de colas FIFO, una por cada conversación. Entonces se envía el tráfico de una manera parecida a Round Robin, dando a cada sesión por turnos la oportunidad de enviar datos. Esto lleva a un comportamiento bastante equitativo y evita que una única conversación ahogue a las demás. SFQ se llama «estocástica» porque realmente no crea una cola para cada sesión, sino que tiene un algoritmo que divide el tráfico en un número limitado de colas usando un algoritmo de hash.

Debido al hash, existe la posibilidad de que varias sesiones (flujo) puedan acabar en el mismo bucket, lo que dividirá por dos las posibilidades de cada sesión de enviar un paquete, de esta forma, reduciendo a la mitad la velocidad efectiva disponible. Para evitar que esta situación acabe siendo detectable, SFQ cambia a menudo su algoritmo hash, de manera que dos sesiones sólo colisionen durante unos pocos segundos.

Sintaxis:

sfq [perturb SECS] [quantum BYTES]

pertub permite especificar qué tan seguido sfq cambia su algoritmo de hashing. Si no se indica, el hash no se reconfigurará nunca, lo que no es recomendable. Un valor de 10 segundos es probablemente un buen valor.

quantum controla cuantos bytes son sacados de cada FIFO interna con algoritmo round robin. Su valor no puede ser menor al MTU.

Ejemplo: tc qdisc add dev ppp0 root sfq perturb 10

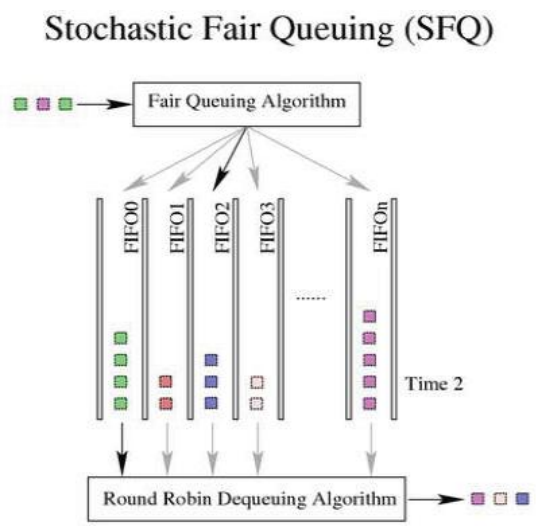


Figura 3. Disciplina de Cola Stochastic Fair Queuing (SFQ)

- ***Extended Stochastic Fairness Queueing (ESFQ)***

Conceptualmente, esta qdisc no es diferente a SFQ, sin embargo permite que el usuario controle más parámetros que en SFQ simple, ya que es una versión mejorada del SFQ original. Esta qdisc permite que el usuario controle qué algoritmo de hashing se utilizará para distribuir el acceso al ancho de banda, de esta manera, es más probable que el usuario pueda configurar una distribución de ancho de banda realmente más justa, ya que, permite hacer el reparto del ancho de banda por la dirección IP de la conexión, en vez de hacerlo por la conexión misma; esto es útil para controlar los programas P2P, que se aprovechan al crear múltiples conexiones.

Sintaxis:

esfq [perturb SECS][quantum BYTES][depth FLOWS][divisor HASHBITS][limit PKTS][hash HASHTYPE]

HASHTYPE = {classic | src | dst}
classic: algoritmo de hash igual que en sfq original
dst y src: direcciones IP origen y destino.

Ejemplo: tc qdisc add dev ppp0 root esfq perturb 10 limit 10 deph 20 hash src

Este comando, permite hacer la distribución del ancho de banda en base a la dirección IP origen de la conexión.

- ***Random Early Detection (RED)***

Esta disciplina, es utilizada para detectar la congestión. Se asegura de que la cola no se llene. Limita su tamaño de cola inteligentemente. Las colas regulares, cuando están llenas, simplemente descartan paquetes al final de la cola, lo cual puede no ser el comportamiento más óptimo. RED también realiza un descarte al final, pero lo hace de una manera más gradual.

Cuando la cola llega a un largo promedio, los paquetes encolados tienen un chance configurable de ser marcados (lo cual puede terminar en descarte). Este chance aumenta linealmente, hasta un punto llamado el máximo promedio del largo de la cola, sin embargo la cola puede ser más grande. Esto tiene muchos beneficios sobre el simple descarte del final, y consume poco procesador.

Previene las retransmisiones sincronizadas después de una ráfaga de tráfico, lo cual causa más retransmisiones. El objetivo es tener una cola de tamaño pequeño, lo cual es bueno para la interactividad y además no molesta al tráfico TCP con demasiados descartes después de una ráfaga de tráfico.

Sintaxis:

red limit BYTES min BYTES max BYTES avpkt BYTES burst PACKETS probability PROBABILITY bandwidth KBPS [ecn]

limit Límite estricto sobre el tamaño real de la cola (no el promedio) en bytes. Se recomienda que sea mayor que max. Los paquetes por encima de este valor son descartados.

min Tamaño promedio de la cola, al cual el marcado se vuelve una posibilidad.

max Tamaño promedio de la cola, al cual la probabilidad de marcado es máxima. Debería ser por lo menos dos veces el valor de *min* para prevenir retransmisiones sincronizadas.

avpkt Especificado en bytes. Se utiliza con burst para determinar la constante de tiempo para los cálculos del tamaño promedio de la cola. 1000 es un buen valor.

burst determina que tan rápido es influenciado el tamaño promedio de la cola por su tamaño real. Se sugiere la siguiente regla orientativa: (min+min+max)(3*avpkt).

probability Máxima probabilidad de marcado, especificada como un número en punto flotante desde 0.0 a 1.0. Valores sugeridos son 0.01 ó 0.02 (1% ó 2%, respectivamente).

bandwidth se utiliza para calcular el tamaño promedio de la cola después de un período de inactividad. Debería configurarse igual al valor de ancho de banda de la interfaz. Esto no significa que RED hará shaping.

Ecn - Explicit Congestion Notification permite que RED notifique a hosts remotos que su tráfico excede el ancho de banda disponible. Los hosts que no soportan ECN solo pueden ser notificados descartando sus paquetes.

Ejemplo: tc qdisc add dev ppp0 root RED limit 4096kbit min 512bit max 1024kbit avpkt 1000 burst 100 probability 0.02 bandwidth 1024kbit

2.2. Disciplinas de Cola con Clases

[Ref. a y k]

Las disciplinas de cola con clases (classfull qdisc) son muy útiles cuando se tienen diferentes tipos de tráfico a los que se quiere dar un tratamiento separado. Los filtros asociados a la qdisc devuelven una decisión, y la qdisc la usa para encolar el paquete en una de las clases. Cada subclase puede probar otros filtros, para determinar si se imparten más instrucciones, en caso contrario, la clase encola el paquete en la qdisc que contiene. La mayoría de las qdiscs con clases, también realizan «shaping», esto es útil, tanto para reordenar paquetes como para controlar tasas, por ejemplo, una interfaz de gran velocidad (ethernet) enviando a un dispositivo más lento (un cable módem).

Cada interfaz tiene una “qdisc raíz” de salida, que por defecto, es la disciplina de colas pfifo_fast sin clases, mencionada anteriormente. A cada qdisc y a cada clase, se le asigna un controlador (handle), que se puede usar en posteriores sentencias de

configuración para referirse a la qdisc. Aparte de la qdisc de salida, la interfaz también puede tener una de entrada, que dicta las normas sobre el tráfico que entra. Los paquetes se encolan y desencolan en el qdisc raíz, que es lo único con lo que habla el núcleo o kernel del sistema operativo.

La jerarquía típica sería la siguiente:

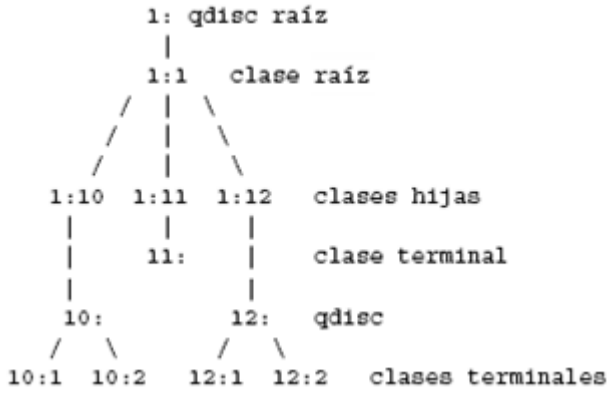


Figura 4. Jerarquía de clases y colas

Cuando el núcleo decide que necesita extraer paquetes para enviarlos a la interfaz, la qdisc raíz 1:, recibe una petición de desencolar, que se pasa a 1:1, que a su vez la pasa a :10, :11, y :12, cada una de las cuales consulta a sus descendientes, e intenta hacer dequeue() sobre ellos. Las clases anidadas solamente hablan a sus qdisc paternas, y nunca a una interfaz. Sólo la qdisc raíz recibe peticiones de desencolado del núcleo. En consecuencia, las clases nunca desencolan más rápido de lo que sus padres permiten. De esta manera, se puede tener, por ejemplo, SFQ como clase interna, que no hace ajustes, sólo reordena, y una qdisc externa, que es la que hace los ajustes.

Una clase siempre tiene asociada una cola, una cola puede tener asociadas de 0 a n clases y una clase puede tener de 0 a n clases hijas.

- ***Qdisc PRIO***

La qdisc PRIO es un reorganizador conservativo. Funciona como una pfifo_fast reforzada, en la que cada banda es una clase separada, en lugar de una simple FIFO.

Cuando se encola un paquete a la qdisc PRIO, se escoge una clase basándose en las órdenes de filtrado. Por defecto, se crean tres clases y estas clases por defecto contienen qdiscs que son puras FIFO sin estructura interna y hace la clasificación del tráfico en base a priomap y filtros.

Cuando se necesite desencolar un paquete, se intenta primero con la clase :1. Las clases más altas sólo se usan si no se ha conseguido el paquete en las clases más bajas.

PRIO definitivamente dará una mejor latencia para el tráfico de alta prioridad, desde el momento que la qdisc es diseñada para el propósito de prioridad. Esta qdisc es muy útil en caso de que se quiera dar prioridad a cierto tráfico sin usar sólo las marcas TOS, sino usando el potencial de los filtros de tc (Ver *Herramientas de Control de Tráfico para Gestión de Ancho de Banda en Linux,* en este mismo capítulo). También puede contener cualquier qdisc, mientras que pfifo_fast está limitada a qdisc de fifo sencillas. Esta cola no hace ajustes (shaping), entonces se le aplica el mismo aviso que a SFQ: usarla solamente si el enlace físico está realmente lleno o ponerla dentro de una qdisc con clases que haga ajustes. Esto último se aplica a la mayoría de dispositivos DSL y cable módems.

Sintaxis:

PRIO [bands num][priomap band band band…]

bands
Número de bandas a crear. Cada banda es una clase. Si cambia este número, también deberá cambiar:

priomap
Las bandas son clases, y todas se llaman de mayor:1 a mayor:3 por defecto, de manera que si la qdisc PRIO se llama 12:, tc filtrará el tráfico a 12:1 para garantizar la mayor prioridad, ya que la banda 0 va al número menor 1, La banda 1 al número menor 2, etc.

Si no se proporciona filtros de tc para clasificar el tráfico, la qdisc PRIO examina la prioridad TC_PRIO para decidir cómo encolar el tráfico. Este proceso funciona igual que con la qdisc pfifo_fast mencionada anteriormente, ver esta cola para mayor detalle.

Ejemplo: tc qdisc add dev eth0 root handle 1: prio

Crea instantáneamente las clases 1:1, 1:2, 1:3

- ***DSMARK***

[Ref. h]

La disciplina de cola DSMARK fue diseñada en especial para cumplir completamente con las especificaciones de DiffServ. Se encarga del marcado de paquetes dentro de la implementación de DiffServ en Linux. Su comportamiento es realmente muy simple cuando se compara con otras disciplinas de cola del control de Tráfico en Linux. Contrariamente a otras disciplinas de cola, DSMARK no hace shaping, control o policing de tráfico. Tampoco prioriza, retarda, reordena o descarta paquetes, solo marca paquetes usando el campo **DS**.

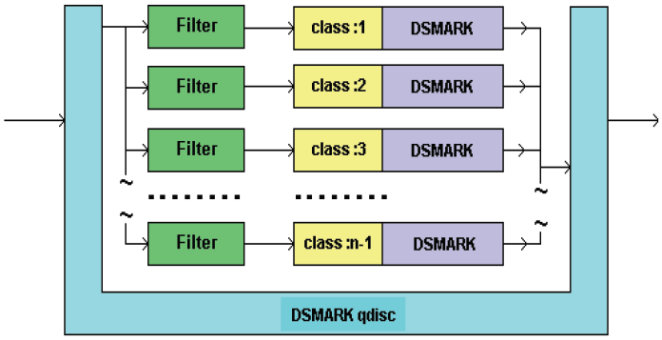


Figura 5. Disciplina de Cola DSMARK

Las clases son numeradas 1,2,3,4,...., n-1, donde *n* es un parámetro que define el tamaño de una tabla interna necesaria para implementar el comportamiento de la disciplina de cola, este parámetro se denomina *índices*. El elemento 0 es la cola principal misma, los elementos a partir de 1 a n-1 son las clases de la disciplina de cola. Cada una de esas clases, pueden ser seleccionadas utilizando un filtro conectado a la disciplina de cola, los paquetes que están siendo seleccionados por el filtro, son puestos en su clase DSMARK respectiva.

Los paquetes son marcados justo antes de dejar la disciplina de cola para ser colocados en la interfaz de salida. Son marcados en el campo DS (Differentiated Services Field, para más información consultar [RFC2474]) con un valor entero que se define para cada clase cuando se configura la disciplina de cola.

Sintaxis:
dsmark indices <id> [default_index<did>] [set_tc_index]

indices define el tamaño de una tabla interna necesaria para implementar el comportamiento de la disciplina de cola. El valor máximo es 2^n donde n>0.

default_index índice por defecto. Los paquetes que no corresponden a ninguna de las clases existentes, serán ubicados en la clase por defecto definida por este índice.

set_tc_index instruye a la disciplina de cola para recuperar el campo DS y almacenarlo en la variable skb-tc_index.

Ejemplo: tc qdisc add dev eth0 root handle 1:0 dsmark indices 32 default_index 7 set_tc_index

- ***Class Based Queueing (CBQ)***

CBQ es la qdisc más compleja disponible, la más publicitada, la menos comprendida, y probablemente la más difícil de configurar correctamente. Esto se debe a que el algoritmo CBQ no es tan preciso y realmente no se ajusta del todo a la manera de trabajar de Linux.

CBQ, además de ser una qdisc de clases, también es ajustadora (shaper) y es en este aspecto en el que no funciona del todo bien. Trabaja asegurándose de que el enlace está ocioso sólo lo necesario para reducir el ancho de banda real hasta la tasa configurada. Para hacerlo, realiza cálculos del tiempo que debería pasar entre los paquetes medios. Por ejemplo, si se intenta ajustar a 1mbit/s una conexión de 10mbit/s, el enlace debería estar ocioso el 90% del tiempo, si no lo está, se necesita acelerar de manera que realmente lo esté. Esto es bastante difícil de medir, de manera que en su lugar, CBQ deriva el tiempo ocioso del número de microsegundos que se tarda entre cada petición de más datos por parte de la capa de hardware. Combinados, se pueden usar para aproximar qué tan lleno o vacío está el enlace. Esto es bastante tortuoso y no siempre lleva a resultados adecuados. Por ejemplo, ¿qué pasaría si la verdadera velocidad del enlace no es capaz de transmitir realmente todos los 10mbit/s de datos, quizá debido a un driver mal implementado? Una tarjeta de red PCMCIA tampoco alcanzará nunca los 10mbit/s debido a la manera en que está diseñado el bus. Se vuelve aún peor si se considera dispositivos de red no del todo reales, como PPP sobre Ethernet o PPTP sobre TCP/IP. El ancho de banda efectivo en ese caso,

probablemente se determina por la eficiencia de los canales al espacio de usuario (que es enorme).

Por los motivos que se mencionan, surge HTB que es una disciplina de cola conocida como el reemplazo de la cola CBQ en Linux, más entendible y más intuitiva.

- ***Hierarchical Token Bucket (HTB)***

[Ref. i, j, k]

HTB surge como una disciplina más entendible, intuitiva y más rápida para reemplazar a la qdisc CBQ en Linux.

HTB permite controlar el uso del ancho de banda saliente sobre un enlace dado. Permite que se utilice un sólo enlace físico para simular varios enlaces más pequeños y para enviar diferentes tipos de tráfico sobre diferentes enlaces simulados. Se necesita especificar cómo dividir el enlace físico en enlaces simulados y como decidir qué enlace simulado tiene que utilizar un paquete dado para ser enviado. Permite dividir el ancho de banda disponible entre una tasa mínima y una tasa máxima. El algoritmo asegura la disponibilidad del mínimo, y si se permite, alcanzar el máximo a través del mecanismo de “préstamos” de ancho de banda sobrante. Este mecanismo se explica más adelante.

HTB está basado en clases, por eso tiene la habilidad de crear una estructura de clases que puede dividir el ancho de banda en diferentes clases. Está elaborado para garantizar ancho de banda y no interactividad, por tanto, no cuenta paquetes, sino bytes.

Dentro de una instancia HTB, pueden existir muchas clases, cada una de estas clases contienen una qdisc, por defecto pfifo_fast. Cuando encola un paquete, HTB comienza en la raíz hacia los nodos hijos y usa varios métodos para determinar qué clase debería recibir los datos. Si la clase encontrada es un nodo hoja, se encola allí el paquete, si no, se busca un nodo hoja empezando desde ese nodo.

HTB usa los conceptos de *Tokens* y *Bukets*, que corresponden al sistema basado en clases, y filtros para permitir un control complejo sobre el tráfico. Con un modelo complejo de "préstamos", HTB puede realizar una variedad de técnicas de control de tráfico sofisticadas. Una de las aplicaciones más comunes de HTB involucra el shaping, por la transmisión del tráfico a una tasa específica.

Shaping

El shaping ocurre en las clases hojas, no en las clases internas ni en la raíz, ya que solo existe para sugerir cómo el mecanismo de "préstamos" debería distribuir los *tokens* disponibles.

El mecanismo de préstamos o Borrowing

Si una clase tiene una tasa de transferencia máxima mayor que la mínima garantizada, está habilitada para pedir ancho de banda "prestado" de otras clases. La clase hija hace la solicitud a su padre una vez que ya ha usado todo su ancho de banda garantizado y continuará intentando hacer préstamos hasta alcanzar el ancho de banda máximo alcanzable. La clase padre concede el ancho de banda solicitado siempre que haya ancho de banda excedente. Así que, el préstamo fluye hacia las clases hojas y el cobro del uso de ancho de banda, fluye hacia la raíz. Ver Figura 6. Disciplina de Cola HTB.

Sintaxis:

qdisc ….dev dev **(parent** classid **| root) [handle** major:**] htb [default** minor-id**]**

class … dev dev **parent** major:[minor] [**classid** major:minor] **htb rate** rate [**ceil** rate] **burst** bytes [**cburst** bytes][**prio** priority]

Para Qdisc:

Parent major:minor | *root* determina la ubicación de una instancia HTB, ya sea como la raíz de una interfaz de red o dentro de una clase existente.

Handle major: identifica a la qdisc. Debe consistir únicamente de un número **major** seguido de dos puntos.

Default minor-id indica que el tráfico no clasificado será enviado por defecto a la clase identificada *minor-id.* Este parámetro es opcional y su valor por defecto es 0, lo que provocaría que el tráfico no clasificado sea desencolado a la velocidad del hardware sin pasar por ninguna de las clases unidas a la qdisc raíz, y por tanto no habría un límite para la tasa de transferencia.

Para Class:

Parent major:minor indica la ubicación de la clase dentro de la jerarquía.

Classid major:minor identifica a la clase. El número major debe ser igual al número major de la qdisc a la cual corresponde.

Rate rate tasa de transferencia mínima o garantizada para la clase.

Ceil rate tasa de transferencia máxima, a la cual la clase puede enviar información, si su padre tiene ancho de banda disponible. Su valor por defecto es el valor del *rate*.

Burst bytes cantidad de bytes que pueden ser enviados a velocidad del *ceil*, en exceso del *rate* configurado. Debería ser siempre al menos igual o mayor que el *burst* de cualquiera de sus hijos.

Cburst bytes cantidad de bytes que pueden ser enviados a velocidad infinita, o sea, a la velocidad que lo permita la interfaz de red. Debería ser siempre al menos igual o mayor que el *cburst* de cualquiera de sus hijos.

Prio es un parámetro empleado en el mecanismo de préstamos. Determina el orden en que las clases serán atendidas cuando solicitan a su padre más ancho de banda, después que han alcanzado su *rate* configurado. La clase con el prio más bajo será de mayor prioridad y la clase con el prio más alto de menor prioridad y solo pueden pedir prestado si aún queda ancho de banda excedente. El objetivo es ofrecer ancho de banda excedente por prioridades. El valor por defecto es 0 que es la mayor prioridad.

R2q significa rate to quantum y es empleado para calcular el *quantum,* usando el *rate* especificado. Su valor por defecto es 10 y puede ser especificado al crear una qdisc HTB raíz.

Quantum es un parámetro clave usado por HTB para el control de préstamos. Es la cantidad de bytes que una clase puede recibir prestados para transmitir por encima de su *rate,* pero debajo de su *ceil* configurado. Debe ser mayor a 1500 (MTU) y menor a 60000. Es calculado por HTB (y no por el usuario) de la siguiente manera: rate (en bytes) / r2q. Sin embargo, puede ser sobre escrito al agregar una clase HTB, esto es útil para evitar los warnings en los logs cuando el valor calculado por HTB esté fuera del rango permitido, lo que puede ocurrir con valores de *rate* muy altos. Cuando se especifica el quantum en la línea de comandos, el r2q es ignorado para esa clase.

Los parámetros *burst* y *cburst* controlan la cantidad de datos que pueden ser enviados a máxima velocidad (la velocidad del hardware) sin intentar dar servicio a otra clase. Para un procesador Intel, hay 100 timers events por segundo, en este caso se puede realizar el cálculo de burst y cburst de la siguiente manera:

El *burst* mínimo se calcula como burst = rate/8/100. Rate en kbit.

El *cburst* mínimo se calcula como cburst = ceil/8/100. Ceil en kbit.

Se requiere que los valores de estos parámetros sean mínimos, si no se especifica un valor para ellos, HTB se encargará de calcular el tamaño más pequeño que se pueda. Como el cálculo depende del hardware, según muestra la fórmula anterior, es recomendable que sea realizado automáticamente.

HTB establece las siguientes reglas o restricciones a sus tasas de transferencia (rate / ceil), lo que conduce a realización de shaping para su cumplimiento:

a. El *Rate* de una clase debe ser menor o igual a su *Ceil*.
b. La suma de los *Rate* de las clases hijas debe ser menor o igual al *Ceil* de la clase padre.
c. La suma de los *Rate* de las clases hijas debe ser menor o igual al *Rate* de la clase padre.
d. El *Rate* de la clase raíz es igual a su *Ceil*.

Nota: La clase raíz se refiere a la que está unida al qdisc raíz y de la que dependen todas las clases del árbol.

La siguiente tabla resume las acciones que realiza HTB en base a su configuración:

Tipo de Clase	**Estado de Tasas**	**Estado Interno de HTB**	**Acción Tomada por HTB**
Hoja	< rate	HTB_CAN_SEND	La clase hoja desencolará bytes encolados.
Hoja	> rate, < ceil	HTB_MAY_BORROW	La clase hoja intentará pedir ancho de banda prestado a su padre. Si hay ancho de banda disponible, será dado en proporciones del *quantum* de la clase y la clase hoja desencolará a *cburst* bytes.
Hoja	> ceil	HTB_CANT_SEND	No serán desencolados paquetes. Esto causará retardos e incrementará la latencia para encontrar el *rate* deseado.
interna, raíz	< rate	HTB_CAN_SEND	Las clases internas darán ancho de banda prestado a sus hijos.
interna, raíz	> rate, < ceil	HTB_MAY_BORROW	La clase interna intentará pedir prestado a su padre. Existe una competencia entre las clases hijas en proporción de sus *quantum* por solicitud.
interna, raíz	> ceil	HTB_CANT_SEND	La clase interna no intentará pedir prestado de su padre y no se dará prestado a las clases hijas.

Tabla 4. Acciones de HTB en base a su configuración

Ejemplo: tc qdisc add dev eth0 root handle 1: htb default 3 r2q 15

tc class add dev eth0 parent 1: classid 1:1 htb rate 128kbit ceil 128kbit burst 2k quantum 5000

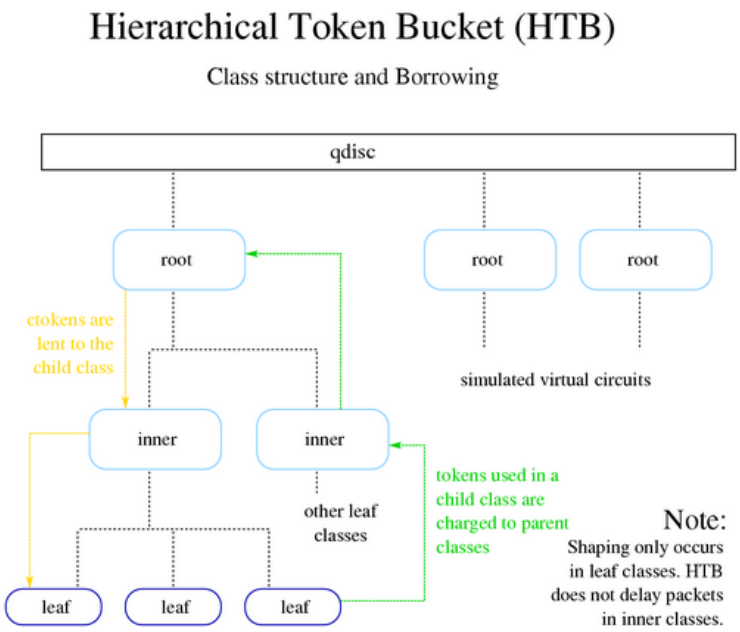


Figura 6. Disciplina de Cola HTB

2.3. Consejos sobre qué Disciplina de Cola usar en determinado momento

[Ref. a]

- Para simplemente ralentizar el tráfico de salida, usar el Token Bucket Filter. Esta disciplina de cola funciona para grandes anchos de banda.

- Si el enlace está realmente lleno y se necesita asegurarse de que ninguna sesión domine el ancho de banda de salida, usar Stochastical Fairness Queueing.

- Para garantizar ancho de banda por servicios, durante su ejecución, usar la disciplina de cola Hierarchical Token Bucket.

- Si se tiene un backbone grande y se sabe lo que está haciendo, considere usar Random Early Drop.

- Si no se quiere ajustar, sino determinar si la interfaz está tan cargada, usar la cola pfifo (no pfifo_fast). Carece de bandas internas pero lleva la cuenta del tamaño de su búfer.

3. Técnica de Encolado

[Ref. g]

Consiste en marcar el tráfico con un identificador único (generalmente cuando pasa a través del firewall), luego, se crea una jerarquía de clases, las cuales tienen una cola asociada que saca el tráfico a la red (Ver *Disciplinas de Cola* en este mismo capítulo) y finalmente, se introduce el tráfico en las clases según sus marcas, a través del uso de filtros. De este modo, el tráfico es clasificado para recibir un tratamiento diferenciado según su tipo. Ver figura 7. *Esquema de la técnica de encolado.*

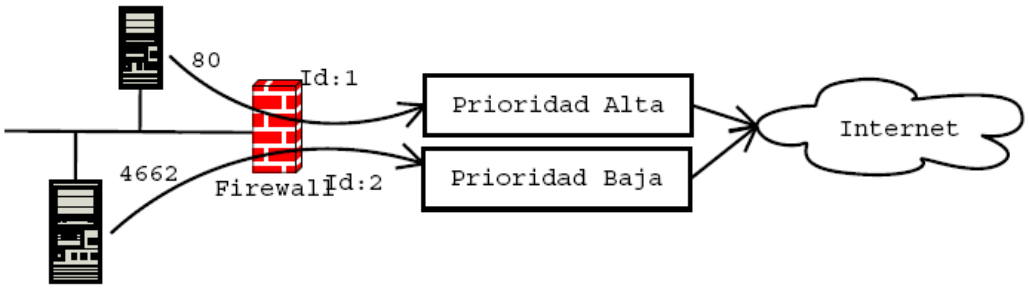


Figura 7. Esquema de la técnica de encolado.

4. Herramientas de Control de Tráfico para Gestión de Ancho de Banda en Linux

4.1. Iptables

[Ref. a y r]

Iptables, es una herramienta de cortafuegos, construida sobre Netfilter, que permite filtrar paquetes, realizar traducción de direcciones de red (NAT) para IPv4 o mantener registros de log. Es una herramienta de espacio de usuario, ejecutada desde la consola de Linux, mediante la cual el administrador puede definir políticas (o reglas) de filtrado del tráfico que circula por la red y está disponible en la mayoría de las distribuciones actuales de Linux, licenciado bajo la licencia GPL (GNU General Public License). Este proyecto fue incorporado al núcleo Linux 2.3 y ha sustituido a la herramienta ipchains de Linux 2.2 con la introducción de su framework que hace de iptables una herramienta más sencilla y flexible.

Marcado de paquetes

[Ref. d y m]

Iptables, realiza la identificación y determinación del contexto de los paquetes (flujos) con marcas de firewall. La marca es un campo especial que solo existe durante el viaje del paquete dentro del kernel de Linux (no confundir con marcas que viajan en algún campo de la cabecera IP) y consiste de un valor entero de 32 bits.

El objetivo del marcado, es identificar al paquete, para luego tomar decisiones con el mismo, de acuerdo a su marca y poderlo tratar diferente al resto de paquetes. Cuando iptables detecta el paquete (por su información de configuración), le establece una marca de firewall, que posteriormente es asociada a una clase de tráfico, quedando así el paquete mismo asociado a dicha clase.

Esquemas de filtrado

[Ref. n, s y t]

Tablas y Cadenas

Iptables tiene tres tablas incorporadas: FILTER, NAT y MANGLE, cada una de las cuales contiene cadenas predefinidas INPUT, OUTPUT y FORWARD, que no se pueden eliminar. Es posible crear nuevas tablas mediante módulos de extensión y crear / eliminar cadenas definidas por usuarios dentro de cualquier tabla. Inicialmente, todas las cadenas están vacías y tienen una política de destino que permite que todos los paquetes pasen sin ser bloqueados o alterados.

Tabla filter (o Tabla de filtros - iptables actúa como cortafuegos) es la responsable del filtrado de paquetes (es decir, de bloquear o permitir que un paquete continúe su camino). Este es el lugar donde se toma acción contra los paquetes para aceptarlos o rechazarlos (DROP / ACCEPT) dependiendo de su contenido. Todos los paquetes pasan a través de la tabla de filtros. Contiene las siguientes cadenas predefinidas y cualquier paquete pasará por una de ellas:

Cadena INPUT (Cadena de ENTRADA). Todos los paquetes destinados al propio ordenador, atraviesan esta cadena (y por esto se le llama algunas veces LOCAL_INPUT o ENTRADA_LOCAL).

Cadena OUTPUT (Cadena de SALIDA). Todos los paquetes creados por este ordenador, atraviesan esta cadena (a la que también se le conoce como LOCAL_OUTPUT o SALIDA_LOCAL) antes de enviarlos a la red.

Cadena FORWARD (Cadena de REDIRECCIÓN). Todos los paquetes que meramente pasan por este ordenador (para ser ruteados), o sea que van dirigidos a otro ordenador, recorren esta cadena.

Tabla NAT (o Tabla de traducción de direcciones de red - iptables actúa como router) es la responsable de configurar las reglas de reescritura de direcciones o de puertos de los paquetes. Únicamente debería ser usada para NAT en los paquetes. Solo el primer paquete en cualquier conexión pasa a través de esta tabla, después el resto de paquetes tomarán la misma acción que el primer paquete. Contiene las siguientes cadenas predefinidas:

Cadena PREROUTING (Cadena de PRERUTEO). Los paquetes entrantes pasan a través de esta cadena antes de que se consulte la tabla de ruteo local, principalmente para DNAT (destination-NAT o traducción de direcciones de red de destino).

Cadena POSTROUTING (Cadena de POSRUTEO). Los paquetes salientes pasan por esta cadena después de haberse tomado la decisión del ruteo, principalmente para SNAT (source-NAT o traducción de direcciones de red de origen).

Cadena OUTPUT (Cadena de SALIDA). Permite hacer un DNAT limitado en paquetes generados localmente.

Tabla mangle (o Tabla de destrozo - altera paquetes especiales) es la responsable de ajustar las opciones de los paquetes, por ejemplo la calidad de servicio. Todos los paquetes pasan por esta tabla. Debido a que está diseñada para efectos avanzados, contiene todas las cadenas predefinidas posibles:

Cadena PREROUTING (Cadena de PRERUTEO). Todos los paquetes que logran entrar a este ordenador, antes de que el ruteo decida si el paquete debe ser reenviado (cadena de REENVÍO) o si tiene destino local (cadena de ENTRADA).

Cadena INPUT (Cadena de ENTRADA). Todos los paquetes destinados para este ordenador, pasan a través de esta cadena.

Cadena FORWARD (Cadena de REDIRECCIÓN). Todos los paquetes que son dirigidos a otro ordenador, pasan a través de esta cadena.

Cadena OUTPUT (Cadena de SALIDA). Todos los paquetes creados por este ordenador, pasan a través de esta cadena antes de enviarlos a la red.

Cadena POSTROUTING (Cadena de POSRUTEO). Todos los paquetes que abandonan este ordenador, pasan a través de esta cadena.

Los esquemas de filtrado son resumidos en la figura 8. *Esquemas de filtrado*.

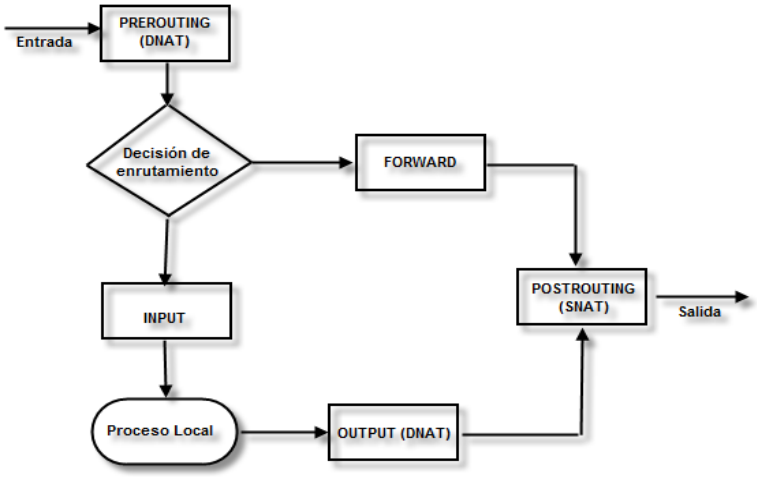


Figura 8. Esquemas de filtrado

Iptables permite definir reglas acerca de qué hacer con los paquetes de red. Cada cadena contiene una lista de reglas. Las cadenas se agrupan en tablas y una tabla está asociada con un tipo diferente de procesamiento de paquetes.

Cuando un paquete se envía a una cadena, lo compara en orden, contra cada regla en la cadena. La regla especifica qué propiedades debe tener el paquete para aceptarlo, como número de puerto o dirección IP. Si el paquete no corresponde con la regla, el procesamiento continúa con la regla siguiente. Si la regla, por el contrario, corresponde con el paquete, las instrucciones de destino de las reglas se siguen (y cualquier otro procesamiento de la cadena normalmente se aborta). Si el paquete alcanza el fin de una cadena predefinida sin haberse correspondido con ninguna regla de la cadena, la política de destino de la cadena dicta qué hacer con el paquete. Si el paquete alcanza el fin de una cadena definida por el usuario sin haber cumplido ninguna regla de la cadena o si la cadena definida por el usuario está vacía, el recorrido continúa en la cadena que hizo la llamada (lo que se denomina implicit target RETURN o RETORNO de destino implícito). Solo las cadenas predefinidas tienen políticas.

Algunas propiedades de los paquetes solo pueden examinarse en ciertas cadenas (por ejemplo, la interfaz de red de salida no es válida en la cadena de ENTRADA). Algunos destinos solo pueden usarse en ciertas cadenas y/o en ciertas tablas (por ejemplo, el destino SNAT solo puede usarse en la cadena de POSRUTEO de la tabla de traducción de direcciones de red). Estas consideraciones serán explicadas en detalle más adelante.

El comando iptables

[Ref. m y n]

Iptables requiere privilegios elevados para operar, por lo que el único que puede ejecutarlo es el superusuario.

Sintaxis:

Los elementos entre llaves, {...|...|...}, son requeridos, pero solo se puede indicar uno de ellos separados por '|', y los que están entre corchetes, [...], son opcionales.

iptables [-t table] {-A | --append | -D | --delete} cadena rule-specification [options]
iptables [-t table] {-R | --replace | -I | --insert} cadena [rulenum] rule-specification [options]
iptables [-t table] {-D | --delete} cadena rulenum [options]
iptables [-t table] {-N | --new-chain} chain
iptables [-t table] {-X | --delete-chain} [chain]
iptables [-t table] {-P | --policy} chain target [options]
iptables [-t table] {-E | --rename-chain} old-chain-name new-chain-name

Opciones comunes
[Ref. m y n]
-t tabla Hace que el comando se aplique a la tabla especificada. Si esta opción se omite, el comando se aplica a la tabla filter por defecto.
-v Produce una salida con detalles.
-n Produce una salida numérica (números de puerto en lugar de nombres de servicio y direcciones IP en lugar de nombres de dominio).
--line-numbers Cuando se listan reglas, agrega números de línea al comienzo de cada regla, correspondientes a la posición de esa regla en su cadena.

Operaciones para manejar cadenas en general
[Ref. m y n]
-N Crear una nueva cadena
-X Eliminar una cadena vacía
-P Cambiar la política de una cadena predefinida
-L Mostrar las cadenas
-F Vaciar las reglas de una cadena
-Z Poner a cero los contadores de paquetes y bytes en todas las reglas de una cadena

Manipular las reglas dentro de una cadena
[Ref. m y n]
-A Añadir una nueva regla a una cadena
-I Insertar una nueva regla en la misma posición de una cadena
-R Reemplazar una regla en la misma posición de una cadena
-D Eliminar una regla de una cadena

Destinos de reglas

[Ref. m y n]

El destino de una regla puede ser el nombre de una cadena definida por el usuario o uno de los destinos ya incorporados.

Cuando un destino es el nombre de una cadena definida por el usuario, al paquete se dirige a esa cadena para ser procesado (como ocurre con una llamada a un módulo en un lenguaje de programación). Si el paquete consigue atravesar la cadena definida por el usuario sin que ninguna de las reglas de esa cadena actúe sobre él, el procesamiento del paquete continúa donde había quedado en la cadena actual. Estos llamados entre cadenas se pueden anidar hasta cualquier nivel deseado.

Existen los siguientes destinos ya incorporados:

ACCEPT (aceptar)
Este destino hace que netfilter acepte el paquete. Esto depende de la cadena que realiza la aceptación: un paquete aceptado en la cadena de ENTRADA, se le permite ser recibido por la terminal (host); un paquete aceptado en la cadena de SALIDA, se le permite abandonar la terminal y un paquete aceptado en la cadena de REDIRECCIÓN se le permite ser ruteado a través de la terminal.

DROP (descartar)
Este destino hace que netfilter descarte el paquete sin ningún otro tipo de procesamiento. El paquete simplemente desaparece sin indicación de que fue descartado, al ser entregado a la terminal de envío o a una aplicación. Esto, a menudo, se le refleja al remitente como un communication timeout (alcance del máximo tiempo de espera en la comunicación), lo que puede causar confusión (aunque el descarte de paquetes entrantes no deseados se considera a veces una buena política de seguridad, pues no da ni siquiera el indicio a un posible atacante de que la terminal existe).

QUEUE (encolar)
Este destino hace que el paquete sea enviado a una cola en el espacio de usuario. Si no hay ninguna aplicación que lea la cola, este destino es equivalente a DROP.

RETURN (retorno)
Hace que el paquete deje de circular por la cadena en cuya regla se ejecutó el destino RETURN. Si dicha cadena es una subcadena de otra, el paquete continuará por la cadena superior como si nada hubiera pasado. Si por el contrario, la cadena es una cadena principal (por ejemplo la cadena INPUT), al paquete se le aplicará la política por defecto de la cadena en cuestión (ACCEPT, DROP o similar).

Hay muchos destinos de extensión disponibles ya incorporados. Algunos de los más comunes son:

REJECT (rechazo)

Este destino tiene el mismo efecto que 'DROP', salvo que envía un mensaje de error “Host Unreachable” a quien envió originalmente. Se usa principalmente en las cadenas de ENTRADA y de REDIRECCIÓN de la tabla FILTER.

LOG (bitácora)

Este destino lleva un log o bitácora del paquete. Puede usarse en cualquier cadena y cualquier tabla, y muchas veces se usa para depurar (análisis de fallos, como la verificación de qué paquetes están siendo descartados).

ULOG

Este destino lleva un log o bitácora del paquete, pero no de la misma manera que el destino LOG. El destino LOG le envía información al log del núcleo, pero ULOG hace multidifusión de los paquetes que cumplen esta regla a través de un socket netlink, de manera que programas del espacio de usuario puedan recibir este paquete conectándose al socket.

DNAT

Este destino hace que la dirección (y opcionalmente el puerto) de destino del paquete sean reescritos para traducción de dirección de red. Mediante la opción '--to-destination' debe indicarse el destino a usar. Esto es válido solamente en las cadenas de SALIDA y PRERUTEO dentro de la tabla de NAT.

SNAT

Este destino hace que la dirección (y opcionalmente el puerto) de origen del paquete sean reescritos para traducción de dirección de red. Mediante la opción '--to-source' debe indicarse el origen a usar. Esto es válido solamente en la cadena de POSRUTEO dentro de la tabla de NAT y, como DNAT, se recuerda para todos los paquetes que pertenecen a la misma conexión.

MASQUERADE

Esta es una forma especial, restringida de SNAT para direcciones IP dinámicas, como las que proveen la mayoría de los proveedores de servicios de Internet (ISPs) para modems o línea de abonado digital (DSL). En vez de cambiar la regla de SNAT cada vez que la dirección IP cambia, se calcula la dirección IP de origen a la cual hacer NAT, fijándose en la dirección IP de la interfaz de salida, cuando un paquete cumple esta regla. Si la dirección de la interfaz cambia (por ejemplo, por reconectarse al ISP), todas las conexiones que hacen NAT a la dirección vieja se olvidan.

MARK

Permite seleccionar paquetes en función a marcas previamente realizadas dentro de la misma computadora. Solo es válido en la tabla MANGLE. Las marcas son usadas para poder hacer un ruteo más avanzado y organizado. La marca solo es mantenida dentro del host y no se debe esperar que la misma permanezca en un paquete enviado a otro host, ya que es una asociación que se hace dentro del kernel.

TOS (Type Of Service)

Se usa dentro de la tabla MANGLE para establecer el valor TOS. Permite seleccionar paquetes en función al campo TOS de la cabecera IP. Se puede usar conjuntamente con iproute2 y las marcas previamente descritas para hacer un ruteo avanzado, por ejemplo, se podría marcar paquetes con un determinado TOS y luego con iproute2 enrutarlos por una interfaz más veloz. Los type of services pueden escribirse como números en hexadecimal o escribirse como nombres sacados de iptables -m tos -h.

TTL

Se usa dentro de la tabla MANGLE para especificar el valor TTL en los paquetes. Permite seleccionar paquetes de un determinado tiempo de vida. Esto se usa en muchos casos para depurar el funcionamiento de una LAN.

Especificaciones de las reglas

[Ref. m y n]

Las siguientes opciones, se usan (frecuentemente combinadas unas con otras) para crear especificaciones de reglas.

-A, --append
Agrega una regla a una cadena especificada.
Ejemplo: iptables -A INPUT ...

-s, --src, --sport, --source
Permite seleccionar paquetes en función de la dirección IP origen. Es posible especificar un host o un rango de IPs agregando la máscara de subred (por ejemplo: 192.168.0.0/24).
Ejemplo: iptables -A INPUT -s 192.168.1.1

Para seleccionar todos los paquetes que no vengan de un determinado host o red, se usa el símbolo "!" como negación.
Ejemplo: iptables -A INPUT -s ! 192.168.1.0/24

-d, --dst, --dport, --destination
IDEM anterior pero con dirección o rango de direcciones de destino.
Ejemplo: iptables -A INPUT -d 192.168.1.1

-j destino, --jump destino
Especifica el destino de una regla. El destino es el nombre de una cadena destino (definida por el usuario o incorporada) y especifica qué hacer con paquetes que cumplen la regla. Si esta opción es omitida en una regla, entonces el cumplimiento de la regla no tendrá efecto en el destino del paquete, pero los contadores en la regla se incrementarán.
Ejemplo: iptables -A INPUT --dport 80 -j DROP

-D, --delete
Borra una regla de una cadena especificada. Existen dos maneras de borrar una regla en la cadena:

> Pasar como argumento toda la regla, la cual deberá coincidir exactamente con la existente. Su sintaxis debe ser igual a la del comando –A (o –I o –R), pero sustituyendo –A por –D. Si hay múltiples reglas iguales en la misma cadena, sólo la primera será eliminada.
> **Ejemplo:** iptables -D INPUT --dport 80 -j DROP
>
> Pasar como parámetro el número de línea que ocupa la regla en la cadena.
> **Ejemplo:** iptables -D INPUT 1

-R, --replace
Reemplaza una regla. Recibe como argumento la posición de la regla (a reemplazar) dentro de la cadena y la nueva regla.
Ejemplo: iptables -R INPUT 1 -s 192.168.0.1 -j DROP

-I, --insert
Inserta una regla en el lugar de la cadena que se pasa como argumento. En IPTables es de suma importancia el orden en que están las reglas dentro de las cadenas. Si no se especifica la posición de la regla, se inserta en la posición 1 por defecto.
Ejemplo: iptables -I INPUT 1 --dport 80 -j ACCEPT

-L, --list
Muestra las reglas que contiene la cadena que se pasa como argumento.
Ejemplo: iptables -L INPUT

-F, --flush
Borra todas las reglas de una cadena.
Ejemplo: iptables -F INPUT

-Z, --zero
Pone en cero todos los contadores de una determinada cadena.
Ejemplo: iptables -Z INPUT

-N, --new-chain
Permite al usuario crear su propia cadena. En este ejemplo la cadena se llamará "prueba"
iptables -N prueba

-X, --delete-chain
Borra la cadena especificada. Si se escribe -X solamente, borrará todas las cadenas creadas en esa tabla.
Ejemplo: iptables -X prueba

-P, --policy
Le dice al Kernel que hacer con los paquetes que no coinciden con ninguna regla.
Ejemplo: iptables -P INPUT DROP

-E, --rename-chain
Cambia el nombre de una cadena.
Ejemplo: iptables -E allowed disallowed

-p, --protocol
El paquete debe pertenecer al protocolo especificado después de -p. Cada protocolo se corresponde a un entero. Por ejemplo, ICMP es equivalente a 0. La opción -p ALL son los tres (ICMP, TCP, UDP). Una lista de todos los protocolos válidos puede encontrarse en el archivo /etc/protocols.
Ejemplo: iptables -A INPUT -p tcp

-i, --in-interface
Permite seleccionar paquetes que vienen de una determinada interface. Solo es válida en las cadenas INPUT, FORWARD y PREROUTING, y enviaría un error en cualquier otra. Es posible usar el símbolo "+" al final del nombre de la interfaz para especificar a todas las interfaces que comienzan con el nombre especificado, por ejemplo eth+ indica todas las interfaces ethernet. También puede usarse el símbolo de negación "!" para invertir el significado, por ejemplo, -i ! eth0 indica todas las interfaces, excepto la eth0.
Ejemplos: iptables -A INPUT -i eth0
iptables -A INPUT -i ! eth0
iptables -A INPUT -i eth+

-o, --out-interface
Permite seleccionar paquetes que salen de una determinada interface. Solo es válida en las cadenas OUTPUT, FORWARD y POSTROUTING y enviaría un error en cualquier otra. Es posible usar el símbolo "+" al final del nombre de la interfaz para especificar a todas las interfaces que comienzan con el nombre especificado, por ejemplo eth+ indica todas las interfaces ethernet. También puede usarse el símbolo de negación "!" para invertir el significado, por ejemplo, -i ! eth0 indica todas las interfaces excepto la eth0.
Ejemplo: iptables -A INPUT -o eth0, iptables -A INPUT -o ! eth0, iptables -A INPUT -o eth+

-f, --fragment
Permite seleccionar las segundas o terceras partes de paquetes fragmentados. Estos paquetes son peligrosos. Es posible usar el símbolo "!".
Ejemplo: iptables -A INPUT –f, iptables -A INPUT ! –f

--sport, --source-port
Permite seleccionar o excluir paquetes de un determinado puerto o rango de puertos TCP o UDP (dependiendo del argumento de la opción -p) origen. El puerto puede escribirse también con el nombre del servicio (en linux están en /etc/services), pero esto supone una pequeña sobrecarga que se hace considerable con gran cantidad de reglas. Es posible especificar el símbolo "!" para especificar los puertos o rango de puertos que no vienen de los especificados.
Ejemplo: iptables -A INPUT -p tcp --sport 22
--source-port 22:80 (puertos de 22 a 80)
--source-port :80 (puertos de 0 a 80)
--source-port ! 6:1024 (todos los puertos, menos los del rango de 6 a 1024)

--dport, --destination-port
IDEM anterior pero con puerto tcp destino
Ejemplo: iptables -A INPUT -p tcp --dport 22

--icmp-type
Evita el tipo de paquetes icmp especificado.
Ejemplo: iptables -A INPUT -p icmp --icmp-type 8. El ejemplo es peligroso, ya que puede redirigir a lugares peligrosos.

--src-range
Selecciona paquetes de ese rango de IPs origen, pudiendo también usarse en forma invertida.
Ejemplos: iptables -A INPUT -p tcp -m iprange --src-range 192.168.1.13-192.168.2.19
iptables -A INPUT -p tcp -m iprange ! --src-range 192.168.1.13-192.168.2.19

--dst-range
IDEM anterior pero con IPs destino
Ejemplo: iptables -A INPUT -p tcp -m iprange --dst-range 192.168.1.13-192.168.2.19

-mac-source
Permite seleccionar paquetes en función a su dirección MAC origen. Solo puede ser usada en las cadenas PREROUTING, FORWARD e INPUT. Es posible usar el símbolo "!" para excluir direcciones MAC.
Ejemplos: iptables -A INPUT -m mac --mac-source 00:00:00:00:00:01
iptables -A INPUT -m mac --mac-source ! 00:00:00:00:00:01

--set-mark
A los paquetes que cumplen las condiciones especificadas, se les establece un determinado valor de marca, que asocia al paquete con la marca. El valor de la marca puede ser un número de 32 bits.
Ejemplo: iptables -t mangle -A PREROUTING -p tcp --dport 22 -j MARK --set-mark 1

--set-tos
Establece el valor de TOS en numeración hexadecimal.
Ejemplo: iptables -t mangle -A PREROUTING -p TCP --dport 22 -j TOS --set-tos 0x10

--ttl-set
Establece un determinado valor de TTL en los paquetes que cumplen la condición especificada.
Ejemplo: iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-set 64

4.2. Trafic Control – TC

[Ref. p]

TC (traffic controller) es un programa del espacio de usuario, el cual puede ser usado para crear y asociar varios tipos de colas con los dispositivos de red, asociar clases con cada una de las colas y configurar filtros, por medio de los cuales el paquete es clasificado.

Sintaxis:

Los elementos entre llaves, {...|...|...}, son requeridos, pero solo se puede indicar uno de ellos separados por '|', y los que están entre corchetes, [...], son opcionales.

tc [OPTIONS] OBJECT { COMMAND | help }
OBJECT = {qdisc | class | filter}
OPTIONS = {-s[tatistics] | -d[etails] | -r[aw]}

Usa abreviaturas para denotar las unidades de las tasas: Kbps significa kilobytes y kbit significa kilobits. Entiende algunos sufijos, para los valores numéricos: kbps (*1024), mbps (*1024*1024), kbit (*1024/8) y mbit (*1024*1024/8). Si el comando recibe un número sin un sufijo, éste asume Bits por Segundo bps, lo que significa que divide entre 8 el número que se le ha proporcionado.

- ***Colas***

Acerca de disciplinas de cola, ya se ha explicado suficiente en el inciso *2. Disciplinas de Cola* en este Capítulo**.**

Sintaxis del comando tc para administrar disciplinas de cola:

tc qdisc [add | del | replace | change | get] dev STRING
[handle QHANDLE][root | ingress | parent CLASSID][estimator INTERVAL TIME_CONSTANT][[QDISC_KIND][help | OPTIONS]]

tc qdisc show [dev STRING] [ingress]

QDISC_KIND = { [p|b]fifo | tbf | prio | cbq | red | htb | etc. }
OPTIONS = ... try tc qdisc add <desired QDISC_KIND> help

add agrega una qdisc al dispositivo dev.

del elimina una qdisc del dispositivo dev.

replace reemplaza la qdisc con otra qdisc.

handle representa el nombre único asignado por el usuario a la disciplina de cola. No hay dos disciplinas de cola con el mismo handle. (Ya ha sido explicado antes).

root indica que la cola está en la raíz en la jerarquía. Solo es posible tener una qdisc raíz por dispositivo. (Ya ha sido explicado antes).

ingress policing en el ingress

parent representa el handle de la disciplina de cola padre. (Ya ha sido explicado antes).

dev es el dispositivo de red, al cual se une la qdisc.

estimator usado para determinar si los requerimientos de la cola han sido satisfechos.

- ***Clases***

Las clases son usadas para agrupar a otras clases o para agrupar paquetes. Una clase siempre tiene asociada una cola. (Ya ha sido explicado antes).

Sintaxis del comando tc para administrar clases:

tc class [add | del | change | get] dev STRING [classid CLASSID][root | parent CLASSID] [[QDISC_KIND][help | OPTIONS]]

tc class show [dev STRING] [root | parent CLASSID]

QDISC_KIND = { prio | cbq | htb |etc. } Una de las disciplinas de cola que soportan clases.
OPTIONS = ... try tc class add <desired QDISC_KIND> help

classid representa el handle que el usuario asigna a la clase. (Ya ha sido explicado antes).

root indica que la clase es la clase raíz en la jerarquía. (Ya ha sido explicado antes).

parent Indica el handle del padre. (Ya ha sido explicado antes).

- ***Filtros o clasificadores***

Los filtros son usados para clasificar paquetes, basados en ciertas propiedades del mismo, por ejemplo, direcciones IP, números de puerto, etc. Las disciplinas de cola usan filtros para asignar paquetes a una de sus clases, lo hacen indicando qué paquetes con una determinada marca “caen” a una clase y son llamados desde dentro de las qdisc con clases. Los filtros son guardados en listas de filtros, las que son ordenas por prioridad en orden ascendente.

Sintaxis del comando tc para administrar filtros:

tc filter [add | del | change | get] dev STRING [pref PRIO] [protocol PROTO]
[estimator INTERVAL TIME_CONSTANT]
[root | classid CLASSID |parent] [handle FILTERID]
[[FILTER_TYPE] [help | OPTIONS]]

tc filter show [dev STRING] [root | parent CLASSID]

FILTER_TYPE = { rsvp | u32 | fw | route | etc. }
FILTERID = ... formato depende del clasificador (se explican más adelante)
OPTIONS = ... try tc filter add <desired FILTER_KIND> help

pref representa la prioridad asignada al filtro. No hay dos filtros que puedan tener la misma prioridad.

protocol usado para identificar solamente a los paquetes que corresponden con el protocolo.

root indica que el filtro está en la raíz de la jerarquía.

classid representa el *handle* de la clase al cual se aplica el filtro.

parent representa el controlador al que estará asociado este clasificador. Este controlador debe ser un nodo ya existente.

handle representa el *handle* por el cual el filtro es identificado de forma única. Su formato es diferente por clasificador.

Estimator es usado para determinar si los requerimientos de la cola han sido satisfechos. Estima el ancho de banda usado por cada clase sobre el intervalo de tiempo apropiado.

Show muestra las opciones de un filtro existente.

Tipos de filtros

- **Rsvp**

Usa el protocolo RSVP – Resource Reservation Protocol para la clasificación. Este filtro encamina los paquetes basándose en RSVP y sólo es útil en las redes internas, ya que Internet no respeta RSVP.

Sintaxis:

rsvp ipproto PROTOCOL session DST[/PORT | GPI]
[sender SRC[/PORT | GPI]
[classid CLASSID] [police POLICE_SPEC]
[tunnelid ID] [tunnel ID skip NUMBER]
[police POLICE_SPEC]

GPI= {flowlabel NUMBER | spi/ah SPI | spi/esp SPI | u{8|16|32} NUMBER mask MASK at OFFSET}
ipproto es uno de los protocolos IP (TCP, UDP)
session es una dirección destino con o sin puerto
police su propósito es asegurar que el tráfico no exceda ciertos límites.

- **U32**

El filtro U32 es el más avanzado disponible en la implementación actual, sin embargo es complejo y resulta tedioso. Se basa completamente en tablas hash, lo que lo hace robusto cuando hay muchas reglas de filtrado. Basa la decisión en campos del interior del paquete (dirección IP de origen, etc).

Sintaxis:

u32 [SELECTOR ...] [match link HTID] [flowid CLASSID] [police POLICE_SPEC]

SELECTOR = u{32|16|8}

match representa la condición que los paquetes deben cumplir para ser filtrados.

flowid el *classid* de la clase a la que se envían los paquetes filtrados.

police su propósito es asegurar que el tráfico no exceda ciertos límites.

- **Fw**

Basa la decisión en la forma en que el cortafuegos o firewall ha marcado el paquete. Es una manera sencilla de clasificar sin necesidad de aprender la sintaxis de filtros de tc.

Sintaxis:

fw [classid CLASSID] [police POLICE_SPEC]

CLASSID classid de la clase a la que se envían los paquetes filtrados.

police su propósito es asegurar que el tráfico no exceda ciertos límites.

- **Route**

Este clasificador filtra los paquetes basándose en los resultados de las tablas de rutas. Este tipo de filtros, toman su decisión en función del resultado obtenido al pasar el paquete IP por la tabla de ruta.

Sintaxis:

route [from REALM | fromif TAG] [to REALM] [flowid CLASSID] [police POLICE_SPEC]

from realm REALM en la table de enrutamiento ip.

fromif etiqueta de la interface.

to realm REALM en la table de enrutamiento ip.

flowid es la clase a la cual se asocia el paquete.

police su propósito es asegurar que el tráfico no exceda ciertos límites.

- **Tcindex**

[Ref. q]

Basa su comportamiento en el campo *skb->tc_index* ubicado en la estructura de buffer *sk_buff.* Este espacio de buffer es creado para cada paquete que ingresa o es creado desde Linux, por eso, este clasificador debe ser usado únicamente con las disciplinas de cola que pueden reconocer e inicializar el campo *skb->tc_index,* tales como GRED, DSMARK e INGRESS.

Sintaxis:

tcindex [hash SIZE] [mask MASK] [shift SHIFT] [pass_on | fall_through] [classid CLASSID][police POLICE_SPEC]

hash es el tamaño de la tabla lookup.

mask es la máscara de bit.

shift la máscara correcta por número SHIFT.

pass_on define que este paquete pasará.

fall_through define que este paquete no pasará.

classid es la clase a la cual se une el filtro.

police su propósito es asegurar que el tráfico no exceda ciertos límites.

4.3. Ejemplo de Control de Tráfico con la Disciplina de Cola HTB

[Tomado del Manual de HTB] [Ref. i]

Compartiendo el enlace

Problema: Se tienen dos clientes, A y B, ambos conectados a la Internet a través de la interfaz eth0. Se quiere dar 60kbps a B y 40 kbps a A. Luego se quiere subdividir el ancho de banda de A en dos partes, una de 30 kbps para WWW y otra de 10 kbps para el resto. El ancho de banda que no está en uso puede ser utilizado por cualquier clase que lo necesite (en proporción al ancho de banda que se le ha asignado). Ver Figura 9. *HTB – Servicios a Controlar [HTBLUG]*

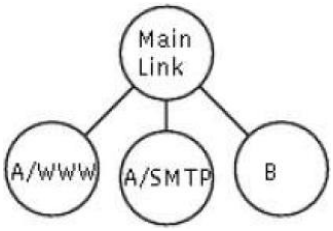


Figura 9. HTB – Servicios a Controlar [HTBLUG]

HTB asegura que la **cantidad de ancho de banda provista a cada clase sea al menos el mínimo entre la cantidad solicitada y la cantidad asignada**. Cuando una clase solicita menos que la cantidad asignada, el ancho de banda remanente (excedente) se distribuye a las otras clases que soliciten servicio.

Nota: esto se denomina “borrowing” (pedir prestado) del ancho de banda excedente. Es importante aclarar, que éste no parece ser el término correcto, porque no hay obligación de devolver el recurso que se está “pidiendo prestado”.

Los diferentes tipos de tráfico son representados por clases en HTB, como lo muestra la figura 9. *HTB – Servicios a Controlar [HTBLUG].*

Se usan los siguientes comandos:

```
tc qdisc add dev eth0 root handle 1: htb default 12
```

Este comando asocia una disciplina de cola HTB “root” a la eth0 y le da el “handle” 1: (esto es solo un nombre o identificador para poder referenciarlo posteriormente). La

parte **default 12** significa que cualquier tráfico que no sea clasificado será asignado a la clase 1:12.

Nota: Los **handles** se escriben **x:y** donde **x** es un entero identificando a una qdisc, mientars **y** es un entero identificando a una clase que pertenece a la qdisc **x.** El **handle** para una qdisc debe tener como valor **y** cero (0) y el **handle** para una clase no debe tener como valor **y** cero (0).

```
tc class add dev eth0 parent 1: classid 1:1 htb rate 100kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:10 htb rate 30kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:11 htb rate 10kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:12 htb rate 60kbps ceil 100kbps
```

La primer línea crea una “root” class, 1:1 debajo de la qdisc 1:. Se define como root class a la que tiene como padre a la qdisc root. Una root class, debajo de una qdisc htb permite que sus hijos “tomen prestado” unos de otros, pero una root class no puede pedir prestado de otra. Se podría haber creado a las otras tres clases directamente debajo de la qdisc htb, pero cuando hubiera tráfico excedente en una, no estaría disponible para las otras. En este caso, se requiere permitir que las clases “tomen prestado”, es por eso que se ha creado una clase extra para servir como root y poner a las clases que realmente mueven el tráfico debajo de ésta.

Nota: El motivo por el cual se repite “dev eth0” en todos los casos y no se coloca solamente el handle, es porque los handles son locales a la interfaz en la que están definidos, por ejemplo, las interfaces eth0 y eth1 pueden tener cada una, clases con el handle 1:1.

A parte de definir las clases hay que describir que paquetes pertenecen a cada clase: (Se identifica a A hipotéticamente con dirección IP 1.2.3.4).

```
tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 \
match ip src 1.2.3.4 match ip dport 80 0xffff flowid 1:10

tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 \
match ip src 1.2.3.4 flowid 1:11
```

Nota: El clasificador U32 tiene una falla de diseño no documentada, el cual causa entradas duplicadas en la lista generada por “tc filter show” cuando se usan clasificadores U32 con valores de prioridad diferente.

Nota: No se ha creado un filtro para la clase 1:12 porque es la clase por omisión (default) y todo el tráfico que no sea clasificado cae directamente sobre ella.

Opcionalmente se pueden conectar disciplinas de cola a las clases hojas. Si no se especifica nada, se les asigna la cola pfifo.

```
tc qdisc add dev eth0 parent 1:10 handle 20: pfifo limit 5
tc qdisc add dev eth0 parent 1:11 handle 30: pfifo limit 5
tc qdisc add dev eth0 parent 1:12 handle 40: sfq perturb 10
```

A continuación se presentarán resultados de varios escenarios utilizando HTB. El autor ha usado las herramientas ethloop y gnuplot para simular tráfico con la disciplina HTB. Se verá qué sucede si se envían paquetes a cada clase a 90 kbps y después se deja de enviar paquetes de una clase a cada momento. En la *figura 10. HTB – Comportamiento del Tráfico [HTBLUG]* hay anotaciones como “0:90k”. La posición horizontal en el medio de la etiqueta (en este caso cerca de 9, marcado también con un 1 en rojo) indica el momento en que el tráfico de alguna clase cambia. Antes de los dos puntos (**:**) está un identificador para la clase (0 para la clase 1:10, 1 para la clase 1:11, 2 para la clase 1:12) y después de los dos puntos está la nueva velocidad arrancando en el momento que aparece la anotación. Por ejemplo la velocidad de la clase 0 se cambia a 90 k al momento 0, la clase 0 cambia a 0k en el instante 3, y de nuevo a 90k en el instante 6.

Inicialmente todas las clases generan 90kb. Dado que esto es mayor que cualquiera de las velocidades especificadas, cada clase es limitada a su velocidad especificada. En el momento 3 cuando se deja de enviar paquetes de la clase 0, la velocidad dada a la clase 0 es reubicada a las otras dos clases en proporción a sus velocidades especificadas, 1 parte para la clase 1 y 6 partes a las clase 2. (El aumento en la clase 1 es difícil de ver porque solo son 4kbps). Similarmente en el momento 9 cuando el tráfico de clase 1 se detiene, su ancho de banda es reubicado a las otras dos (y el aumento en la clase 0 es difícil de ver). Al momento 15 es más fácil ver que la asignación, a la clase 2 se divide en 3 partes para la clase 0 y 1 parte para la clase 1. Al momento 18 ambas clases 1 y 2 se detienen por lo tanto la clase 0 toma todos los 90 kbps que solicita.

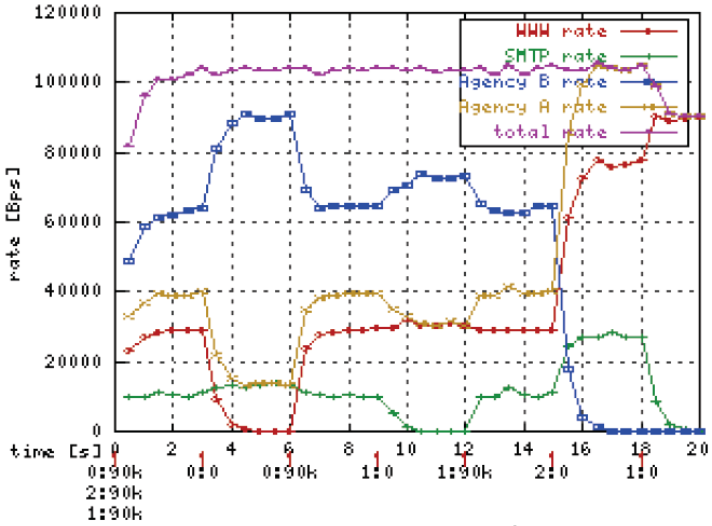


Figura 10. HTB – Comportamiento del Tráfico [HTBLUG]

Debería verse que si varias clases están compitiendo por el ancho de banda de su padre (parent) entonces consiguen ancho de banda en proporción a sus quantums. Es importante saber que para una operación precisa, los quantums deben ser lo más pequeño posible y mayores que la MTU. Por lo general, no es necesario configurar los quantums manualmente, dado que HTB selecciona valores precomputados.

Esto pareciera ser una buena solución si A y B no son clientes diferentes. Sin embargo, si A está pagando por 40kbps, entonces podría preferir que su ancho de banda de WWW en desuso, sea utilizado por otro servicio de él mismo, en vez de que lo utilice B. Este requerimiento puede cubrirse con HTB, utilizando jerarquías de clases, como muestra la Figura 11. *Servicios a controlar con jerarquía*.

Jerarquía de compartición

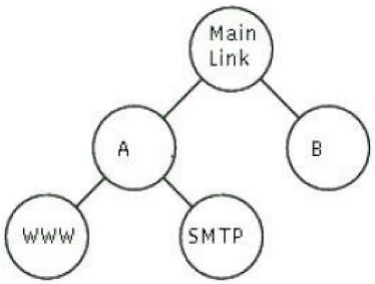


Figura 11. HTB – Servicios a controlar con jerarquía [HTBLUG]

El cliente A ahora está representado explícitamente por su propia clase. Tómese en cuenta que la cantidad de servicio provista a cada clase es al menos el mínimo entre la cantidad que solicita y la cantidad asignada a la clase. Esto se aplica a las clases hojas.

Para las clases HTB que son padres de otras clases HTB, la regla es que la cantidad de servicio es al menos el mínimo entre la suma de la cantidad solicitada por todos sus hijos y la cantidad asignada a la clase.

En este caso se asignan 40kbps al cliente A. Esto significa que si A solicita menos que la cantidad asignada para WWW, el excedente será utilizado por otro tipo de tráfico de A (si lo solicita), al menos hasta que la suma sea 40kbps.

Nota: Las reglas de clasificación de paquetes pueden asignar paquetes a nodos interiores. Para eso se debe conectar otra lista de filtro al nodo interior. Finalmente debería llegar a la clase especial 1:0. La velocidad entregada a un padre (parent) debería ser la suma de las velocidades de sus hijos.

Los comandos son los siguientes:

```
tc class add dev eth0 parent 1: classid 1:1 htb rate 100kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:2 htb rate 40kbps ceil 100kbps
tc class add dev eth0 parent 1:2 classid 1:10 htb rate 30kbps ceil 100kbps
tc class add dev eth0 parent 1:2 classid 1:11 htb rate 10kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:12 htb rate 60kbps ceil 100kbps
```

La Figura 12. *Comportamiento del tráfico jerárquico*, muestra los resultados de la solución jerárquica. Cuando el tráfico WWW de A se interrumpe, su ancho de banda asignado es reubicado a otro tráfico de A, de manera que el ancho de banda total de A se mantenga en los 40kbps asignados. Si todo el tráfico de A solicita menos de los 40kbps, el excedente será asignado a B.

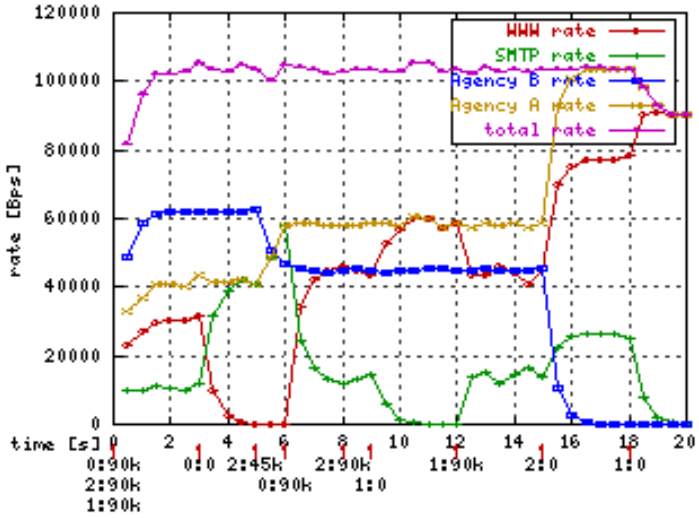


Figura 12. Comportamiento del tráfico jerárquico [HTBLUG]

Tope de las tasas

El argumento *ceil* (techo) limita cuanto ancho de banda puede "pedir prestado" una clase. Ahora se cambia el *ceil* de 100kbps para las clases 1:2 (A) y 1:11 (Otro tráfico de A) por 60kps y 20kbps respectivamente.

En el momento 3, cuando el tráfico de WWW se para, el otro tráfico de A está limitado a 20kbps. Por lo tanto, el cliente A consigue solo 20kbps en total y los 20kbps inutilizados son asignados a B.

La segunda diferencia es al momento 15 cuando B para. Sin el ceil (techo), todo el ancho de banda es entregado a A, pero ahora A solo tiene permitido utilizar 60kpbs, entonces los otros 40kbps quedan inutilizados. Esta funcionalidad debería ser de mucha utilidad para los ISP porque ellos seguramente quieren limitar la cantidad de servicio que consigue un cliente en particular aún cuando los otros clientes no están solicitando servicio. (Los ISP probablemente quieren que los clientes paguen más dinero por un servicio mejor).

Es importante aclarar que las clases *root* no tienen permitido pedir prestado, entonces su ceil es igual al rate.

Nota: El *ceil* para una clase debería ser al menos igual al *rate*. También, el *ceil* de una clase debería ser al menos igual al *ceil* de cualquiera de sus hijos.

Ráfagas

El hardware de networking solo puede enviar un paquete a la vez y a la velocidad del hardware. El software para compartir un enlace puede utilizar esta habilidad para aproximar el efecto de múltiples enlaces a diferentes velocidades. Por lo tanto, la tasa garantizada (*rate*) y la máxima (*ceil*) no son realmente medidas instantáneas, sino promedios durante un tiempo que tome enviar algunos paquetes.

Lo que ocurre realmente es que se deja que una clase curse tráfico enviando unos pocos paquetes a máxima velocidad durante un momento y luego se hace lo mismo con las otras clases.

Los parámetros *burst* y *cburst* controlan la cantidad de datos que pueden ser enviados a máxima velocidad (la velocidad del hardware) sin intentar dar servicio a otra clase. Si *cburst* es más pequeño (idealmente el tamaño de un paquete), modela las ráfagas (shaping) para no exceder la velocidad *ceil*, de la misma forma que lo hace el peakrate de TBF.

Cuando se ve que el *burst* de una clase padre es menor que el de alguna clase hija, entonces es de esperar que la clase padre se quede congelada algunas veces (porque la clase hija solicitará más que lo que la clase padre puede manejar). HTB recordará estas ráfagas negativas hasta 1 minuto.

Las ráfagas son necesarias porque es una forma barata y simple de mejorar los tiempos de respuesta sobre enlaces congestionados. Por ejemplo el tráfico de WWW es de ráfagas. Se puede solicitar una página, que entre en el valor de un *burst* y puede ser leída.

Durante el período de tiempo de inactividad, el valor de *burst* volverá a cargarse de nuevo.

Nota: El valor de *burst* y *cburst* de una clase debería siempre ser al menos igual o mayor que el de cualquiera de sus hijos.

En el gráfico de la Figura 13. *Comportamiento del tráfico jerárquico con cambio en burst y cburst,* puede verse el caso del ejemplo anterior donde se ha cambiado el parámetro *burst* a 20kbps para la clase roja y amarilla (A), pero *cburst* se ha dejado en el valor por defecto (2kb). El crecimiento del verde es al momento 13 debido al valor de *burst* en la clase SMTP. El verde estuvo por debajo del límite desde el momento 9 y acumuló 20kb de *burst*. El crecimiento es hasta 20kbps (limitado por el *ceil* porque tiene *cburst* cercano al valor de 1 paquete).

Se puede llegar a pensar por qué no hay un crecimiento del rojo y el amarillo en el momento 7. Esto es porque el amarillo ya está en el límite de su *ceil* y no tiene más espacio para ráfagas adicionales.

Hay un comportamiento no deseado – el valle que hace el magenta al momento 4. Esto es porque intencionalmente se dejó sin agregar *burst* a la clase del enlace root (1:1). La clase recordó el crecimiento del momento 1 y en el momento 4 cuando la clase azul quiso pedir prestado de la clase amarilla, se lo denegó y se compensó a sí misma.

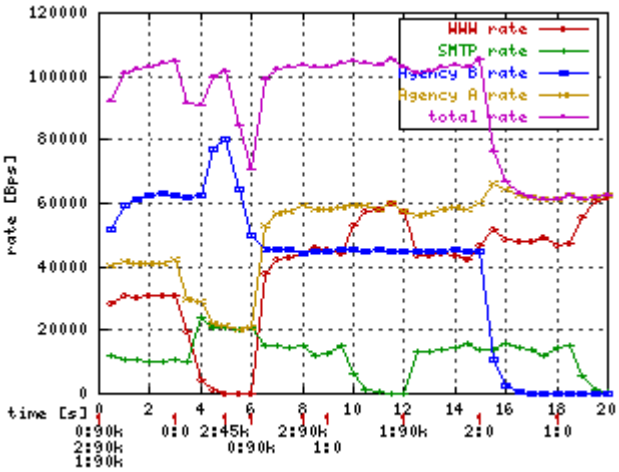


Figura 13. Comportamiento del tráfico jerárquico con cambio en burst y cburst [HTBLUG]

Priorizando la compartición del ancho de banda

Priorizar tráfico tiene dos aspectos:

Primero, cómo el ancho de banda excedente se distribuye entre las clases hermanas. Hasta ahora se ha visto que el ancho de banda excedente se distribuye de acuerdo a los valores de *rate*. Ahora se verá la misma configuración de la parte de jerarquía de clases sin *ceil* ni *burst,* cambiándole la prioridad a todas las clases a 1 excepto a SMTP (verde) a la cual se le asignará 0 (mayor prioridad). Desde el punto de vista del compartido, se puede ver que la clase toma todo el ancho de banda excedente. **La regla es que se le ofrece el ancho de banda excedente primero a las clases con prioridades superiores.** Pero las reglas sobre el *rate* y el *ceil* todavía se respetan. Ver figura 14. *Comportamiento del tráfico con cambios en las prioridades*.

Segundo, el retardo (delay) total del paquete. Es difícil de medir sobre una ethernet porque es muy rápida (el retraso es insignificante), pero hay una ayuda simple. Se puede agregar a la qdisc root, una clase HTB simple, con el *rate* limitado a menos de 100kbps y agregar una segunda qdisc HTB (la que se desea medir) como hija. Entonces se puede simular un enlace más lento con retardos mayores.

Por simplicidad se plantea un escenario de dos clases:

```
# qdisc for delay simulation
tc qdisc add dev eth0 root handle 100: htb
tc class add dev eth0 parent 100: classid 100:1 htb rate 90 kbps
# real measured qdisc
tc qdisc add dev eth0 parent 100:1 handle 1: htb
AC="tc class add dev eth0 parent"
$AC 1: classid 1:1 htb rate 100kbps
$AC 1:2 classid 1:10 htb rate 50kbps ceil 100kbps prio 1
$AC 1:2 classid 1:11 htb rate 50kbps ceil 100kbps prio 1
tc qdisc add dev eth0 parent 1:10 handle 20: pfifo limit 2
tc qdisc add dev eth0 parent 1:11 handle 21: pfifo limit 2
```

Nota: Una qdisc HTB como hija de otra qdisc HTB no es lo mismo que una clase debajo de otra clase dentro de la misma qdisc HTB. Esto se debe a que cuando una clase en HTB puede enviar tráfico, lo hará tan rápido como el hardware pueda. Por lo tanto, el retardo del límite inferior de la clase, está condicionado solo por el equipo. En el caso

de una qdisc HTB debajo de otra qdisc HTB, la disciplina de cola de afuera simula un nuevo equipo de hardware con todas las consecuencias (mayor delay).

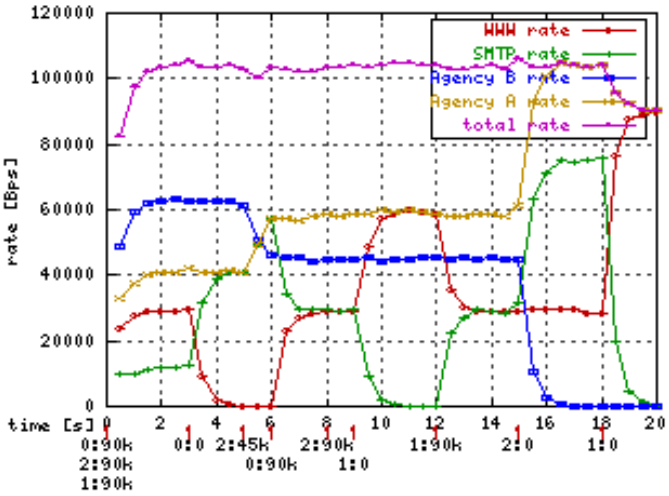


Figura 14. Comportamiento del tráfico jerárquico con cambio en las prioridades [HTBLUG]

Se coloca un simulador para generar 50kbps para ambas clases y a los 3 segundos se ejecuta el siguiente comando:

```
tc class change dev eth0 parent 1:2 classid 1:10 htb rate 50kbps ceil 100kbps burst 2k prio 0
```

Como se puede ver en la figura 15. *Comportamiento del tráfico jerárquico con cambio en las prioridades a los 3 segundos*, el delay de la clase WWW cayó cercano a cero, mientras el delay de SMTP creció. Cuando se prioriza para tener un mejor delay, siempre se hace sobre la base que el delay de las otras clases empeorará. Más tarde (a los 7 segundos) el simulador comienza a generar tráfico WWW a 60 kbps y SMTP a 40kbs. Ahí se puede ver un comportamiento interesante, cuando la clase (WWW) está por encima de sus límites, HTB prioriza la parte del ancho de banda que está por debajo de sus límites.

¿Qué clases deberían priorizarse?

Generalmente aquellas clases donde realmente se necesita bajos delays. El ejemplo anterior podría haber sido tráfico de video o audio o tráfico interactivo (telnet, SSH) que es de ráfagas naturalmente y no afectará negativamente a los otros flujos. Un truco común es priorizar ICMP para tener delays de ping muy buenos, aún sobre enlaces totalmente utilizados (pero técnicamente visto, esto no es lo que se quiere cuando se mide conectividad).

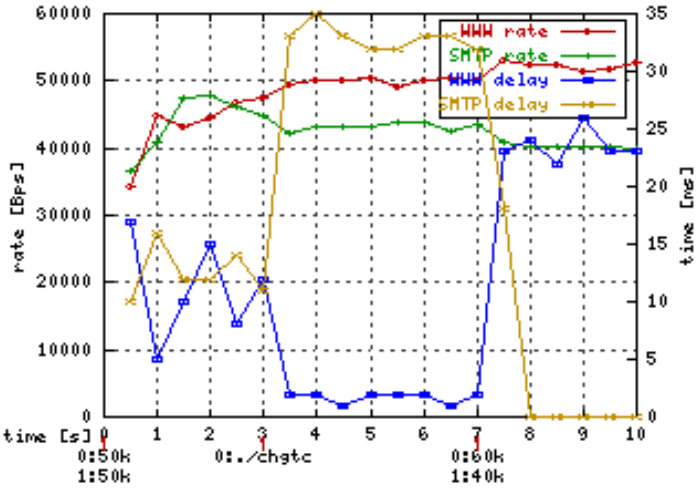


Figura 15. Comportamiento del tráfico jerárquico con cambio en las prioridades a los 3 segundos [HTBLUG]

Analizando las estadísticas

La herramienta tc permite obtener estadísticas de las disciplinas de cola en Linux. Del ejemplo mostrado, se ha tomado las siguientes estadísticas:

Escribir el siguiente comando para las estadísticas de colas:

```
tc -s qdisc show dev eth0
```

Estadísticas obtenidas:

```
qdisc pfifo 22: limit 5p
Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

qdisc pfifo 21: limit 5p
Sent 2891500 bytes 5783 pkts (dropped 820, overlimits 0)

qdisc pfifo 20: limit 5p
 Sent 1760000 bytes 3520 pkts (dropped 3320, overlimits 0)

qdisc htb 1: r2q 10 default 1 direct_packets_stat 0
Sent 4651500 bytes 9303 pkts (dropped 4140, overlimits 34251)
```

Las primeras tres disciplinas son hijas de HTB.

Overlimits indica cuantas veces la disciplina ha retrasado un paquete.

direct_packets_stat indica cuantos paquetes han sido enviados a través de la cola.

Escribir el siguiente comando para las estadísticas de clases:
tc -s class show dev eth0

Estadísticas obtenidas:
class htb 1:1 root prio 0 rate 800Kbit ceil 800Kbit burst 2Kb/8 mpu 0b
cburst 2Kb/8 mpu 0b quantum 10240 level 3
Sent 5914000 bytes 11828 pkts (dropped 0, overlimits 0)
rate 70196bps 141pps
lended: 6872 borrowed: 0 giants: 0

class htb 1:2 parent 1:1 prio 0 rate 320Kbit ceil 4000Kbit burst 2Kb/8 mpu 0b
cburst 2Kb/8 mpu 0b quantum 4096 level 2
Sent 5914000 bytes 11828 pkts (dropped 0, overlimits 0)
rate 70196bps 141pps
lended: 1017 borrowed: 6872 giants: 0

class htb 1:10 parent 1:2 leaf 20: prio 1 rate 224Kbit ceil 800Kbit burst 2Kb/8 mpu 0b cburst 2Kb/8 mpu 0b quantum 2867 level 0
Sent 2269000 bytes 4538 pkts (dropped 4400, overlimits 36358)
rate 14635bps 29pps
lended: 2939 borrowed: 1599 giants: 0

No se han considerado las clases 1:11 y 1:12 para hacer la salida más corta. En la salida obtenida, se puede ver que los parámetros de las clases están configurados, se encuentra la información para los niveles y para los quantum.

Overlimits muestra cuantas veces una clase ha sido solicitada para enviar un paquete, pero no lo ha podido hacer por las restricciones de las tasas (*rate* / *ceil*). Actualmente se cuenta únicamente para las clases hojas.

Lended es el número de paquetes donados por la clase (de su *rate*).

Borrowed es el número de paquetes que se solicitan prestados al padre.

Los paquetes que se dan en préstamo, siempre son calculados localmente para la clase, mientras que los paquetes solicitados (que se piden prestados) son transitivos (cuando 1:10 solicita prestado de 1:2 que a su vez solicita de 1:1, tanto el contador de la clase 1:10 como el de la clase 1:2 son incrementados).

Giants es el número de paquetes más grandes que la MTU. HTB trabajará con este tipo de paquetes, pero las tasas no serán exactas.

5. El Dispositivo Intermedio de Encolado IMQ

IMQ (Intermediate Queing Device): se usa para encolar paquetes y limitar cuánto tráfico puede llegar a la interfaz de red. Tiene utilidad a la hora de limitar el tráfico **entrante.**

IMQ no es una qdisc, pero su uso está muy unido a las qdisc. Dentro de Linux, las qdiscs se asocian a dispositivos de red y todo lo que se encola en el dispositivo se encola antes en la qdisc. Partiendo de este concepto, surgen dos limitaciones:

- Sólo se pueden hacer ajustes de salida (existe una qdisc de entrada, pero sus posibilidades son muy limitadas comparadas a las qdiscs con clases y no está aún documentada). [Ref. o]
- Una qdisc sólo puede ver el tráfico de una interfaz, de manera que no se pueden imponer limitaciones globales.

IMQ trata de resolver ambas limitaciones. Como *tc* solo tiene colas de salida, entonces se puede aplicar un truco para tratar el tráfico de entrada como si fuera el de salida, desviando todo el tráfico de entrada a un dispositivo de red virtual, IMQ, esto se hace a través de *iptables*. De manera que, el tráfico que entra por una determinada interfaz, se convierte en tráfico de salida de IMQ y así podrá ser controlado al gusto.

Los objetivos IMQ de *iptables* son válidos en las cadenas PREROUTING y POSTROUTING de la tabla MANGLE.

Sintaxis:
IMQ [--todev n]

n: número del dispositivo imq

Ejemplo:

Definir las clases y colas

```
tc qdisc add dev imq0 root handle 1: htb default 20
tc class add dev imq0 parent 1: classid 1:1 htb rate 2mbit burst 15k
tc class add dev imq0 parent 1:1 classid 1:10 htb rate 1mbit
tc class add dev imq0 parent 1:1 classid 1:20 htb rate 1mbit
```

```
tc qdisc add dev imq0 parent 1:10 handle 10: pfifo
tc qdisc add dev imq0 parent 1:20 handle 20: sfq
```

Filtrar el tráfico

```
tc filter add dev imq0 parent 1:0 protocol ip prio 1 u32 match ip dst 10.0.0.230/32 flowid 1:10
```

Escoger y marcar el tráfico para encolarlo en imq0

```
iptables -t mangle -A PREROUTING -i eth0 -j IMQ --todev 0 ip link set imq0 up
```

Esta regla envía todo el tráfico que viene por eth0 a imq0 antes de empezar el proceso de routing y habilita el dispositivo virtual imq0.

VI. Análisis y Presentación de Resultados

Capítulo 2. Algoritmo de Control de Tráfico en Linux

El control de tráfico en Linux es realizado a través de herramientas de Calidad de Servicio QoS (Capítulo 1. *Control de Tráfico en Linux. Calidad de Servicio*) que son proporcionadas por el propio kernel de Linux desde su versión 2.2. Estas herramientas deben ser utilizadas por los administradores de red para implementar el control del tráfico en sus redes. [Ref. a]

En este capítulo, se proporciona un algoritmo que ayudará a los administradores a realizar esta labor. En el Capítulo 3. *Caso de Estudio*, se aplicará dicho algoritmo a un caso de estudio de una red particular.

En los ejemplos, se utilizarán las disciplinas de cola **HTB** y **SFQ**; los comandos **iptables,** para marcado de paquetes, y **tc** para crear las clases, colas y filtros. La selección de las herramientas de control de tráfico y disciplinas de cola, depende de las características de las mismas, del escenario y los requerimientos de la red a tratar. Por lo que, es necesario leer el Capítulo 1. *Control de Tráfico en Linux,* para entender los conceptos y comandos utilizados.

Para controlar el tráfico de salida, se emplea la técnica de encolado (Capítulo 1. *Control de Tráfico en Linux. Técnica de Encolado*). El algoritmo consta de tres pasos indispensables: Marcado de tráfico, creación del árbol de preferencias, filtrado de paquetes; los que se explican en detalle a continuación.

Nota: El filtrado de paquetes es el puente que une al marcado de paquetes con el árbol de preferencias. Por lo que, el filtrado, es el último de los pasos a realizar, pero el orden de realización del marcado y la creación del árbol de preferencias, es indistinto.

El algoritmo es el siguiente:

1. **Marcar el tráfico que interesa caracterizar.** Las posibilidades de clasificación de los filtros de la arquitectura de control de tráfico (tc) son bastante limitadas en comparación con el soporte de netfilter (firewalling de Linux). Es por esto que se recomienda utilizar iptables, como herramienta para realizar la etapa de marcado, que permite mayores capacidades de selección de paquetes. [Ref. d]

   Usando iptables, se requiere establecer "marcas" a los paquetes. La marca debe ser un valor entero de 32 bits (2^{32} es su valor máximo) que se le asocia a los paquetes para luego tomar decisiones con los mismos según su marca. [Ref. o, d, m]

Por ejemplo, se puede realizar de la siguiente manera:

```
>iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 80 -tmangle - j MARK --set-mark 1
```

Este comando marca con un 1 (--set-mark) todos los paquetes que se reenvíen (FORWARD) a otra máquina, que lleguen por la interfaz eth1 y que salgan a través de la interfaz eth0 con un protocolo tcp (-p), destinados al puerto 80 (--dport), se use la tabla de manejo de paquetes (-t mangle), se cargue el módulo de marcado (- j MARK).

Cabe aclarar, que el orden de las reglas de marcado de paquetes es importante para iptables.

2. **Construir la jerarquía de clases para la clasificación de los paquetes.** Las clases se estructuran en forma de árbol con:
   - Un nodo raíz, que está unido directamente a la interfaz de red y siempre tiene asociada una disciplina de cola.
   - Sus clases hijas, que en caso de ser clases hojas, se les asigna el tráfico según las marcas establecidas en el paso 1.

Según muestra la figura 16. *Estructura jerárquica de clases*, los nombres de los nodos deben ser escritos tomando en cuenta lo siguiente: [Ref. d y b]

a. Cada nombre debe tener dos partes: un número Mayor y un número Menor, separados de dos puntos, de esta manera **Mayor: Menor**.
b. Los nombres son únicos para las colas y clases.
c. Si el nombre está unido a una disciplina de cola y no a una clase, *Menor* debe ser cero o simplemente no ser escrito.
d. Si el nombre está unido a una clase, *Mayor* debe ser igual al número *Mayor* del nodo al que corresponde la clase en la estructura jerárquica, y *Menor* no debe ser cero.
e. El número *Mayor* debe ser único dentro de una configuración de entrada o salida.
f. El número *Menor* debe ser único dentro de una cola y sus clases.
g. Los nombres son locales a la interfaz de red a la que son definidos. Por ejemplo las interfaces eth0 y eth1 pueden tener cada una, clases definidas con el mismo nombre.

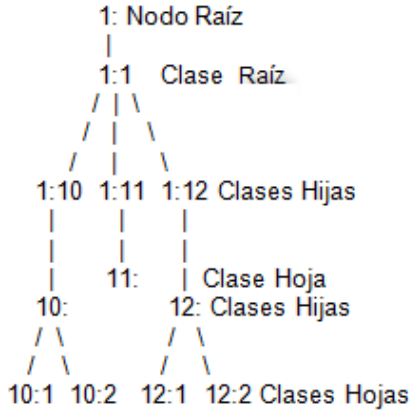


Figura 16. Estructura jerárquica de clases.

La clase raíz mostrada en la Figura 16, tiene una razón especial en la estructura jerárquica, la cual será explicada más adelante.

2.1. Crear el nodo raíz. [Ref. o]

El nodo raíz es una qdisc, es el principal y está asociado a la tarjeta de red. Siempre debe estar presente porque es requerido para colgar de éste las clases que se encargarán de mover el tráfico y establece la clase por defecto donde se enviará el tráfico no clasificado.

Por ejemplo:

```
>tc qdisc add dev eth0 root handle 1: htb default 11
```

Este comando agrega el nodo raíz (root) de una disciplina de cola htb al interfaz eth0 (la interfaz de salida del tráfico), lo nombra 1: (handle) e indica que el tráfico no clasificado (default) será enviado a la clase con id 11.

2.2. Crear la clase raíz y colgarla del nodo raíz. [Ref. o]

La razón de la clase raíz, es aprovechar la característica de realización de "préstamos" de ancho de banda que proporciona el algoritmo HTB, para que éste sea utilizado por las clases que lo soliciten y no sea desperdiciado. Sin embargo, el nodo raíz no puede prestar ancho de banda, ni una clase "hermana" puede pedir prestado de otra directamente, sino a través de su padre. Para solucionar este problema, se debe colocar un nodo intermedio entre el nodo raíz y los nodos hijos. Dicho nodo, debe tener garantizado el total del ancho de banda del enlace (máximo que podrá ser alcanzado por las clases que mueven el tráfico) y sí podrá "prestar" ancho de banda a otras clases. En caso de que no se pretenda aprovechar el tráfico excedente de las clases, o no se utilice la cola HTB, esta clase raíz no es de utilidad funcional.

Por ejemplo:

```
>tc class add dev eth0 parent 1: classid 1:1 htb rate 300 kbps ceil 300 kbps
```

Este comando agrega una clase con identificador 1 y padre 1: (parent) que fue creada en *inciso 2.1*, por lo que, al depender del nodo 1, se le denomina 1:1 (classid), la asocia al interfaz eth0, establece su ancho de banda garantizado (rate) y el máximo que puede alcanzar (ceil).

2.3. Crear las clases a las que se asignarán los paquetes. En cada clase se indica un comportamiento de limitación de tráfico (rate / ceil). La jerarquía de clases, debe ser realizada según las agrupaciones que se definan por tipos de servicios para el escenario de red en cuestión. Las clases deben colgar de la clase raíz que fue creada en *inciso 2.2.* y cumplir las siguientes reglas de limitación de tráfico: [Ref o y i]

a. El ceil por defecto es el rate configurado, lo cual implica, que en caso de no especifica un ceil, no hay préstamos.
b. El rate debe ser menor o igual al ceil de la clase. Rate <= Ceil
c. El rate es igual al ceil para la clase raíz. Rate = Ceil root
d. La suma de los rate de las clases hijas debe ser menor o igual al ceil de la clase padre. Suma (Rate de hijos) <= Ceil padre
e. La suma de los rate de las clases hijas debe ser menor o igual al rate de la clase padre. Suma (Rate hijos) <= Rate padre
f. La suma de los ceil de las clases hijas debe ser menor o igual al ceil de la clase padre. Suma (Ceil hijos) <= Ceil padre

Por ejemplo:

```
>tc class add dev eth0 parent 1:1 classid 1:10 htb rate 150 kbps ceil 300 kbps
>tc class add dev eth0 parent 1:1 classid 1:11 htb rate 100 kbps ceil 300 kbps
```

Este comando crea dos clases identificadas como 10 y 11 respectivamente, con nodo padre 1:1 (parent) que fue creado en *inciso 2.2*, por lo que se les denomina 1:10 y 1:11 (classid) respectivamente y se ha establecido su ancho de banda garantizado (rate) y el máximo que pueden alcanzar (ceil) cada una de ellas.

2.4. Asociar colas a las clases creadas. [Ref. o]

A cada clase se asocia una disciplina de cola que se encargará de enviar los paquetes a la red. De obviar este paso, la disciplina de cola por defecto es pfifo_fast.

Por ejemplo:

```
>tc qdisc add dev eth0 parent 1:10 handle 10: sfq
>tc qdisc add dev eth0 parent 1:11 handle 11: sfq
```

Este comando asocia una cola sfq a cada clase creada en *el inciso 2.3*, de manera que cuando el tráfico es asignado a la clase respectiva, éste sea repartido a la red a través del algoritmo sfq.

3. Configurar los filtros. [Ref. o]

Los filtros son usados para indicar qué paquetes, a través de su marca, son asignados a una clase determinada. Las clases son como las puertas por donde pasa el ancho de banda, entonces se deberá clasificar el tráfico en la puerta correcta según sus características, de modo que el tráfico total queda clasificado en tipos para ser tratados de forma diferente.

Por ejemplo:

```
>tc filter add dev eth0 protocol ip parent 1:  handle 1 fw classid 1:10
```

Este comando crea un filtro asociado al nodo raíz 1: (parent) e interfaz eth0 e indica que cualquier paquete con la marca 1 (handle) se reenvíe (fw) a la clase 10.

Con esta jerarquía de clases, se asegura ancho de banda a cada clase y se aprovecha el ancho de banda no usado de otras clases.

Capítulo 3. Control de Tráfico. Caso de Estudio: Red Informática UNI.

La Universidad Nacional de Ingeniería UNI, a través de la División de Informática y Tecnología de la Información DITI, quien administra la red interna, requiere de una orientación técnica, que le permita realizar una distribución equitativa del ancho de banda total del enlace de comunicación, entre los servicios que ofrece la red, conforme al uso de los mismos y el nivel de importancia que tienen para los usuarios.

Según el mapa topológico actual de la red de la UNI (*Ver Anexo 1. Mapa Topológico de la Red Informática UNI*), el escenario de la red consiste en cinco servidores Linux principales con kernel 2.6.26, éstos son gateway0 - GW0, gateway1 - GW1, gateway4 - GW4, gateway7 - GW7 y gateway8 - GW8. Estos Gateway tienen múltiples interfaces de red y controlan el tráfico de todos los equipos conectados a la red.

El GW0 es el principal, ya que se encarga de enrutar el tráfico de los otros Gateway y permite el acceso a internet vía transceiver y fibra óptica, con un enlace total disponible de ancho de banda de 10 Mbits. Tiene dos interfaces de red, una de entrada eth0 (192.168.103.1) a la red interna de la UNI y una de salida eth1 (165.98.225.242) hacia el proveedor de servicio de Internet, con la cual se le da salida al tráfico proveniente de los otros Gateway, GW8 eth2 (192.168.103.244), GW7 eth2 (192.168.103.220), GW1 eth0 (192.168.103.96), GW4 eth0 (192.168.103.245).

A fin de determinar las necesidades de la red y ofrecer un mejor servicio, se requiere hacer un análisis de su tráfico, para conocer el tráfico a priorizar y así aplicarle un tratamiento diferente al resto del tráfico de la red. Atendiendo este requerimiento, se ha realizado un monitoreo de la red aplicado a una muestra de la misma. Para el muestreo se usó el GW8, donde se ejecutó el programa tshark, que es un software analizador de paquetes de red en Linux, y Wireshark, que es la herramienta gráfica de tshark. Como resultado del monitoreo, se ha obtenido estadísticas del uso de la red en horas del día y la noche, durante la semana y el fin de semana (*Ver Anexo 2. Estadísticas de Red*).

Según el análisis de las estadísticas obtenidas del monitoreo y las políticas actuales de servicio de la administración, las necesidades son las siguientes:

- Priorizar el tráfico http (puerto tcp/udp 80) y https (puerto tcp 443), como servicios a garantizar a los usuarios de la red.
- Tener disponibilidad de los siguientes servicios locales:

  Dnscache (puerto tcp/udp 53)

  Webcaché (puerto tcp/udp 3128)

  Squid (puerto tcp/udp 3128)

  Dhcp para redes internas (puertos udp 67 y 68)

  Ssh(puerto tcp/udp 22)

  SNMP (puerto tcp 161)

  Webmin(puerto tcp 10000)
- Restringir el acceso a redes para descarga de videos. La administración ha detectado redes, a las cuales se conectan los usuarios vía web, para descargar videos que consumen en grande el ancho de banda.
- Denegar los siguientes servicios:
  - Microsoft NET (puertos tcp/udp 135, 139, 445)
  - P2P (puertos tcp/udp 1214, 1900, 2422, 4660-3, 6699, 7777, 7880, tcp 4662, 5222, udp 4672, 23399)
  - MSN (puertos tcp/udp 1863, tcp 6891- tcp/udp 6901, tcp 28800 – tcp 29000) con excepción de los usuarios que el administrador ha designado privilegiados para el uso de este servicio. Estos se distinguen por los números MAC de sus interfaces de red.
- Garantizar a los usuarios especiales un libre acceso a todo tipo de navegación (incluyendo descarga de videos, software P2P y MSN) y una mayor velocidad de conexión respecto a los usuarios comunes. Existen casos especiales de usuarios privilegiados, que la administración ha seleccionado para proveer los servicios mencionados. Estos se distinguen por los números MAC de sus interfaces de red.

Estos requerimientos, son válidos para los Gateway GW8, GW7, GW4 y GW1.

De acuerdo al escenario y los requerimientos planteados, se recomienda implementar una solución basada en Calidad de Servicio, bajo un entorno Debian GNU/Linux, para realizar un control de tráfico que permita distribuir el ancho de banda de una manera justa entre los servicios que se ofrecen y establecer niveles de prioridad de usuarios y ejecución de servicios.

Para implementar el sistema de control de tráfico, se necesita de las siguientes herramientas:

- El Kernel de Linux versión 2.3 o superior. Disponible en el sitio http://www.kernel.org. Es recomendable instalar el último disponible para dar soporte a las herramientas y parches que se requieran.

- Iproute2. Contiene la herramienta *tc* necesaria para configurar el árbol de preferencias (Colas y Clases). Disponible en el sitio http://linux-net.osdl.org/index.php/Iproute2

- Iptables versión 1.2.9 o superior. Disponible en el sitio http://www.netfilter.org

De acuerdo al escenario de red, se proponen tres posibles alternativas de solución:

Alternativa No.1

Definir reglas de control de tráfico en los Gateway GW8, GW7, GW1 y GW4. Con esta alternativa, las reglas son aplicadas en cada Gateway conectado al GW0, siendo la administración distribuida en cada equipo.

Alternativa No.2

Definir reglas de control de tráfico en cada uno de los Gateway existentes GW0, GW8, GW7, GW1, y GW4. Con esta alternativa las reglas son aplicadas en cada Gateway, siendo la administración distribuida en cada equipo. Además, el tráfico proveniente de los Gateway GW8, GW7, GW1, y GW4 es nuevamente filtrado en el GW0.

Alternativa No.3

Definir reglas de control de tráfico en el GW0 para filtrar el tráfico proveniente de los GW8, GW7, GW1 y GW4. Con esta alternativa las reglas son aplicadas únicamente en el GW0, centralizando la administración en este equipo.

Metodología de Trabajo

El estudio será realizado en base a la alternativa número uno y haciendo uso de una muestra de la red, en este caso, el GW8. La razón por la cual se ha elegido esta alternativa y este Gateway es el acceso que la universidad ha proporcionado para hacer las pruebas pertinentes al estudio. El GW8 administra un segmento de la red, en el que, realizar pruebas, no afecta el enlace total universitario, de esta manera, se minimiza cualquier impacto negativo que pudieran ejercer las pruebas en el rendimiento de la red. Mientras que, en el GW0 convergen todas las redes de la universidad y cualquier prueba con resultados fallidos en este Gateway, puede provocar una caída en el enlace global universitario.

El trabajo consistirá en escribir una propuesta de reglas de control de tráfico para ejecutar en el Gateway 8, basadas en los requerimientos de los usuarios de la red y especificando las consideraciones necesarias para realizar un control óptimo, luego estas reglas serán puestas en funcionamiento en dicho Gateway para observar el desempeño del segmento de red en cuestión. Finalmente, se obtendrán estadísticas del comportamiento del tráfico (una vez las reglas en ejecución) y se mostrarán gráficamente.

La solución a aplicar en el GW8 es replicable al resto de los Gateway y a las otras alternativas de solución. Cada Gateway sugiere cambios en la reglas de control de tráfico al especificar las interfaces de red, y cada alternativa puede requerir cambios en los números MAC para acceso a ciertos servicios, u otras particularidades que la administración pudiera estimar convenientes. Se escribirá un script flexible para generar las reglas, de manera que estos cambios sean fáciles de realizar al replicar la solución.

Propuesta de Control de Tráfico

Tomando como base lo siguiente:

- El algoritmo definido en el Capítulo 2. *Algoritmo de Control de Tráfico en Linux.*
- El escenario de red existente y los requerimientos solicitados (anteriormente definidos en este capítulo).

Se proporciona una propuesta de solución para la **Alternativa No.1** Administración Distribuida: Definir reglas de control de tráfico en los Gateway GW8, GW7, GW1 y GW4.

Para el GW8:

En primer lugar, es necesario **Marcar el Tráfico** que interesa caracterizar, en este caso, el tráfico web, los servicios locales, MSN para MAC permitidos, usuarios especiales y video.

Se establecen marcas para los paquetes a priorizar que lleguen por los interfaces de entrada eth1, eth3, eth4 y que salen a través de la interfaz de salida eth2 del GW8. *Ver figura 17.Interfaces de Red del GW8.*

DATOS PARA EL MARCADO DE PAQUETES

GATEWAY	IP GATEWAY	IP ETHERNET	INTERFAZ ENTRADA	INTERFAZ SALIDA	MARCA	TIPO DE CADENA	UBICACION
GW8	192.168.103.244	192.168.103.244		eht2	1	FORWARD	BACKBONE
		192.168.141.1	eth3				RED SOTANO
		192.168.73.1	eth4				FEC 3ER PISO
		192.168.145.1	eth1				RECTORIA

Figura 17. Interfaces de Red del GW8

REGLAS PARA BLOQUEO DE TRAFICO TCP y UDP PARA LOS USUARIOS COMUNES

```
iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 135 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 139 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 445 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 2422 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 139 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 445 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 1214 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 1900 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 2422 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 4660:4663 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 6699 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 7777 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 7880 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 4672 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth3 -o eth2 -p udp --dport 23399 -m mac --mac-source ! 00:21:29:6d:12:a5 -t filter -j REJECT
iptables -A FORWARD -i eth3 -o eth2 -p tcp --dport 1900 -m mac --mac-source ! 00:16:44:bd:62:0e -t filter -j REJECT
iptables -A FORWARD -i eth3 -o eth2 -p tcp --dport 2422 -m mac --mac-source ! 00:16:44:bd:62:0e -t filter -j REJECT
iptables -A FORWARD -i eth3 -o eth2 -p tcp --dport 4660:4663 -m mac --mac-source ! 00:16:44:bd:62:0e -t filter -j REJECT
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 135 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 139 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 445 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
```

```
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 1214 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 1900 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 2422 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 4660:4663 -m mac --mac-source ! 00:0f:66:ed:65:de -t filter -j REJECT
```

La denegación de los servicios es realizada en la cadena FORWARD de la table **filter**, ya que el paquete es reenviado por el GW8. La tabla *filter* es donde se hace el filtrado principal para decidir si el paquete es descartado o aceptado.

REGLAS PARA MARCADO DE TRÁFICO WEB TCP y UDP

```
iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 80 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1
iptables -A FORWARD -i eth3 -o eth2 -p tcp --dport 80 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 80 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1
iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 443 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1
iptables -A FORWARD -i eth3 -o eth2 -p tcp --dport 443 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 443 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1
iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 3128 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1
iptables -A FORWARD -i eth3 -o eth2 -p tcp --dport 3128 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 3128 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 80 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1
iptables -A FORWARD -i eth3 -o eth2 -p udp --dport 80 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1
iptables -A FORWARD -i eth4 -o eth2 -p udp --dport 80 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 3128 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1
```

iptables -A FORWARD -i eth3 -o eth2 -p udp --dport 3128 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1
iptables -A FORWARD -i eth4 -o eth2 -p udp --dport 3128 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 1

REGLAS PARA MARCADO DE TRÁFICO LOCAL TCP

iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 53 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth3 -o eth2 -p tcp --dport 53 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 53 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 22 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth3 -o eth2 -p tcp --dport 22 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 22 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 161 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth3 -o eth2 -p tcp --dport 161 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 161 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 10000 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth3 -o eth2 -p tcp --dport 10000 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 10000 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2

REGLAS PARA MARCADO DE TRÁFICO LOCAL UDP

iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 53 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth3 -o eth2 -p udp --dport 53 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth4 -o eth2 -p udp --dport 53 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 22 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth3 -o eth2 -p udp --dport 22 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2

iptables -A FORWARD -i eth4 -o eth2 -p udp --dport 22 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 67 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth3 -o eth2 -p udp --dport 67 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth4 -o eth2 -p udp --dport 67 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth1 -o eth2 -p udp --dport 68 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth3 -o eth2 -p udp --dport 68 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2
iptables -A FORWARD -i eth4 -o eth2 -p udp --dport 68 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 2

REGLAS PARA MARCADO DE TRÁFICO MSN PARA USUARIOS PERMITIDOS

iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 1863 -m mac --mac-source 00:08:54:1f:7d:22 -t mangle -j MARK --set-mark 3
iptables -A FORWARD -i eth3 -o eth2 -p tcp --dport 1863 -m mac --mac-source 00:08:54:1f:7d:22 -t mangle -j MARK --set-mark 3
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 1863 -m mac --mac-source 00:08:54:1f:7d:22 -t mangle -j MARK --set-mark 3
iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 6891:6901 -m mac --mac-source 00:08:54:1f:7d:22 -t mangle -j MARK --set-mark 3
iptables -A FORWARD -i eth3 -o eth2 -p tcp --dport 6891:6901 -m mac --mac-source 00:08:54:1f:7d:22 -t mangle -j MARK --set-mark 3
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 6891:6901 -m mac --mac-source 00:08:54:1f:7d:22 -t mangle -j MARK --set-mark 3
iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 28800:29000 -m mac --mac-source 00:08:54:1f:7d:22 -t mangle -j MARK --set-mark 3
iptables -A FORWARD -i eth3 -o eth2 -p tcp --dport 28800:29000 -m mac --mac-source 00:08:54:1f:7d:22 -t mangle -j MARK --set-mark 3
iptables -A FORWARD -i eth4 -o eth2 -p tcp --dport 28800:29000 -m mac --mac-source 00:08:54:1f:7d:22 -t mangle -j MARK --set-mark 3

REGLAS PARA MARCADO DE TRÁFICO DE VIDEO

iptables -A FORWARD -i eth1 -o eth2 -p tcp -d 69.80.234.0/24 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 4
iptables -A FORWARD -i eth1 -o eth2 -p tcp -d 206.169.213.0/24 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 4
iptables -A FORWARD -i eth1 -o eth2 -p tcp -d 72.246.0.0/16 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 4

```
iptables -A FORWARD -i eth3 -o eth2 -p tcp -d 69.80.234.0/24 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 4
iptables -A FORWARD -i eth3 -o eth2 -p tcp -d 206.169.213.0/24 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 4
iptables -A FORWARD -i eth3 -o eth2 -p tcp -d 72.246.0.0/16 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 4
iptables -A FORWARD -i eth4 -o eth2 -p tcp -d 69.80.234.0/24 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 4
iptables -A FORWARD -i eth4 -o eth2 -p tcp -d 206.169.213.0/24 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 4
iptables -A FORWARD -i eth4 -o eth2 -p tcp -d 72.246.0.0/16 -m mac --mac-source ! 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 4
```

REGLAS PARA MARCADO DE TRÁFICO DE USUARIOS ESPECIALES

```
iptables -A FORWARD -i eth1 -o eth2 -m mac --mac-source 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 5
iptables -A FORWARD -i eth3 -o eth2 -m mac --mac-source 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 5
iptables -A FORWARD -i eth4 -o eth2 -m mac --mac-source 00:0f:66:ed:65:de -t mangle -j MARK --set-mark 5
iptables -A FORWARD -i eth1 -o eth2 -m mac --mac-source 00:1f:5b:c6:e8:d0 -t mangle -j MARK --set-mark 5
iptables -A FORWARD -i eth3 -o eth2 -m mac --mac-source 00:1f:5b:c6:e8:d0 -t mangle -j MARK --set-mark 5
iptables -A FORWARD -i eth4 -o eth2 -m mac --mac-source 00:1f:5b:c6:e8:d0 -t mangle -j MARK --set-mark 5
```

El marcado se ha realizado en la cadena FORWARD de la tabla MANGLE, ya que el paquete únicamente se reenvía y no hay ninguna acción sobre él. Es el GW0 quien se encarga de accionar sobre los paquetes. La tabla mangle es la requerida para establecer marcas a los paquetes.

En Segundo lugar, se debe **Construir la Jerarquía de Clases** que caracterizan el comportamiento del tráfico, para este propósito y en este caso, se estima conveniente mezclar las disciplinas de cola HTB, PRIO y SFQ. (*Ver capítulo 1. Disciplinas de Cola*).

Según los requerimientos, se ha determinado que se requieren seis tipos de tráfico, entre los cuales serán repartidos los paquetes, y serán representados en clases:

- Tráfico Web. Servicio prioritario que requiere suficiente ancho de banda para funcionar de modo fluido.
- Tráfico para los servicios locales corriendo en la red, como una segunda prioridad.
- Tráfico para el servicio MSN por MAC permitidos, con una prioridad baja. Para el trabajo interno de la UNI no es un servicio indispensable.
- Tráfico para videos a través de la Web. Este servicio tiene una prioridad media. Por ser tráfico de ráfagas, requiere bajos retardos y no debe ser de última prioridad.
- Tráfico para usuarios especiales. Con alta prioridad.
- Tráfico menos prioritario al que se le debe garantizar menos ancho de banda donde se concentra el tráfico no clasificado (es la clase donde va el tráfico por defecto).

La estructura jerárquica de clases sería la mostrada en la siguiente figura, Figura 18. *Estructura jerárquica de clases*:

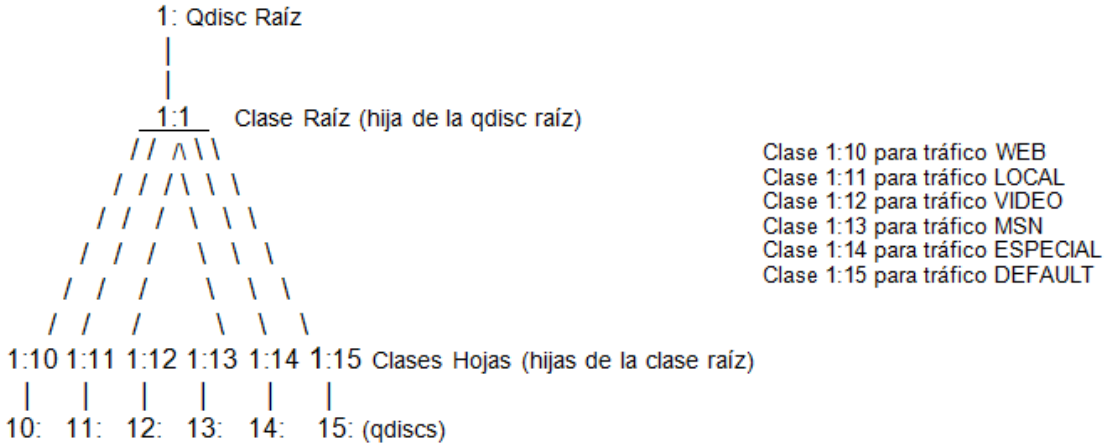


Figura 18. Estructura jerárquica de clases

Para el GW8 se dispone de 3Mbit total de ancho de banda. La distribución del uso de ancho de banda, se ha realizado en base a las decisiones de la administración. Ver *Anexo 3. Distribución del Ancho de Banda.*

Las reglas para crear las clases y las colas son escritas de la siguiente manera:

Qdisc Raíz
tc qdisc add dev eth2 root handle 1: htb default 15

Clase Raíz
tc class add dev eth2 parent 1: classid 1:1 htb rate 3Mbit ceil 3Mbit

Clases Hojas

Tráfico WEB
tc class add dev eth2 parent 1:1 classid 1:10 htb rate 900kbit ceil 3Mbit prio 0

Tráfico LOCAL
tc class add dev eth2 parent 1:1 classid 1:11 htb rate 150kbit ceil 3Mbit prio 1

Tráfico de VIDEO
tc class add dev eth2 parent 1:1 classid 1:12 htb rate 150kbit ceil 3Mbit prio 2

Tráfico MSN (MAC permitidos)
tc class add dev eth2 parent 1:1 classid 1:13 htb rate 150kbit ceil 3Mbit prio 3

Tráfico de usuarios ESPECIALES
tc class add dev eth2 parent 1:1 classid 1:14 htb rate 1500kbit ceil 3Mbit prio 0

Tráfico DEFAULT
tc class add dev eth2 parent 1:1 classid 1:15 htb rate 150kbit ceil 3Mbit prio 4

Qdiscs

Es necesario crear colas bajo las clases creadas:

```
tc qdisc add dev eth2 parent 1:10 handle 10: sfq
tc qdisc add dev eth2 parent 1:11 handle 11: sfq
tc qdisc add dev eth2 parent 1:12 handle 12: sfq
tc qdisc add dev eth2 parent 1:13 handle 13: sfq
tc qdisc add dev eth2 parent 1:14 handle 14: prio
tc qdisc add dev eth2 parent 1:15 handle 15: sfq
```

La clase de *usuarios especiales* (handle 14) ha sido unida a una qdisc PRIO, ya que en esta clase puede fluir todo tipo de tráfico incluyendo tráfico interactivo, para el cual se considera una mejor opción la qdisc PRIO.

En tercer lugar, se debe asignar el tráfico marcado a las clases creadas, esto se logra **asociando las marcas con las clases**:

```
tc filter add dev eth2 protocol ip parent 1:  handle 1 fw classid 1:10
tc filter add dev eth2 protocol ip parent 1:  handle 2 fw classid 1:11
tc filter add dev eth2 protocol ip parent 1:  handle 3 fw classid 1:12
tc filter add dev eth2 protocol ip parent 1:  handle 4 fw classid 1:13
tc filter add dev eth2 protocol ip parent 1:  handle 5 fw classid 1:14
```

Para clasificar el tráfico, en base a la marca del paquete, se ha usado el modificador **fw,** también sería posible usar el clasificador **u32** del comando **tc,** que permite hacer clasificaciones en base a la cabecera IP, pero es más complejo y tedioso, además iptables ofrece mayores posibilidades de marcado que los filtros tc.

Es importante aclarar, que en este documento solo se escriben algunas reglas de control de tráfico, por ser similares entre sí, además, en este momento, el objetivo es explicar cómo se construye la solución. Por lo que se proporciona el script en anexo y las reglas completas en formato digital adjunto al documento. Ver *Anexo 4. Script para generación de las Reglas de Control de Tráfico Propuestas para el Gateway 8*.

Una vez que el script ha sido completado, se requiere guardarlo en un archivo, por ejemplo *reglas-GW8.txt*, y luego ejecutarlo desde la consola de Linux para que las reglas de control de tráfico que han sido escritas anteriormente, se pongan en funcionamiento. Se deberá realizar lo siguiente:

Si las reglas han sido ingresadas a través de una terminal, ejecutar los comandos:

```
# service iptables save
# service iptables restart
```

Si las reglas han sido escritas en un archivo script, realizar lo siguiente:

- Asignar privilegios de ejecución al archivo de configuración

  ```
  # chmod u+x reglas-GW8.txt
  ```

- Ejecutar el archivo de configuración

  #. / reglas-GW8.txt

- Ejecutar los comandos

  # service iptables save
  # service iptables restart

- Obtener estadísticas de las reglas de control de tráfico en funcionamiento

  Para mostrar el marcado de paquetes: # iptables -t TABLE -L y iptables-save -t TABLE -L
  Para mostrar las qdiscs: # tc [-s | -d] qdisc show [dev DEV]
  Para mostrar las clases: # tc [-s | -d] class show dev DEV
  Para mostrar los filtros: # tc filter show dev DEV

En el *Anexo 5. Análisis de las Estadísticas del Tráfico,* se hace un análisis del comportamiento del tráfico de la red en el Gateway 8, una vez que han sido aplicadas las reglas de control de tráfico proporcionadas en esta solución.

Consideraciones para el Futuro

A continuación, se mencionan algunas consideraciones, basadas en las opiniones de los expertos [Ref. b], para ser analizadas según el criterio del administrador de la red.

1. **Para el control del tráfico P2P**

   a. Layer 7

   Para controlar el tráfico P2P desde la capa de aplicaciones, es recomendable usar Layer7. Layer 7 es un clasificador para Netfilter de Linux que identifica paquetes basados en la capa de aplicación. La ventaja es que se puede controlar aplicaciones como Skype (que es un software inteligente y hace uso de los puertos http / https) a pesar de su puerto, ya que el control se hace a nivel de la capa superior, antes de permitirle al paquete llegar a la capa de red y que pueda tomar puertos de acceso que se encuentran abiertos.

   El tráfico Skype podría ser marcado o bloqueado directamente de la siguiente manera:

   ```
   iptables -t mangle -A FORWARD -i eth1 -p tcp -m layer7 --l7proto skypeout -j MARK --set-mark 5

   iptables -A FORWARD -m layer7 --l7proto skypeout -j DROP

   iptables -A FORWARD -m layer7 --l7proto skypetoskype -j DROP
   ```

   b. Cola ESFQ

   Cuando se permite el paso de tráfico P2P, se recomienda usar la disciplina de cola ESFQ en la clase a la que se asocia este tipo de tráfico. Con ESFQ es posible el reparto equitativo del ancho de banda en base a la dirección IP de la conexión, lo que es útil para controlar los programas P2P, que “se aprovechan” al crear múltiples conexiones.

Para hacer uso de layer7 y ESFQ, se necesita de las siguientes herramientas:

- Parche de Kernel e Iproute, para hacer uso de ESFQ. Disponible en el sitio http://fatooh.org/esfq-2.6/
- Layer7: Parche de kernel e iptables, útil para identificar protocolos P2P en base al contenido de los paquetes. Disponible en el sitio http://l7-filter.sourceforge.net/.

Por tanto, las reglas escritas con Layer7 y ESFQ solo funcionan si el kernel es compilado para soporte de estas herramientas.

Parchar y compilar el kernel está fuera de los objetivos de este estudio, por lo que se recomienda el sitio http://www.insflug.org/COMOs/Kernel-Como/Kernel-Como.html para mayor información.

La realización de parche y compilaciones al kernel y las herramientas implica:

- La posibilidad de tiempos más largos de caída del servicio en el Gateway.
- Servidor con instalaciones más “sucias”, por lo que, las actualizaciones de software pueden resultar más complejas.
- Probablemente, a cada actualización del kernel, habrá que aplicarle los parches, ya que algunos parches funcionan con una versión, pero no con otra.

Por estos motivos es importante el análisis y la decisión del administrador de la red, respecto a la ejecución de parches.

2. Para el control del tráfico de entrada

Para un control óptimo del tráfico de la red, es recomendable controlar el tráfico de salida (egress) y el tráfico de entrada (ingress), sin embargo, el control del tráfico entrante aún no ha sido documentado y no es posible hacer un control como en el caso del tráfico de salida. Cabe mencionar y aclarar, que en el presente estudio, se ha realizado un control del tráfico de salida.

No es posible limitar el tráfico entrante, al menos de forma inmediata, por ejemplo, es posible controlar lo que se envía a internet, pero no lo que llega de internet.

Según los expertos en control de tráfico, hay una manera de limitar este tráfico. Usando el dispositivo virtual de encolado IMQ (*Ver capítulo I. Control de Tráfico en Linux. El dispositivo intermedio de encolado IMQ*)**,** es posible limitar este tráfico, sin embargo, esto es un truco y no se trata de un sistema perfecto.

Con IMQ, por ejemplo, se puede limitar las descargas HTTP a 128kbit:

```
iptables -t mangle -A POSTROUTING -o eth2 -j IMQ --todev 0
```
Este comando traslada todo el tráfico entrante al dispositivo IMQ

```
tc qdisc add dev imq0 handle 1: root htb default 1
tc class add dev imq0 parent 1: classid 1:1 htb rate 128kbit
tc class add dev imq0 parent 1:1 classid 1:11 htb rate 128kbit
```
Crea qdisc y clases para IMQ con tope 128kbit

```
iptables -t mangle -A PREROUTING -i imq0 -p tcp --sport 80 -j MARK --set-mark 1
```
Marca el tráfico HTTP

```
tc filter add dev imq0 parent 1:0 protocol ip handle 1 fw classid 1:11
```
Asocia la marca a la clase

Para hacer uso de IMQ, se necesita el siguiente parche.

- Parche de kernel e iptables para usar IMQ. Disponible en el sitio http://www.linuximq.net/, y las instrucciones de instalación en http://wiki.nix.hu/cgi-bin/twiki/view/IMQ/HowToInstall

VII. Conclusiones

La propuesta de Control de Tráfico, proporcionada a través del presente estudio, para el uso de la administración de la red informática UNI, contempla los pasos y parámetros necesarios a considerar para realizar control de tráfico diferenciado en Linux y los aplica en la práctica, a través del caso de estudio particular de la red UNI, tomando como base los requerimientos y topología de la misma.

En el caso de estudio, se ha logrado clasificar el tráfico y aplicarle un tratamiento diferenciado por tipos, según su nivel de importancia para los usuarios en la red. Esto significa, que se ha realizado control de tráfico diferenciado.

Finalmente, se puede decir que las reglas de control de tráfico propuestas en este documento, están listas para ser utilizadas por la administración de la red, y además se proporciona la documentación técnica suficiente, que ha sido utilizada para la elaboración de dicha propuesta.

VIII. Recomendaciones

- Analizar las “Consideraciones para el Futuro”, planteadas en el caso de estudio, para valorar la posibilidad de realizar parches al núcleo del sistema operativo y a las herramientas de control de tráfico, a fin de emplear la potencialidad proporcionada por éstos, en el control del tráfico.
- Poner en funcionamiento la solución propuesta en el presente estudio y expandirla a los Gateway restantes, lo que beneficiaría directamente el desempeño de la red, al hacerla más eficiente.
- Recurrir a la documentación técnica proporcionada en este documento, antes de realizar cualquier cambio en las reglas de control de tráfico propuestas.

IX. Referencia Bibliográfica

Metodología de la Investigación
Mc Graw Hill Interamericana de México, S.A de C.V. Primera Edición. 1991
ISBN: 968-422-931-3
Baptista Lucio Pilar. Fernández Collado Carlos. Hernández Sampieri Roberto.
P. 58-81. P. 107-114

Referencias Web:

a. http://lartc.org/lartc.pdf
Linux Advanced Routing & Traffic Control HOWTO.
Capítulo 9. Disciplinas de cola (qdiscs) para gestión de ancho de banda.

b. http://usuarios.lycos.es/ccd_illusions/QoS-3.pdf
Implantación de QoS en un entorno GNU/Linux. Bruno Herrero (mines) Octubre 2006
Introducción
Cómo Funciona QoS en Linux
Cómo usar las disciplinas de cola para clasificar el tráfico
Implantando QoS En Linux
Regular el tráfico de entrada

c. http://securepoint.com/lists/html/LARTC/2007-05/msg00077.html
[LARTC] PRIO and TBF is much better than HTB?

d. Tesis de Maestría en Telemática. Controlador de Ancho de Banda
Ing. Diego Fernando Navarro. Cap. 6.4, 6.5 y 7.3.

e. http://almacen.gulic.org/03_www/comos/LARTC/html/x660.html
Disciplinas de Cola Simples, sin clases

f. http://linux-ip.net/articles/Traffic-Control-HOWTO/classless-qdiscs.html
Classless Queuing Disciplines (Qdiscs)

g. http://www.davidfv.net/articulos/trabajo_redes.pdf
Implementación de IPv6, QoS e IPSec con Linux
Cristina Duran Sanles. Manuel David Fernández Vaamonde. 27 de mayo de 2003
Pág.18.

h. http://www.opalsoft.net/qos/DS-29.htm
DSMARK queuing discipline

i. http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm
HTB Linux queuing discipline manual - user guide.
Martin Devera aka devik (devik@cdi.cz)
Manual: devik and Don Cohen
Last updated: 5.5.2002

j. http://linux.die.net/man/8/tc-htb
tc-htb(8) – Linux man page

k. http://linux-ip.net/articles/Traffic-Control-HOWTO/classful-qdiscs.html
Classful Queuing Disciplines (Qdiscs)

l. http://www.docum.org/docum.org/faq/cache/10.html
HTB shapping rules

m. http://www.efn.uncor.edu/escuelas/computacion//files/Manual%20de%20Uso%20de%20IPTables%20-%20Jorge%20Kleinerman.pdf
Comandos de IPtables. Manual de uso de IPTables – Jorge E. Kleinerman
Condiciones de Marca.

n. http://www.linuxtopia.org/Linux_Firewall_iptables/index.html
Linux Packet Filtering and iptables
Chapter 6. Traversing of tables and chains

o. http://www.davidfv.net/articulos/trabajo_redes.pdf
Implementación de IPv6, QoS e IPSec con Linux
Cristina Durán Sanlés / Manuel David Fernández Vaamonde
QoS en Linux.

p. http://www.rns-nis.co.yu/~mps/linux-tc.html
tc - traffic control
Linux QoS control tool. Milan P. Stanic. mps@rns-nis.co.yu
What is QoS
Command syntax
Classes
Filters (or classifier)

q. http://www.opalsoft.net/qos/DS-210.htm
TCINDEX Classifier

r. http://es.wikipedia.org/wiki/Netfilter
Netfilter/Iptables
Historia
Tablas
Destinos de Reglas

s. http://stuff.gpul.org/2004_jornadas/doc/iptables.pdf
IV Jornadas Sistema Operativo Linux. Cortafuegos en Linux con Iptables. Andrés J. Diaz
Reglas de Cortafuego.

t. http://dns.bdat.net/documentos/cortafuegos/x89.html
Cortafuegos para redes. Filtrado de paquetes.
Esquemas de Filtrado.

u. RFC 1157

v. http://greco.dit.upm.es/~david/TAR/trabajos2002/05-Contro-Trafico-Linux-Fernando-David-Gomez-res.pdf
Control de Tráfico Utilizando Linux
Introducción
Posibilidades de Linux dentro del ámbito del networking
Introducción a iproute2

X. Glosario de Términos

Dhcp - Dynamic Host Configuration Protocol - Protocolo de Configuración Dinámica de Anfitrión: es un protocolo de red que permite a los nodos de una red IP obtener sus parámetros de configuración automáticamente. Se trata de un protocolo de tipo cliente/servidor en el que generalmente un servidor posee una lista de direcciones IP dinámicas y las va asignando a los clientes conforme éstas van estando libres, sabiendo en todo momento quién ha estado en posesión de dicha IP, cuánto tiempo la ha tenido y a quién se la ha asignada después.

DiffServ, Differentiated Services: es un protocolo QoS (Quality of service, calidad de servicio) que prioriza paquetes provenientes del servicio de VoIP frente a los demás para asegurar una buena calidad de voz, aún cuando el trafico de red es alto.

Dirección MAC - Media Access Control Address - Dirección de Control de Acceso al Medio: es un identificador de 48 bits (6 bytes) que corresponde de forma única a una tarjeta o interfaz de red. Es individual, cada dispositivo tiene su propia dirección MAC determinada y configurada por el IEEE (los últimos 24 bits) y el fabricante (los primeros 24 bits). No todos los protocolos de comunicación usan direcciones MAC, y no todos los protocolos requieren identificadores globalmente únicos. Las direcciones MAC son únicas a nivel mundial, puesto que son escritas directamente, en forma binaria, en el hardware en su momento de fabricación, debido a esto, las direcciones MAC son a veces llamadas Direcciones Quemadas" (BIA, por las siglas de Burned-in Address) o dirección física.

Dnscache: almacena datos de configuración para el DNS, atrapando consultas de resolución de los clientes DNS. Este servicio ahorra el tiempo de respuesta del DNS, de manera que no se tiene que repetir consultas por la misma información.

DSL - Digital Subscriber Line – Línea de Abonado Digital: es un conjunto de normas, para la conexión de red de banda ancha, sobre líneas telefónicas normales.

Gateway – Puerta de Enlace: es un dispositivo que permite interconectar redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación. Su propósito es traducir la información del protocolo utilizado en una red, al protocolo usado en la red de destino. Es normalmente un equipo informático configurado para hacer posible a las máquinas de una red local (LAN) conectadas a él, el acceso hacia una red exterior, generalmente realizando para ello, operaciones de traducción de direcciones IP (NAT: Network Address Translation). Esta capacidad de traducción de direcciones permite aplicar una técnica llamada IP Masquerading (enmascaramiento de IP), usada muy a menudo para dar acceso a Internet a los equipos de una red de área local, compartiendo una única conexión a Internet, y por tanto, una única dirección IP externa.

Hash: una función o método para generar claves o llaves que representen de manera casi unívoca a un documento, registro, archivo, etc. Una propiedad fundamental del hashing es que si dos resultados de una misma función son diferentes, entonces las dos entradas que generaron dichos resultados también lo son. Sin embargo, es posible que existan claves resultantes iguales para objetos diferentes, ya que el rango de posibles claves es mucho menor que el de posibles objetos a resumir.

Iftop: programa de Linux usado para escuchar el tráfico de la red en una interface nombrada y muestra una tabla del uso de ancho de banda actual por pares de Host. Está incluido en la distribución Linux Debian.

Iproute2: es un paquete de utilidades desarrollado por Alexey Kuznetsov. Incluye un conjunto de herramientas muy potentes para administrar interfaces de red y conexiones en sistemas Linux. Este paquete reemplaza completamente las funcionalidades presentes en ifconfig, route y arp y las extiende llegando a tener características similares a las provistas por dispositivos exclusivamente dedicados al ruteo y control de tráfico. Iproute2 está incluido en distribuciones de Debian y RedHat con versiones del núcleo mayores a 2.2.

IPTV - Internet Protocol Television: se ha convertido en la denominación más común para los sistemas de distribución por subscripción de señales de televisión y/o vídeo

usando conexiones de banda ancha sobre el protocolo IP. A menudo se suministra junto con el servicio de conexión a Internet, proporcionado por un operador de banda ancha sobre la misma infraestructura, pero con un ancho de banda reservado.

Jitter: son oscilaciones de la separación temporal entre paquetes. En aplicaciones que requieren sincronización (videoconferencia, sincronizar audio con vídeo), es muy importante que esas oscilaciones sean pequeñas.

Kernel: se puede definir como el corazón del sistema operativo. Es el encargado de que el software y el hardware del ordenador puedan trabajar juntos. Sus funciones más importantes son:

- Administración de la memoria para todos los programas y procesos en ejecución.
- Administración del tiempo de procesador que los programas y procesos en ejecución utilizan.
- Permitir acceso a los periféricos / elementos del ordenador de una manera cómoda.

Latencia: suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red. Otros factores que influyen en la latencia de una red son:

- El tamaño de los paquetes transmitidos.
- El tamaño de los buffers dentro de los equipos de conectividad. Ellos pueden producir un Retardo Medio de Encolado.

MTU – Maximum Transfer Unit - Unidad Máxima de Transferencia: expresa el tamaño en bytes de la unidad de datos más grande que se puede transmitir por unidad, a través de una red física, usando un protocolo de internet. Su valor es 1500.

NAT - Network Address Translation - Traducción de Dirección de Red: es un mecanismo utilizado por routers IP para intercambiar paquetes entre dos redes que se asignan direcciones incompatibles mutuamente. Consiste en convertir en tiempo real las direcciones utilizadas en los paquetes transportados. Un router NAT cambia la dirección

origen en cada paquete de salida y, dependiendo del método, también el puerto origen para que sea único. Estas traducciones de dirección se almacenan en una tabla, para recordar qué dirección y puerto le corresponde a cada dispositivo cliente y así saber donde deben regresar los paquetes de respuesta. Si un paquete que intenta ingresar a la red interna no existe en la tabla de traducciones, entonces es descartado.

Netfilter: es un framework disponible en el núcleo Linux que permite interceptar y manipular paquetes de red. Dicho framework permite realizar el manejo de paquetes en diferentes estadios del procesamiento. Netfilter es también el nombre que recibe el proyecto que se encarga de ofrecer herramientas libres para cortafuegos, basadas en Linux.

P2p – peer to peer - par a par o punto a punto: se refiere a una red que no tiene clientes ni servidores fijos, sino una serie de nodos que se comportan simultáneamente como clientes y como servidores respecto de los demás nodos de la red. Son redes que aprovechan, administran y optimizan el uso de banda ancha que acumulan de los demás usuarios en una red por medio de la conectividad entre los mismos usuarios participantes de la red, obteniendo como resultado mucho más rendimiento en las conexiones y transferencias que con algunos métodos centralizados convencionales, donde una cantidad relativamente pequeña de servidores provee el total de banda ancha y recursos compartidos para un servicio o aplicación. Las redes peer to peer se usan muy a menudo para compartir toda clase de archivos que contienen: audio, video, telefonía VoIP, texto, software y datos en cualquier formato digital.

Round- Robin: es uno de los algoritmos de planificación de procesos más simples dentro de un sistema operativo. Asigna a cada proceso una porción de tiempo equitativa y ordenada, tratando a todos los procesos con la misma prioridad. En este tipo de planificación, cada proceso tiene asignado un quantum de tiempo para ejecutarse y en el caso de que no pueda terminar la ejecución en su quantum, el proceso pasa de nuevo a la cola de procesos para ser ejecutado por otro quantum, luego de recorrer la cola para asegurarse que todos los procesos reciban ese quantum de procesamiento.

Router – enrutador - ruteador - encaminador: es un dispositivo de hardware para interconexión de redes de ordenadores que opera en la capa tres (nivel de red). Este dispositivo permite asegurar el enrutamiento de paquetes entre redes o determinar la ruta que debe tomar el paquete de datos.

RSVP - Resource Reservation Protocol: es un protocolo de señalización de reservas encargado de transportar especificaciones de tráfico, peticiones de reservas y disponibilidad de recursos. Se utiliza para garantizar la calidad de servicio, QoS, por medio de la reserva de ancho de banda para flujos de datos con capacidad para RSVP en todos los nodos de acceso a datos.

Scheduler: mecanismo de ordenar o desordenar los paquetes antes de ser desencolados de una cola. Los mecanismos de scheduling tratan de compensar varias condiciones de red, por ejemplo, prevenir que un solo flujo se apodere del uso de la red.

Snmp - Protocolo Simple de Administración de Red: es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red. Es parte de la familia de protocolos TCP/IP y permite a los administradores supervisar el desempeño de la red, planear su crecimiento, buscar y resolver sus problemas. SNMP en su última versión (SNMPv3) posee cambios significativos con relación a sus predecesoras, sobre todo en aspectos de seguridad, sin embargo no ha sido mayoritariamente aceptado en la industria.

Snort_inline: es básicamente una versión modificada de snort, que acepta paquetes desde iptables e ipfw vía libipq (linux) o divert sockets (FreeBSD), en lugar de libpcap. Después usa nuevos tipos de reglas (drop, sdrop, reject) para decirle a iptables/ipfw si el paquete debería ser borrado, rechazado, modificado o permitirle pasar basado en un conjunto de reglas snort. Snort inline se debe pensar como un Instrusion Prevention System (IPS) que usa un sistema de detección de Intrusos (IDS) para hacer decisiones respecto a los paquetes que atraviesan snort inline.

Squid: es un tipo de web caché. Es un popular programa de software libre que implementa un servidor proxy y un demonio para caché de páginas web, publicado bajo licencia GPL. Tiene una amplia variedad de utilidades, desde acelerar un servidor web, guardando en caché peticiones repetidas a DNS y otras búsquedas para un grupo de gente que comparte recursos de la red, hasta caché de web, además de añadir seguridad filtrando el tráfico. Está especialmente diseñado para ejecutarse bajo entornos tipo Unix. Squid ha sido desarrollado durante muchos años y se le considera muy completo y robusto. Aunque es orientado principalmente a HTTP y FTP, es compatible con otros protocolos como Internet Gopher e implementa varias modalidades de cifrado como TLS, SSL y HTTPS.

Ssh - Secure Shell - Intérprete de comandos seguro: es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora mediante un intérprete de comandos. Además de la conexión a otras máquinas, SSH permite copiar datos de forma segura (tanto ficheros sueltos como simular sesiones FTP cifradas), gestionar claves RSA para no escribir claves al conectar a las máquinas y pasar los datos de cualquier otra aplicación por un canal seguro tunelizado mediante SSH.

Tabla de enrutamiento: es un documento electrónico que almacena las rutas a los diferentes nodos en una red informática. Los nodos pueden ser cualquier tipo de aparato electrónico conectado a la red. Generalmente se almacena en un router o en la red en forma de una base de datos o archivo. Cuando los datos deben ser enviados desde un nodo a otro de la red, la tabla de enrutamiento se hace referencia con el fin de encontrar la mejor ruta para la transferencia de información. La tablas de enrutamiento para todos los dispositivos de red nunca cambian, a menos que y hasta que el administrador de la red haga los cambios manualmente.

Tethereal: es un analizador de protocolos de red que permite capturar datos de paquetes de una red activa o leer paquetes de archivos previamente salvados con el formato de libpcap. Tethereal captura los paquetes en el formato de libpcap, el cual también es el formato usado por la herramienta tcpdump.

TOS - Type of Service: se refiere a 8 bits reservados en la cabecera IP de un paquete, los cuales pueden ser divididos en cinco subcampos: Los *bits de 0-2,* que son los bits de precedencia e indican la importancia de un datagrama, siendo cero el mejor valor. El *bit 3,* que representa Solicitud de bajo retardo (D). El *bit 4,* que representa Solicitud de alto tráfico de red (T). El *bit 5,* que representa Solicitud de alta confiablidad (R). Los *bits 6 y 7* que representan el campo ENC (Explicit Congestion Notification) y es reservado para uso futuro.

Tráfico de ráfagas: cuando una aplicación produce paquetes de longitud variable y se generan en instantes de tiempo aleatorio, separado por intervalos de silencio de duración aleatoria, generalmente larga con respecto a la duración de los paquetes; el tráfico que posee estas características se denomina tráfico a ráfagas. Las aplicaciones de transferencia de bloques en tiempo real son un ejemplo representativo de este tipo de tráfico. Los instantes en que se envían los bloques son impredecibles y el tamaño de los mismos suele ser variable.

Tráfico de red: volumen de información que fluye a través de las redes de datos.

Tráfico interactivo: volumen de información que fluye de manera interactiva en ambas direcciones de la comunicación. Por ejemplo, el tráfico generado por los programas de chats.

Transceiver - Transductor: dispositivo que recibe la potencia de un sistema mecánico, electromagnético o acústico y lo transmite a otro, generalmente en forma distinta. El micrófono y el altavoz son ejemplos de transductores. En comunicaciones, es un transmisor/receptor de señales de radio frecuencia (RF) que sirve para conectar aparatos por vía inalámbrica. El término es aplicado a dispositivos de comunicaciones inalámbricas tales como teléfonos celulares, radios de mano de 2 vías, dispositivos móviles de dos vías. Algunos transceivers son diseñados para permitir recepción de señales durante el periodo de transmisión, este modo de operación es conocido como

full dúplex, y requiere que el transmisor y receptor opere en sustancialmente diferentes frecuencias para que las señales de transmisión no interfieran con la recepción.

Tshark: es un analizador de protocolos de red. Permite capturar paquetes de datos de una red activa o lee paquetes de un archivo de captura previa, ya sea imprimiendo una forma decodificada de estos paquetes a la salida Standard o escribiendo los paquetes a un archivo. TShark captura los paquetes en el formato de libpcap, el cual también es el formato usado por la herramienta tcpdump.

TTL - Time To Live - Tiempo de Vida: es un concepto usado en redes de computadores para indicar por cuántos nodos puede pasar un paquete antes de ser descartado por la red o devuelto a su origen. Es un campo en la estructura del paquete del protocolo IP, sin este campo, paquetes enviados a través de rutas no existentes, o a direcciones erróneas, estarían vagando por la red de manera infinita, utilizando ancho de banda sin una razón positiva. Es utilizado en el paquete IP de manera que los routers puedan analizarlo y actuar según su contenido. Si un router recibe un paquete con un TTL igual a uno o cero, no lo envía a través de sus puertos, sino que notifica vía ICMP a la dirección IP origen que el destino se encuentra "muy alejado". Si un paquete es recibido por un router que no es el destino, éste decrementa el valor del TTL en uno y envía el paquete al siguiente router.

VoIP - Voz sobre Protocolo de Internet: es un grupo de recursos que hacen posible que la señal de voz viaje a través de Internet empleando el protocolo IP. Esto significa que se envía la señal de voz en forma digital en paquetes, en lugar de enviarla (en forma digital o analógica) a través de circuitos utilizables sólo para telefonía, como una compañía telefónica convencional o PSTN (Public Switched Telephone Network, Red Telefónica Pública Conmutada). El tráfico de Voz sobre IP puede circular por cualquier red IP, incluyendo aquellas conectadas a Internet, por ejemplo redes de área local (LAN).

Webcaché: es un servicio de caché que almacena documentos web (es decir, páginas, imágenes, etc) para reducir el ancho de banda consumido, la carga de los servidores y

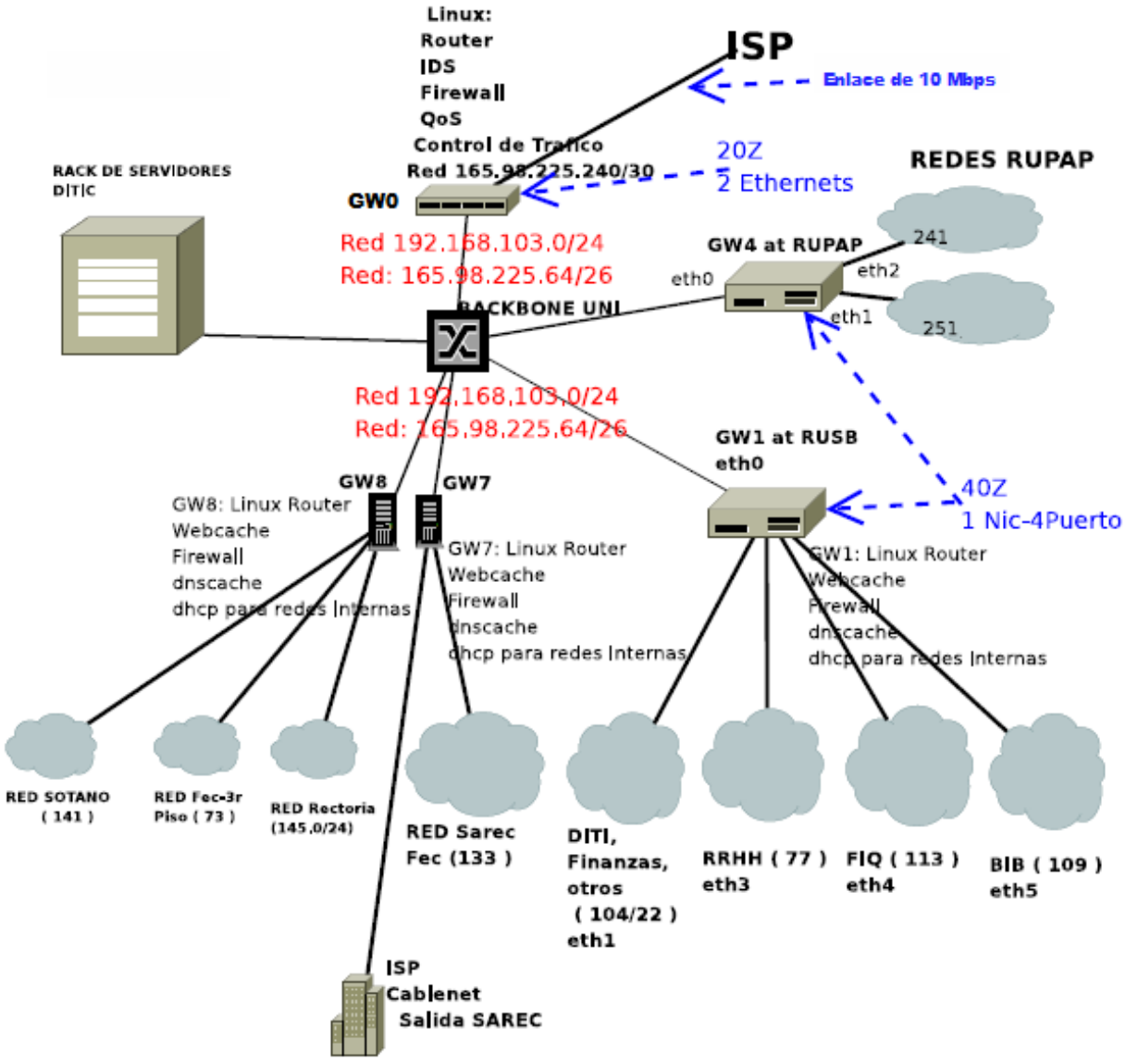
el retardo en la descarga. Un caché web almacena copias de los documentos que pasan por él, de forma que subsiguientes peticiones pueden ser respondidas por el propio caché, si se cumplen ciertas condiciones, en lugar de que la solicitud llegue hasta el servidor.

Webmin: es una herramienta de configuración de sistemas accesible vía web para OpenSolaris, GNU/Linux y otros sistemas Unix. Con él se pueden configurar aspectos internos de muchos sistemas operativos, como usuarios, cuotas de espacio, servicios, archivos de configuración, apagado del equipo, así como modificar y controlar muchas aplicaciones open source, como el servidor web Apache, PHP, MySQL, DNS, Samba, DHCP, entre otros. Está escrito en Perl 5, ejecutándose como su propio proceso y servidor web. Por defecto se comunica a través del puerto TCP 10000, y puede ser configurado para usar SSL si OpenSSL está instalado con módulos de Perl adicionales requeridos. Webmin también permite controlar varias máquinas a través de una interfaz simple, o iniciar sesión en otros servidores webmin de la misma subred o red de área local.

Wireshark (versión mejorada de Ethereal): es un software libre que se ejecuta sobre la mayoría de sistemas operativos Unix y compatibles, incluyendo Linux, Solaris, FreeBSD, NetBSD, OpenBSD, y Mac OS X, así como en Microsoft Windows. Es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones para desarrollo de software y protocolos, y como una herramienta didáctica para educación. Cuenta con todas las características estándar de un analizador de protocolos. Provee una funcionalidad similar a la de tcpdump, pero añade una interfaz gráfica y muchas opciones de organización y filtrado de información. Permite ver todo el tráfico que pasa a través de una red estableciendo la configuración en modo promiscuo. Permite examinar datos de una red viva o de un archivo de captura salvado en disco. También incluye una versión basada en texto llamada tshark.

XI. Anexos

Anexo 1. Mapa Topológico de la red informática de la UNI



Hecho por:
Walter Perez Arauz

Anexo 2. Estadísticas de Red

Máxima demanda de Ancho de Banda

IP	%byte	Nombre de computadora	Comentario	D	T	N	F
68.142.101.55	7.54	cds46.mia.llnw.net	Externo	x			
85.12.58.201	9.34	no tiene nombre	Externo			x	
27.12.10.154	12.53	f262.mail.vip.ukl.yahoo.com	Externo			x	
208.101.34.216	15.78	denizonline.com.34.101.208.in-addr.arpa	Extero	x			x
66.55.141.40	19.12	sin nombre	no se pudo determinar el nombre		x		x
74.53.245.226	20.73	e2.f5.354a.static.theplanet.com	Externo	x			
69.80.237.166	23.77	hosted.by.alphared.com	servidor externo web			x	
74.54.84.202	31.47	ca.54.364a.static.theplanet.com	servidor externo web		x		x
128.107.229.50	31.56	cna-prod-nv.cisco.com	servidor externo web	x			x
68.142.101.93	36.16	cds143.mia.llnw.net	servidor externo web		x		
192.168.103.244	38.62	gateway	Uni	x			x
66.249.83.83	46.20	wx-in-f83-google.com	servidor externo web	x			x
74.54.84.203	48.51	cb.54.364a.static.theplanet.com			x		x
192.168.102.244	50.18	gateway	Uni			x	
85.17.146.9	51.88	hosted-by.leaseweb.com	servidor externo web	x			
68.142.101.13	53.93	cds3.mia.llnw.net	servidor externo web		x		
192.168.141.3	58.68	sotano	Uni	x			x
192.168.102.244	59.52	gateway	Uni	x			
132.156.62.20	86.85	retscreen.net	servidor externo web			x	x
192.168.141.3	90.11	sotano	uni-acceso al lab computación		x		
149.6.120.188	91.51	no tiene nombre		x			
192.168.141.3	91.90	sotano	Uni	x			
192.168.141.3	97.45	sotano	Uni		x		x
192.168.103.244	99.66	gateway				x	x

D: Día. T: Tarde. N: Noche. F: Fin de Semana

La computadora 192.168.141.3 (del sótano) es la que registra más demanda de tráfico, seguido de computadoras con servicios web externos, que son consultados localmente.

Demanda de Servicios

No. Puerto	% bytes	Comentario	D	T	N	F
12241	8.21			x		
net8-cman	9.11	Oracle Net8 CMan Admin		x		
42443	9.56			x		
28521	10.07			x		
16321	10.76			x		
25284	11.86			x		
27444	13.20	aplicacion no asignado		x		x
28474	13.40			x		
7558	13.58			x		
16499	13.98			x		
26103	15.94			x		
27738	20.89		x			
57997	24.27				x	
reftek	26.82			x		
https	31.19	web segura	x			x
ci3-software2	31.50	aplicacion de control remoto		x		x
27534	46.93		x			
LTP	48.02	impresora remota	x			x
17149	48.48	puerto de aplicacion no asignado		x		x
57997	76.98				x	
17681	91.91		x			
http	98.74	servicio web	x			x
http	99.57	servicio web		x		x
http	99.81	servicio web		x		
http	99.83	servicio web	x			
http	99.87	servicio web			x	

D: Día. T: Tarde. N: Noche. F: Fin de Semana

Los puertos de mayor uso se observa que es el puerto 80 (http) con un promedio del 98%, le siguen el puerto 443 (https) con un promedio de 31.19% y el resto de puerto son de aplicaciones.

Datos de Interfaces de Red para el marcado de paquetes

GATEWAY	IP GATEWAY	IP ETHERNET	INTERFAZ ENTRADA	INTERFAZ SALIDA	MARCA	TIPO DE CADENA	UBICACION
GW8	192.168.103.244	192.168.103.244		eht2	1	FORWARD	BACKBONE
		192.168.141.1	eth3				RED SOTANO
		192.168.73.1	eth4				FEC 3ER PISO
		192.168.145.1	eth1				RECTORIA
GW7	192.168.103.220	192.168.103.220		eth2	1	FORWARD	
		192.168.133.1	eth1				RED DE SAREC/FEC
		dinamico	eth0				CABLENET-SALIDA-SAREC
GW1	192.168.103.96	192.168.103.96		eth0			
		192.168.104.1	eth1				DITI, FINANZAS,OTROS
		192.168.77.1	eth3				RRHH
		192.168.113.1	eth4				FIQ
		192.168.109.1	eth5				BIB
GW4	192.168.103.245	192.168.103.245		eth0	1	FORWARD	
		192.168.251.1	eth1				RUPAP
		192.168.241.1	eth2				RUPAP
GW0	192.168.103.1	192.168.103.1	eth0			INPUT	
		165.98.225.242		eth1			
BACKBONE - Switch							
				puertos			
GW8	192.168.103.244		puertos				
GW7	192.168.103.220		puertos				
GW1	192.168.103.96		puertos				
GW4	192.168.103.245		puertos				

Anexo 3. Distribución del Ancho de Banda

La administración de la red interna de la UNI, ha solicitado que se realice la distribución de ancho de banda de la siguiente manera, tomando en cuenta que el enlace total es de 10mbit:

GW8 IP 192.168.103.244 interfaz de salida eht2 Ancho de Banda 3mbit

192.168.141.1 eth3 RED SOTANO -------- 512kbit

192.168.73.1 eth4 FEC 3ER PISO---------512kbit

192.168.145.1 eth1 RECTORIA----------2mbit

GW7 IP 192.168.103.220 interfaz de salida eth2 Ancho de Banda 0.89mbit

192.168.133.1 eth1 RED DE SAREC/FEC----------768kbit

dinámico eth0 CABLENET-SALIDA-SAREC---128kbit

GW1 IP 192.168.103.96 interfaz de salida eth0 Ancho de Banda 3.76mbit

192.168.104.1 eth1 DITI, FINANZAS,OTROS----3mbit

192.168.77.1 eth3 RRHH----------512kbit

192.168.113.1 eth4 FIQ----------128kbit

192.168.109.1 eth5 BIB----------128kbit

GW4 IP 192.168.103.245 interfaz de salida eth0 Ancho de Banda 2.35mbit

192.168.251.1 eth1 RUPAP ------ 1.17mbit

192.168.241.1 eth2 RUPAP ------- 1.17mbit

En base a estos datos, para el Gateway, se ha realizado la siguiente distribución entre servocios:

Ancho de Banda Total: 3 mbit
Para usuarios privilegiados (rectoría) = 1.5mbit
Para usuarios comunes (3er piso y sótano) se distribuyen 1.5mbit entre todos sus servicios, así:

- Tráfico web = 60% - 900kbit
- Tráfico local = 10% - 150kbit
- Tráfico de video = 10% - 150kbit
- Tráfico msn = 10% - 150kbit
- Tráfico no clasificado = 10% - 150kbit

Anexo 4. Script para generación de las Reglas de Control de Tráfico Propuestas para el Gateway 8.

El siguiente script está escrito en el lenguaje Perl, y al ejecutarlo genera el conjunto de reglas de control de tráfico a aplicar en el Gateway 8. Tanto el sript como las reglas mismas son adjuntas a este documento en formato digital.

Si desea ejecutar el script, envíe la siguiente instrucción desde la consola:

>perl script-reglas-GW8.pl > reglas-GW8.txt

Donde script-reglas-GW8.pl es el nombre del script y reglas-GW8.txt es el archivo que contendrá el conjunto de reglas generadas.

A continuación el script:

```
print "#------------------Script de reglas de control de trafico para el GW8--------------\n";

#----DECLARACION DE VARIABLES que guardan los numero de puerto de los servicios-----

@WEB_TCP=("80","443","3128");

@WEB_UDP=("80","3128");

@LOCAL_TCP=("53","22","161","10000");

@LOCAL_UDP=("53","22","67","68");

@VIDEO_RED=("69.80.234.0/24","206.169.213.0/24","72.246.0.0/16","202.105.182.0/24","81.95.10.0/24","85.17.0.0/16","208.167.0.0/16");

@VIDEO_RANGO=("64.15.112.0-64.15.127.0","208.117.224.0-208.117.251.0","208.117.253.0-208.117.255.0");

@MSN_TCP=("1863","6891:6901","28800:29000");

@MSN_UDP=("1863","6901");

#----------------------------Lista de numeros MAC permitidos a usar MSN----------------------------

@MAC_MSN=("00:08:54:1f:7d:22","00:12:79:be:42:f5","00:0f:1f:d8:d8:65","00:12:3f:f1:b1:53","00:e0:4c:a0:15:7b","00:a0:d1:39:71:33","00:0a:5e:4b:f4:6f","00:0f:1f:d9:72:09","00:0f:b0:67:25:c1","00:11:f5:58:81:f1","00:06:4f:0e:a2:af","00:c0:9f:45:2c:d1","00:0e:35:3d:bd:60","00:21:5d:96:16:88","00:1c:7e:0b:28:b0","00:05:5d:a7:26:2a","00:0b:5d:47:4a:4f","00:13:8f:18:ce:72","00:0a:5e:52:a5:72","00:12:17:86:d5:
```

```
a6","00:19:d1:62:c8:97","00:19:d1:62:c8:8a","00:1d:e0:ab:b5:41","00:e0:7d:a9:59:e3","00:14:d1:30:70:a
e","00:1b:9e:D0:1a:2a","00:08:a1:61:95:d0","00:01:6c:fb:1b:36","00:0b:6a:89:4d:47","00:0a:5e:4b:f4:67
","00:11:f5:78:f0:3f","00:12:79:be:5a:1e","00:30:05:bd:48:4a","00:e0:7d:9e:39:99","00:30:05:be:1c:ee","
00:12:17:a0:72:21","00:19:bb:60:79:6c","00:13:02:c2:45:42","00:16:76:7a:81:34","00:01:66:fb:1b:36","0
0:90:4b:e6:f0:98","00:14:22:c2:6d:35","00:15:c5:d0:c6:49","00:19:7d:3a:6a:22","00:19:bb:4d:3b:24","00
:01:6c:fb:1b:42","00:c0:9f:c1:c8:a5","00:30:05:ba:73:a1","00:13:a9:4e:15:61","00:e0:7d:a4:af:7e","00:08
:0d:08:2a:52","00:0e:35:cf:c7:5f","00:19:bb:60:6c:b8","00:0f:66:ed:5a:58");

#----------------------------Lista de numeros MAC de usuarios especiales----------------------------

@MAC_PRIVIL=("00:0f:66:ed:65:de","00:1f:5b:c6:e8:d0","00:12:17:7c:63:57","00:21:29:6d:12:a5","00:1
6:44:bd:62:0e");

@MAC=@MAC_PRIVIL;

#En mac se suma mac privilegiados con mac_msn

push(@MAC,@MAC_MSN);

#Servicos a Bloquear: No incluye MSN

@BLOQUEAR_TCP=("135","139","445","1214","1900","2422","4660:4663","6699","7777","7880","4662"
,"5222");

@BLOQUEAR_UDP=("135","139","445","1214","1900","2422","4660:4663","6699","7777","7880","4672
","23399");

#Interfaces de entrada

@interfaces=("eth1","eth3","eth4");

#Interfaz de salida

$iSalida="eth2";

#Asignacion de ancho de banda

@CUOTA=("1500kbit","900kbit","150kbit");

$TOTAL="3Mbit";

#-------------------------------------------------------------------------------------------------------

#   MARCADO DE PAQUETES

#-------------------------------------------------------------------------------------------------------

print "#-------------------------------------------------------------------------------------------------\n";
```

```
print "#----------REGLAS PARA BLOQUEO DE TRAFICO TCP y UDP-------------------------\n";

print "#-----------------------------------------------------------------------------------------------\n";

foreach $interfaz (@interfaces){

	# Para todos los interfaces de red, Bloquea todos los servicios a denegar, excepto msn, para todos los usuarios excepto los mac privilegiados

	foreach $macp (@MAC_PRIVIL){

		foreach $p (@BLOQUEAR_TCP){

			print "iptables -A FORWARD -i $interfaz -o $iSalida -p tcp --dport $p -m mac --mac-source ! $macp -t filter -j REJECT \n";

		};

		foreach $p (@BLOQUEAR_UDP){

			print "iptables -A FORWARD -i $interfaz -o $iSalida -p udp --dport $p -m mac --mac-source ! $macp -t filter -j REJECT \n";

		};

	};

	#Bloquea msn para todos los usuarios, excepto los mac permitidos para el uso del servicio (que incluye a los privilegiados)

	foreach $mac (@MAC){

		foreach $p (@MSN_TCP){

			print "iptables -A FORWARD -i $interfaz -o $iSalida -p tcp --dport $p -m mac --mac-source ! $mac -t filter -j REJECT \n";

		};

		foreach $p (@MSN_UDP){

			print "iptables -A FORWARD -i $interfaz -o $iSalida -p udp --dport $p -m mac --mac-source ! $mac -t filter -j REJECT \n";

		};

	};

};
```

```
print "#-----------------------------------------------------------------------------------------\n";

print "#------------------------REGLAS PARA TRAFICO WEB TCP y UDP-----------------------\n";

print "#-------------------Se excluye a usuarios especiales ---------------------------------\n";

foreach $macp (@MAC_PRIVIL){

        foreach $p (@WEB_TCP){

                print "iptables -A FORWARD -i eth1 -o $iSalida -p tcp --dport $p -m mac --mac-source !
$macp -t mangle -j MARK --set-mark 1 \n";

                print "iptables -A FORWARD -i eth3 -o $iSalida -p tcp --dport $p -m mac --mac-source !
$macp -t mangle -j MARK --set-mark 1 \n";

                print "iptables -A FORWARD -i eth4 -o $iSalida -p tcp --dport $p -m mac --mac-source !
$macp -t mangle -j MARK --set-mark 1 \n";

        };

        foreach $p (@WEB_UDP){

                print "iptables -A FORWARD -i eth1 -o $iSalida -p udp --dport $p -m mac --mac-source !
$macp -t mangle -j MARK --set-mark 1 \n";

                print "iptables -A FORWARD -i eth3 -o $iSalida -p udp --dport $p -m mac --mac-source !
$macp -t mangle -j MARK --set-mark 1 \n";

                print "iptables -A FORWARD -i eth4 -o $iSalida -p udp --dport $p -m mac --mac-source !
$macp -t mangle -j MARK --set-mark 1 \n";

        };

};

print "#-----------------------------------------------------------------------------------------\n";

print "#---------------------------REGLAS PARA TRAFICO LOCAL TCP-------------------------\n";

print "#-------------------------- Se excluye a usuarios especiales -------------------------------\n";

foreach $macp (@MAC_PRIVIL){

        foreach $p (@LOCAL_TCP){

                print "iptables -A FORWARD -i eth1 -o $iSalida -p tcp --dport $p -m mac --mac-source !
$macp -t mangle -j MARK --set-mark 2\n";
```

```
                print "iptables -A FORWARD -i eth3 -o $iSalida -p tcp --dport $p -m mac --mac-source !
$macp -t mangle -j MARK --set-mark 2\n";

                print "iptables -A FORWARD -i eth4 -o $iSalida -p tcp --dport $p -m mac --mac-source !
$macp -t mangle -j MARK --set-mark 2\n";

        };

};

print "#-------------------------------------------------------------------------------------------\n";

print "#----------------------------REGLAS PARA TRAFICO LOCAL UDP----------------------\n";

print "#-------------------------- Se excluye a usuarios especiales ----------------------------\n";

foreach $macp (@MAC_PRIVIL){

        foreach $p (@LOCAL_UDP){

                print "iptables -A FORWARD -i eth1 -o $iSalida -p udp --dport $p -m mac --mac-source !
$macp -t mangle -j MARK --set-mark 2\n";

                print "iptables -A FORWARD -i eth3 -o $iSalida -p udp --dport $p -m mac --mac-source !
$macp -t mangle -j MARK --set-mark 2\n";

                print "iptables -A FORWARD -i eth4 -o $iSalida -p udp --dport $p -m mac --mac-source !
$macp -t mangle -j MARK --set-mark 2\n";

        };

};

print "#-------------------------------------------------------------------------------------------------\n";

print "#--------------------REGLAS PARA TRAFICO MSN PERMITIDO----------------------------\n";

print "#-------------------- Se excluye a usuarios especiales -------------------------------\n";

#Excluye mac privilegiados porque se marcarán el el grupo de la marca 5

foreach $mac (@MAC_MSN){

        foreach $p (@MSN_TCP){

                print "iptables -A FORWARD -i eth1 -o $iSalida -p tcp --dport $p -m mac --mac-source
$mac -t mangle -j MARK --set-mark 3\n";

                print "iptables -A FORWARD -i eth3 -o $iSalida -p tcp --dport $p -m mac --mac-source
$mac -t mangle -j MARK --set-mark 3\n";
```

```
		print "iptables -A FORWARD -i eth4 -o $iSalida -p tcp --dport $p -m mac --mac-source $mac -t mangle -j MARK --set-mark 3\n";

	};

	foreach $p (@MSN_UDP){

		print "iptables -A FORWARD -i eth1 -o $iSalida -p udp --dport $p -m mac --mac-source $mac -t mangle -j MARK --set-mark 3\n";

		print "iptables -A FORWARD -i eth3 -o $iSalida -p udp --dport $p -m mac --mac-source $mac -t mangle -j MARK --set-mark 3\n";

		print "iptables -A FORWARD -i eth4 -o $iSalida -p udp --dport $p -m mac --mac-source $mac -t mangle -j MARK --set-mark 3\n";

	};

};

print "#-------------------------------------------------------------------------------------------------\n";

print "#-------------------------REGLAS PARA TRAFICO DE VIDEO ----------------------------\n";

print "#------------------------- Se excluye a usuarios especiales -----------------------------\n";

foreach $interfaz (@interfaces){

	foreach $macp (@MAC_PRIVIL){

		foreach $red (@VIDEO_RED){

			print "iptables -A FORWARD -i $interfaz -o $iSalida -p tcp -d $red -m mac --mac-source ! $macp -t mangle -j MARK --set-mark 4\n";

		};

		foreach $rango (@VIDEO_RANGO){

			print "iptables -A FORWARD -i $interfaz -o $iSalida -p tcp -m iprange --dst-range $rango -m mac --mac-source ! $macp -t mangle -j MARK --set-mark 4\n";

		};

	};

};

print "#-------------------------------------------------------------------------------------------------\n";
```

```
print "#----------------REGLAS PARA TRAFICO DE USUARIOS ESPECIALES------------------\n";

print "#-------------------------------------------------------------------------------------------------------\n";

foreach $macp (@MAC_PRIVIL){

        print "iptables -A FORWARD -i eth1 -o $iSalida -m mac --mac-source $macp -t mangle -j MARK --
set-mark 5\n";

        print "iptables -A FORWARD -i eth3 -o $iSalida -m mac --mac-source $macp -t mangle -j MARK --
set-mark 5\n";

        print "iptables -A FORWARD -i eth4 -o $iSalida -m mac --mac-source $macp -t mangle -j MARK --
set-mark 5\n";

};

#-------------------------------------------------------------------------------------------------------

#   CREACION DE CLASES Y COLAS

#-------------------------------------------------------------------------------------------------------

#----------------------------------NODO Qdisc RAIZ-----------------------------------------------

print "#-----------------------------------------------------\n";

print "#----------CREACION DE NODO RAIZ---------------------\n";

print "#-----------------------------------------------------\n";

print "tc qdisc add dev $iSalida root handle 1: htb default 15\n";

#-------------------------------------------NODO CLASE RAIZ-------------------------

print "#-------------------------------------------------------------------------\n";

print "#----------CREACION DE CLASE RAIZ-------------------------------------\n";

print "#-------------------------------------------------------------------------\n";

print "tc class add dev $iSalida parent 1: classid 1:1  htb rate $TOTAL ceil $TOTAL \n";

#----------------------------------NODOS HOJAS-----------------------------------------

print "#---------------------------------------------------------------------------------\n";

print "#-------------------CREACION DE NODOS HOJAS-----------------------------------\n";

print "tc class add dev $iSalida parent 1:1 classid 1:10  htb rate $CUOTA[1] ceil $TOTAL prio 0\n";
```

```
print "tc class add dev $iSalida parent 1:1 classid 1:11  htb rate $CUOTA[2] ceil $TOTAL prio 1\n";
print "tc class add dev $iSalida parent 1:1 classid 1:12  htb rate $CUOTA[2] ceil $TOTAL prio 2\n";
print "tc class add dev $iSalida parent 1:1 classid 1:13  htb rate $CUOTA[2] ceil $TOTAL prio 3\n";
print "tc class add dev $iSalida parent 1:1 classid 1:14  htb rate $CUOTA[0] ceil $TOTAL prio 0\n";
print "tc class add dev $iSalida parent 1:1 classid 1:15  htb rate $CUOTA[2] ceil $TOTAL prio 4\n";
#-------------------COLAS---------------------------------
print "#-------------------------------------------------\n";
print "#----------CREACION DE COLAS ----------------------\n";
print "#-------------------------------------------------\n";
print "tc qdisc add dev $iSalida parent 1:10 handle 10: sfq\n";
print "tc qdisc add dev $iSalida parent 1:11 handle 11: sfq\n";
print "tc qdisc add dev $iSalida parent 1:12 handle 12: sfq\n";
print "tc qdisc add dev $iSalida parent 1:13 handle 13: sfq\n";
print "tc qdisc add dev $iSalida parent 1:14 handle 14: prio\n";
print "tc qdisc add dev $iSalida parent 1:15 handle 15: sfq\n";
#-------------------FILTROS-------------------------------------------------------
print "#---------------------------------------------------------------------------\n";
print "#----------------------------CREACION DE FILTROS----------------------------\n";
print "#---------------------------------------------------------------------------\n";
print "tc filter add dev $iSalida protocol ip  parent 1: handle 1 fw classid 1:10\n";
print "tc filter add dev $iSalida protocol ip  parent 1: handle 2 fw classid 1:11\n";
print "tc filter add dev $iSalida protocol ip  parent 1: handle 3 fw classid 1:12\n";
print "tc filter add dev $iSalida protocol ip  parent 1: handle 4 fw classid 1:13\n";
print "tc filter add dev $iSalida protocol ip  parent 1: handle 5 fw classid 1:14\n";
```

Anexo 5. Análisis de las Estadísticas del Tráfico.

Se han ejecutado las reglas propuestas de control de tráfico en el Gateway 8 y luego se ha aplicado el comando *tc –s class show dev eth2,* el cual obtiene las siguientes estadísticas que determinan el comportamiento instantáneo del tráfico, en intérvalos o momentos de 6 segundos, durante un minuto:

MOMENTO	1	2	3	4	5	6	7	8	9	10
Qdisc HTB Raíz										
OVERLIMITS	5970	5982	6012	6147	6760	8108	9295	10834	12227	13787
Tráfico WEB										
RATE (bit)	105536	87744	73880	73520	91336	75848	57816	51424	46008	41424
LENDED	41640	41921	42196	42591	42802	43044	43204	43418	43627	43858
BORROWED	1048	1048	1061	1100	1100	1100	1100	1100	1102	1102
Tráfico LOCAL										
RATE (bit)	0	0	0	0	0	0	0	0	0	0
LENDED	98	98	98	98	98	98	98	98	98	98
BORROWED	0	0	0	0	0	0	0	0	0	0
Tráfico VIDEO										
RATE (bit)	0	0	0	0	0	0	0	0	0	0
LENDED	0	0	0	0	0	0	0	0	0	0
BORROWED	0	0	0	0	0	0	0	0	0	0
Tráfico MSN										
RATE (bit)	552	416	1296	968	1472	1104	624	464	280	208
LENDED	410	422	428	433	443	443	433	434	434	434
BORROWED	24	24	24	24	24	24	24	24	24	24
Tráfico ESPECIAL										
RATE (bit)	36448	33920	32064	31424	28496	31064	54320	86440	98488	83376
LENDED	74558	74861	75319	75720	76073	76627	77461	78587	79132	79852
BORROWED	600	600	600	600	600	600	600	606	606	606
Tráfico por DEFECTO										
RATE (bit)	109440	113976	101032	102624	157792	318224	479512	527336	628688	674904
LENDED	75134	76070	76796	77492	78157	78776	79171	79582	80038	80461
BORROWED	7857	7909	7911	8089	8498	9162	9644	10223	10928	11648
Tráfico TOTAL										
RATE (bit)	251976	236064	208272	208536	279104	426240	592272	665664	773464	797136
LENDED	9529	9581	9596	9813	10222	10886	11368	11953	12660	13380
BORROWED	0	0	0	0	0	0	0	0	0	0

Estadísticas del Comando TC

Los datos estadísticos son mostrados en el siguiente gráfico:

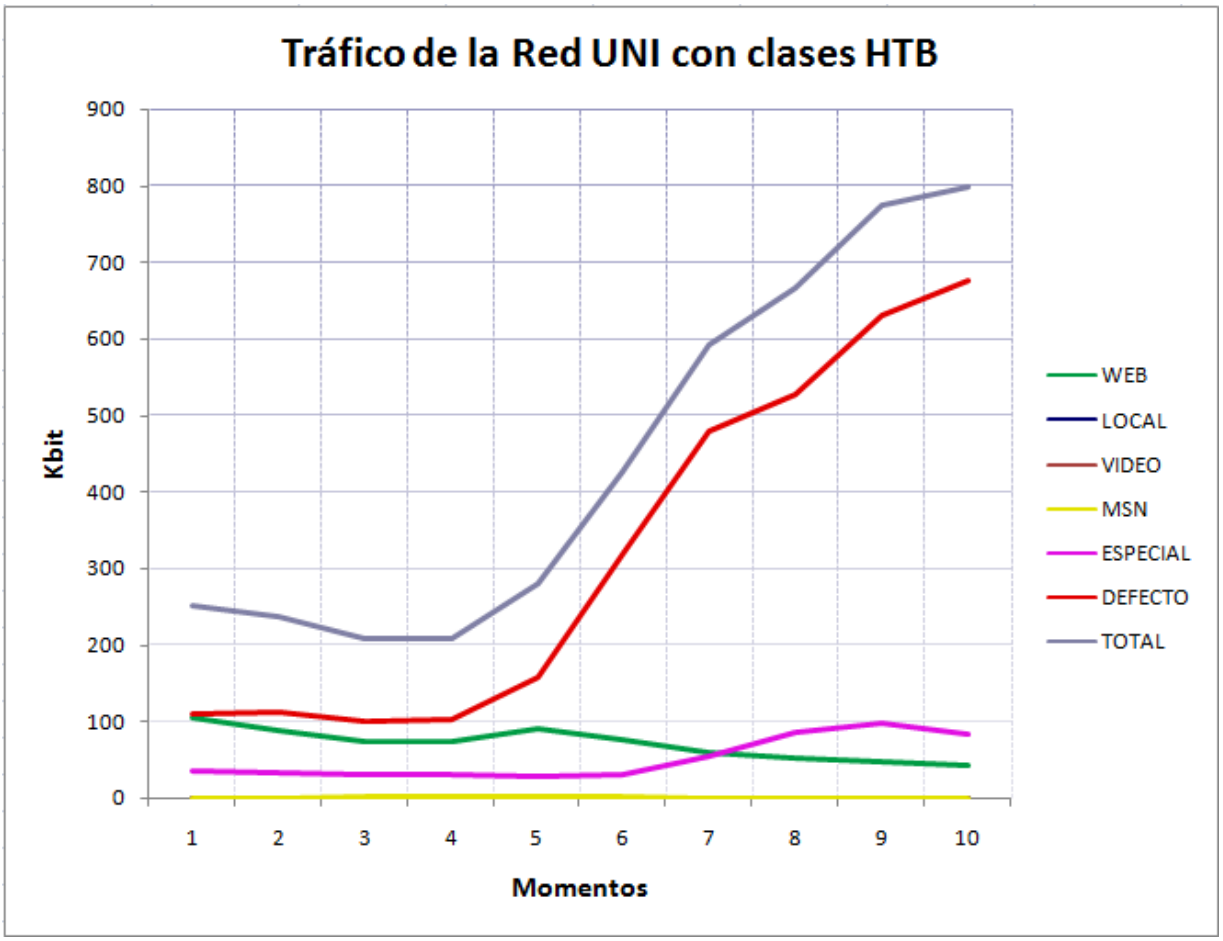


Gráfico de las Estadísticas del Comando TC

El gráfico muestra el consumo del ancho de banda de las clases en diferentes momentos. La clase padre, representada como TOTAL (celeste), muestra la suma del consumo de las clases hijas.

Nota: Durante el tiempo que se realizó la captura de los datos estadísticos, el consumo del ancho de banda de las clases LOCAL (azul), VIDEO (café) y MSN (amarillo) fue mínima o nula, por lo que se considera despreciable en este análisis.

Las clases hacen uso del ancho de banda, respetando las tasas configuradas. Los valores de OVERLIMITS, indican cuantas veces la disciplina HTB ha retrasado paquetes, para evitar que se sobrepase la tasa máxima establecida, lo que significa que HTB ha realizado shaping.

En el gráfico, es posible notar, que la cantidad de servicio que solicitan algunas clases es menor a su mínimo garantizado (*rate*), por lo que, existe ancho de banda excedente, el cual puede ser utilizado por otras clases que requieran y soliciten el servicio. Por ejemplo, en el momento 5, la clase de tráfico por DEFECTO (rojo) alcanza su mínimo garantizado y las clases restantes se mantienen muy por debajo de su mínimo, además de una leve disminución en la solicitud de servicio de la clase de tráfico WEB (verde), lo cual es aprovechado por la clase de tráfico por DEFECTO para solicitar más ancho de banda después de haber consumido el que le ha sido garantizado para su uso. Esta clase, ha recibido en proporción a su quantum, todo el ancho de banda que ha solicitado después de haber consumido el que le corresponde, porque las clases restantes no han hecho uso de su ancho de banda y éste ha quedado disponible para la clase que lo solicite, es por tal motivo que, mientras el tráfico WEB disminuye y el tráfico ESPECIAL solicita muy poco servicio, el tráfico por DEFECTO logra ascender progresivamanente hasta casi 700 kbits, superando su mínimo garantizado de 150 kbits, ya que su ceil le permite llegar hasta los 3000 kbits siempre que haya ancho de banda disponible en el enlace. Esta situación se debe, a que cuando una clase tiene menos demanda, su ancho de banda queda disponible para que otra clase lo use, a través de préstamos.

El comportamiento de cada clase es variable, en dependencia a la demanda de los usuarios, por ejemplo, la clase ESPECIAL, en los momentos de 1 a 6 mantiene un consumo de aproximadamente 30 kbits, luego, del momento 6 al 9, alcanza un consumo de 100 kbit, y finalmente en el momento 10, su consumo baja a 80 kbits. La clase WEB mantiene un consumo bastante constante, con pequeños ascensos a 100kbits en los momentos 1 y 5. La clase de tráfico por DEFECTO, por el contrario, muestra un ascenso continuo desde el momento 4 y superando su tasa mínima a partir del momento 5.

Todas las clases han sido configuradas para alcanzar la tasa máxima (ceil), con el fin de que no quede ancho de banda en desuso cuando hay otro tráfico que lo necesita. Si aún así, hay tráfico en desuso, significa que no hay demanda porque nadie lo ha solicitado, lo que se visualiza en este gráfico. Cabe mencionar, que estos datos han sido capturados en un periodo de vacaciones universitarias, por lo que hay poca

demanda de tráfico en la red. El gráfico muestra un consumo total de hasta 800 kbits, mientras que el enlace total es de 3000 kbits.

Según los parámetros de configuración de cada clase, mostrados en la siguiente tabla, después de la clase ESPECIAL, es a la clase WEB, a quien se le ofrecerá una mayor cantidad del ancho de banda excedente, porque es la que sigue en tamaño de quantum y se le ofrecerá hacer uso de éste, en primera prioridad, porque su valor PRIO es cero. Por tanto, en un momento de inactividad de los usuarios especiales, es el tráfico WEB el de mayores privilegios en la red. La regla es que se ofrece el ancho de banda excedente, primero a las clases con prioridades superiores y respetando las reglas sobre las tasas rate y ceil. Sin embargo, como es notorio en este caso, en un momento que las clases de mayor privilegio, no tengan demanda, cualquier otra clase puede solicitar el ancho de banda excedente y se le concederá en proporción a su quantum, siempre y cuando esté disponible, ya que es importante mencionar, que en caso de que una clase esté muy por encima de su tasa mínima, HTB priorizará a las clases por debajo de su mínimo garantizado cuando ellas soliciten servicio.

Clase / Tráfico	Rate (kbit)	Ceil (kbit)	Prio	Quantum (byte)	Quantum (kbit)
WEB	900	3000	0	11250	90
LOCAL	150	3000	1	1875	15
VIDEO	150	3000	2	1875	15
MSN	150	3000	3	1875	15
ESPECIAL	1500	3000	0	18750	150
DEFECTO	150	3000	4	1875	15

Parámetros de Configuración de las Clases HTB