SISTEMA OPERATIVO LINUX Y CONTROL DE TRÁFICO EN REDES DE COMPUTADORES

JUAN CARLOS RENGIFO SALAZAR

ALVARO URREA LUJÁN

Monografía para optar el título de Especialista en Ciencias Electrónicas e Informática con énfasis en Telemática

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERIA

Departamento de Electrónica

MEDELLÍN, 2004

Tabla de contenidos

RESUMEN

El objeto de este trabajo es describir las posibilidades que ofrece el sistema operativo LINUX, dentro del ámbito de redes de computadores, centrándose en el tema de control de tráfico, específicamente se hará un barrido sobre temas como los conceptos básicos de la teoría de colas en las redes de computación, las herramientas del kernel de Linux que definen las clases, políticas y filtros para el control de tráfico, la administración del ancho de banda en una red de computadores y los elementos que en ella intervienen.

En el primer capítulo se hace una breve introducción a la teoría de colas, sus modelos matemáticos, características y tipos de colas, para terminar con la presencia de las colas en los sistemas de comunicaciones, mas exactamente con las redes de computadores.

En el segundo capítulo se define el concepto de control de Tráfico en general, y se hace una revisión detallada de los elementos relacionados, tales como colas, flujos, paquetes y de los mecanismos usados para control de tráfico, independiente de los sistemas operativos como son: scheduling, policing, Clasifying, etc. Igualmente se hace una aproximación mas detallada del tema de disciplina de colas, haciendo énfasis en aquellas que tiene aplicabilidad en el campo de control de tráfico.

En los capítulos 3 y 4 se aborda el tema de enrutamiento avanzado con LINUX, y su aplicabilidad al control de tráfico, presentando las herramientas con que cuenta el sistema operativo con Kernel 2.4.x.+, incluyendo IPTABLES e IPROUTE2. Esta última herramienta ya contiene la utilidad tc que es la que en LINUX permite hacer el control de tráfico, aplicando los conceptos vistos en capítulos anteriores, tales como disciplina de colas, clases y filtros.

En el capítulo 5 se toma un caso de estudio y mediante la metodología de práctica de laboratorio, se realizan diferentes acciones de control de tráfico sobre una pequeña red. Dichas acciones obviamente son realizadas mediante un sistema operativo LINUX, y mediante la medición hecha con herramientas de monitoreo de tráfico se comprueban los resultados esperados

INTRODUCCIÓN

La amplia presencia redes de computadores, la necesidad de interconexión entre ellas para el intercambio de información y la exigencia de emplear eficientemente los recursos, llevan a considerar el empleo de aplicaciones de software libre para generar soluciones que satisfagan estas necesidades y exigencias, uno de los recursos importantes en la redes de computadores es el ancho de banda que se requiere para la transmisión de la información, es sabido que este es costoso y limitado, para utilizar eficientemente el canal se necesita controlar de alguna forma el trafico en la red o entre las redes, Linux aparece como una opción para dar solución a este problema, pues cuenta en su kernel con un módulo para redes el cual tiene todo el potencial de las soluciones propietarias a nivel comercial.

Linux trabaja bajo un esquema multitarea y multiusuario, cuenta con herramientas de desarrollo y por su estructura permite a los desarrolladores tener acceso al hardware y a las redes que a este se conectan. La siguiente tabla ilustra cual es la presencia de Linux en el mundo de las telecomunicaciones y la informática:

Tabla 1. Presencia de *Linux* en el mundo de las telecomunicaciones y la informática [1]

Áreas de presencia del sistema operativo Linux
Client Interface
Line of Business
Middleware
Data Store
Infraestructura
Operating Systems

Si bien uno de los puntos fuertes de LINUX es su carácter de código abierto, en comparación a sistemas operativos como UNIX y WINDOWS, presenta otras fortalezas:

- Costo total de compra y envío inferior.
- Tecnología creciente y apoyo de la industria.
- Gran numero de ambientes y aplicaciones.
- Crecimiento constante del conocimiento del mismo por parte de los desarrolladores.
- Seguridad.
- Flexibilidad.

Desde la versión 2.0.x el kernel de Linux trae un conjunto de utilidades relacionadas con el enrutamiento avanzado y la conectividad de redes de computadores, que recogen temas como control de trafico, servicios diferenciados, calidad de servicio, firewalls, túneles, etc. Se puede destacar ipfwadm e ipchains en el kernel 2.2.x y netfilter a partir del kernel 2.3.15 y superiores con el comando iptables. Adicionalmente se cuenta con soporte para el modulo de control de trafico la utilidad TC (Traffic Control) de la herramienta IPROUTE2.

Linux nos brinda la posibilidad de efectuar tareas de control de tráfico, para gestión del ancho de banda de una red de computadores, algunas de ellas se listan a continuación:

- Limitar el ancho de banda conocido a un valor predeterminado.
- Limitar el ancho de banda de un cliente en particular.
- Reservar ancho de banda para una aplicación en particular de un usuario específico.

Se hace necesario disponer de soluciones que hagan un uso eficiente del ancho de banda disponible y que además se puedan brindar servicios diferenciados, Linux se convierte entonces en una de ellas, por su creciente popularidad, utilidades, evolución, costos y flexibilidad.

En el presente trabajo se revisan la las herramientas con que cuenta el sistema operativo Linux, para realizar tareas de control de trafico, se toma como referencia la versión REDHAT 9, cuyo kernel es 2.4.20-8, este inicia tratando los conceptos básicos necesarios alrededor de la teoría de colas para comprender su importancia en las redes de computadores, enseguida se abordan los conceptos presentes en el control de trafico

estableciendo así las bases necesarias para el desarrollo de los temas posteriores, luego se revisan las características de Linux relacionadas con el enrutamiento avanzado se presentan las herramientas fundamentales para efectuar tareas tales como túneles, firewall, múltiples tablas de enrutamiento, gestión del ancho de banda, multicasting, etc., se dedica un capitulo a la temática de control de tráfico manejada desde la óptica del sistema operativo Linux, centrándose en presentar los conceptos de disciplinas de colas, clases y filtros con algunos ejemplos muy sencillos para revisar su configuración, usando la utilidad tc de IPROUTE2 como implementación en el sistema operativo, finalmente se realiza un ejemplo practico de aplicación de los conceptos de gestión de ancho de banda sobre una red de computadores muy simple, con el fin de ilustrar el potencial con que cuenta Linux.

1. TEORIA DE COLAS Y LAS REDES DE COMPUTADORES

1.1 INTRODUCCIÓN AL CONCEPTO DE COLAS

Las líneas de espera o colas y los sistemas de colas han sido sujeto de extensos estudios, desde el mismo surgimiento de la telefonía hasta hoy, con la presencia de amplias redes de computadores.

Sobre todo a nivel de redes de computadores el tráfico de datos que circula en estas requiere de evaluación de modelos que incluyan análisis de colas. Se habla entonces de la Teoría de Colas la cual pertenece a las matemáticas y específicamente a una rama aplicada de las probabilidades y los procesos estocásticos, se busca entonces dar respuesta de como es el comportamiento de ingreso a una cola, los tiempos de espera, el servicio, el tiempo de servicio y la salida del sistema buscando la optimización del mismo.

En redes de comunicación, por ejemplo en una red de computadores, los paquetes que fluyen por la red solicitan ser servidos por los diferentes nodos (Routers, Switches, Bridges, Hubs), se requiere considerar entonces el ancho del canal de transmisión, la velocidad de procesamiento, el tamaño del buffer y la priorización del flujo de estos paquetes en la red. Para el diseño y evaluación de equipos de procesamiento y redes de

computadores se necesita hacer uso de la Teoría de Colas, mediante el empleo de modelos de colas y análisis simplificado de estos, soportados en suposiciones que permiten reducir la complejidad del análisis y llevan a resultados satisfactorios.

En general para un modelo de colas se pueden definir tres características básicas: proceso de entrada, mecanismo de servicio y disciplina de cola.

Proceso de entrada: describe la secuencia de solicitudes de servicio. Generalmente el proceso de entrada es descrito en términos de distribución a lo largo del tiempo entre instantes consecutivos de ingreso de clientes.

Mecanismo de servicio: tiene que ver con el número de servidores y la cantidad de tiempo que el cliente usa los servidores.

Disciplina de colas: especifica la disposición de que se tiene para bloquear los clientes (clientes que encuentran todos los servidores ocupados), puede darse que los clientes bloqueados decidan abandonar el sistema inmediatamente o que esperen en la cola por el servicio, los cuales pueden ser atendidos en orden especifico. [2,3]

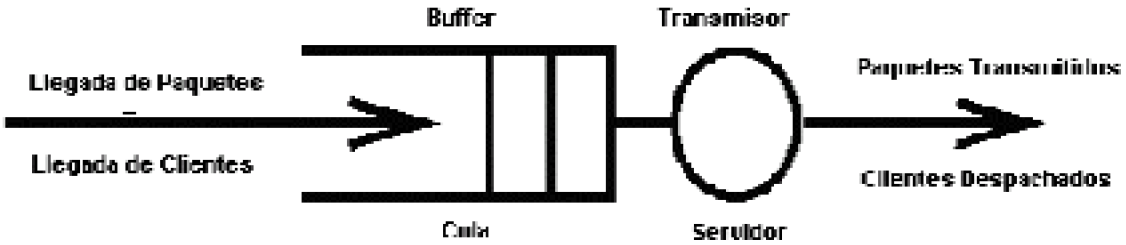


Figura 1. Modelo de Cola de un solo servidor. [4]

La figura 1 presenta un esquema de cola de un solo servidor donde se visualizan en la parte superior los términos de las redes de conmutación de paquetes en un enlace de transmisión y en la parte inferior los términos asociados a la teoría de colas.

La teoría de colas es matemáticamente compleja, sin embargo es posible a nivel aplicativo realizar un análisis de colas de manera directa, basta con tener conocimientos básicos de estadística y comprender la aplicabilidad de la teoría de colas.

En general un sistema de colas se clasifica de acuerdo a sus propiedades, algunas de ellas son:

- La forma de la distribución de las llegada de los clientes
- La forma de la distribución del tiempo de servicio
- El numero de llegadas en un grupo
- El numero de servidores
- La disciplina de servicio, es la manera en el cual el servicio es condicionado, si el sistema tiene prioridades
- Numero de clientes que se les permite esperar
- Numero de clientes en la población. [6]

Algunas de las cantidades de interés en el estudio de sistemas de colas son la

distribución del tiempo de espera, la distribución del tiempo del sistema, la distribución del numero de clientes en el sistema, la probabilidad de que el servidor este ocupado o no, la distribución de la longitud de un periodo de ocupación, la distribución del numero de clientes servidos durante un periodo de ocupación, promedios para tiempos de espera.

Se verá a continuación algunos elementos preliminares de interés para una mejor compresión de la teoría de colas, relacionado con la teoría de probabilidades y los procesos estocásticos.

1.2 ELEMENTOS DE TEORÍA DE PROBABILIDADES Y PROCESOS ESTOCÁSTICOS

1.2.1 Variables Aleatorias.

Dentro de la teoría de probabilidades surge el concepto de variable aleatoria dentro de un espacio muestral como una función de este, y lo que nos interesa es que representa mediante un valor numérico un evento al cual se le puede asociar una probabilidad, ejemplos de variables aleatorias en sistemas de colas es el numero de clientes en espera, el tiempo de ocupación del servidor, el tiempo de espera.

1.2.2 Distribución exponencial.

Se tiene una variable aleatoria continua X con una función de distribución

$$P\{X \leq x\} = F(x)$$

se dice entonces que es exponencial si cumple:

$$F(x) = \begin{cases} 1 - e^{-\mu x} & (x \geq 0) \\ 0 & (x < 0) \end{cases} \quad (1)$$

Y su función de densidad de probabilidad es:

Una distribución exponencial tiene la propiedad de ausencia de memoria o propiedad de Markov, que se define como, una variable aleatoria X sin memoria si y solo si para cada

se da:

$$P\{X > \alpha + \beta \mid X > \beta\} = P\{X > \alpha\} \quad (3)\ [6]$$

Es decir la historia pasada no tiene ninguna influencia en el presente.

1.2.3 Distribución de Poisson.

Una variable aleatoria discreta X sigue una distribución de Poisson con un parámetro

$$\lambda$$

si su función de densidad de probabilidad es:

1.2.4 Proceso de Poisson.

Muchos sistemas de colas en el proceso de llegada son caracterizados como tipo Poisson, pues al asumir que el comportamiento de llegada de los clientes en una cola se asemeja a un proceso de Poisson, nos permite reducir la complejidad analítica del problema permitiendo obtener resultados útiles.

Se tiene un proceso A(t) que corresponde al numero de llegadas en un tiempo t donde A(0)=0, y A(t) - A(s) son el numero de llegadas en un intervalo (s, t].

El proceso A(t), con

$$t \geq 0$$

sigue una distribución de Poisson con parámetro

$$\lambda t$$

de acuerdo a:

Donde

$$\lambda\tau$$

numero medio de llegadas y

$$\lambda$$

es la rata de llegadas.

Cabe destacar dos propiedades importantes de un proceso de Poisson:

1. Mezclado: Se asume que se tienen

$$N_1(t) \text{ y } N_2(t)$$

ambos procesos de Poisson independientes con ratas

$$\lambda_1 \text{ y } \lambda_2$$

respectivamente de estos dos procesos se da origen a un nuevo proceso de Poisson con rata

$$\lambda_1 + \lambda_2 .$$

2. División: Se asume que se tiene un proceso de Poisson

con una rata

$$\lambda$$

y cada llegada está marcada por una probabilidad P independiente de las otras llegadas. Dejando que

$$N_1(t) \text{ y } N_2(t)$$

sean respectivamente los procesos de salida de este, entonces se tiene una distribución de Poisson con ratas

$$p\lambda \text{ y } (1-p)\lambda$$

para cada camino de salida respectivamente.

Se tiene entonces que un proceso de Poisson se mantiene bajo el mezclado o la división.

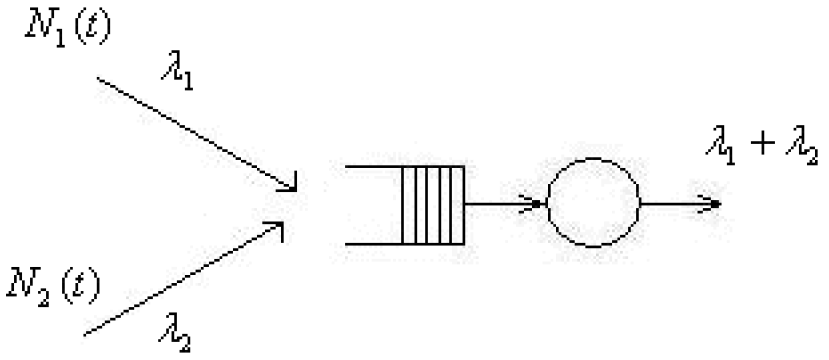


Figura 2 Propiedad de Mezclado de un proceso de Poisson

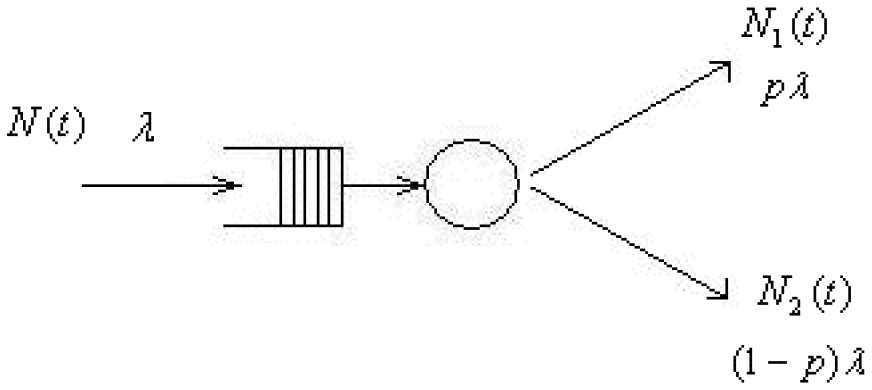


Figura 3. Propiedad de División de un Proceso de Poisson

1.2.5 Proceso de llegada y tiempo de servicio.

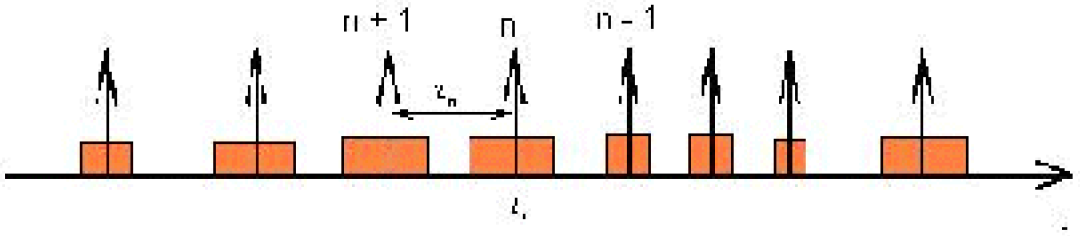


Figura 4. Proceso de llegada [4]

En la figura 2 se tiene

$$\tau_n$$

tiempo entre llegadas de los clientes n y n +1, el cual es aleatorio y responde a un proceso estocástico.

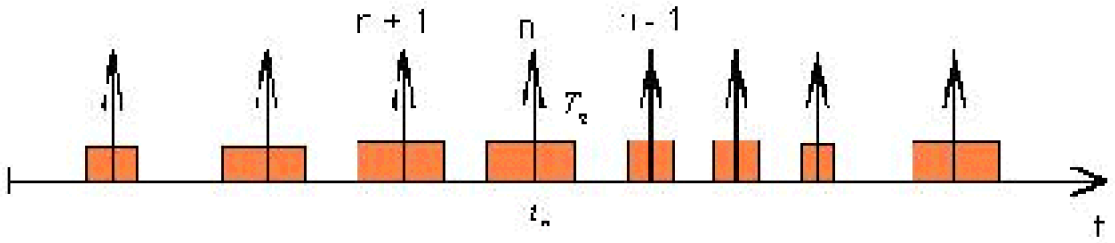


Figura 5. Proceso de tiempo de servicio

Se tiene que

es el tiempo de servicio del cliente n en el servidor, el cual es un proceso estocástico para n>=1

1.2.6 Procesos de Markov.

Un proceso de Markov es un proceso estocástico donde:

$$\{X(t), t \in T\}, \quad X(t) \in E \subset \Re$$

Para el cual se cumple:

$$P(X(t) \le x \mid X(t_1) = x_1, \ldots, X(t_n) = x_n) = P(X(t) \le x \mid X(t_n) = x_n) \quad (6)$$

Para todo $x_1, \ldots, x_n \in E$, $t_1, \ldots, t_n \in T$, $con \quad t_1 < t_2 < \ldots < t_n < t$

De manera intuitiva podemos deducir de (6) que un proceso de Markov donde la evolución del futuro probabilístico después de cualquier tiempo t depende únicamente del estado del sistema en el tiempo t y es independiente de la historia del sistema a priori al tiempo t,

Se diferencia las cadenas de Markov en tiempo discreto y las cadenas de Markov en tiempo continuo.

1.3 NOTACIÓN DE KENDALL

Con el objeto de simplificar la definición de un modelo cola se emplea la notación de Kendall, quien establece:

A/B/c/K (7)

Donde A se refiere al tipo de distribución de llegada de clientes, B define la distribución del tiempo de servicio, c el numero de servidores y K el tamaño de la capacidad del sistema.

Para A y B generalmente se definen distribuciones tipo exponencial (M), determinístico (D) o distribución general (G, GI), cuando la capacidad del sistema es infinito K = ∞, la notación se simplifica quedando A/B/c.

Para el caso de un sistema de colas de un solo servidor la notación según Kendall podría ser M/M/1, que se refiere a que la distribución de llegada es de acuerdo a una distribución de Poisson, la distribución del tiempo de servicio es exponencial, un solo servidor y un buffer infinito.

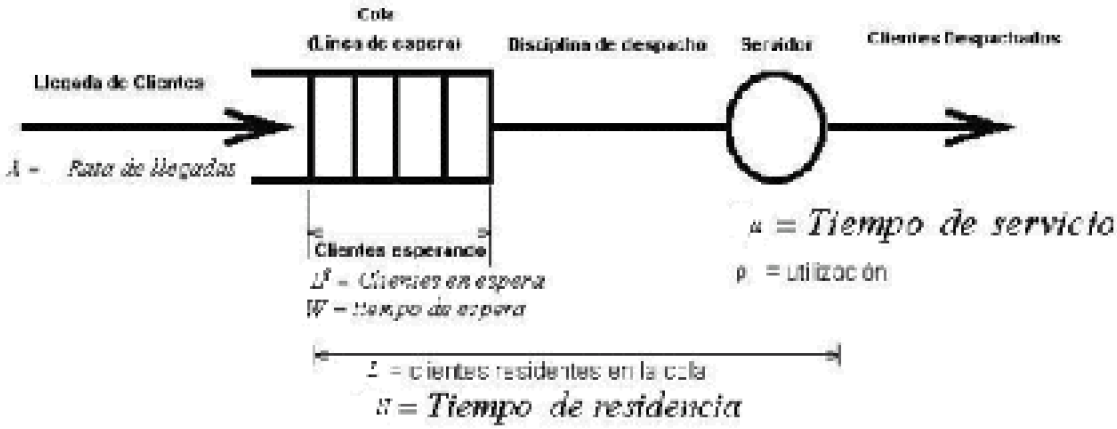


Figura 6. Cola de un solo servidor con sus parámetros. [5]

En la figura 6 aparecen los parámetros de interés en un modelo de colas,

$$\lambda$$

rata de llegadas o numero medio de llegadas por segundo, Lq numero de clientes esperando ser servidos, W tiempo de espera (incluye los clientes que han de esperar y los clientes con tiempo de espera = 0), L numero de clientes en el sistema tanto los que están en espera como los que están siendo servidos, S tiempo que gasta un cliente en el sistema, u tiempo de servicio por cada llegada, que es el tiempo empleado por el servidor para atender un cliente y no incluye el tiempo de espera en la cola , p fracción de tiempo que el servidor o servidores están ocupados. Con estos parámetros podemos no solo caracterizar la cola sino que además permite realizar un análisis de la misma.

Se presentarán a continuación varios modelos de colas que nos permitirán aproximarnos de manera básica al análisis de colas.

1.4 COLA DE UN SOLO SERVIDOR

Se cuenta con un esquema donde existe un solo servidor el cual provee servicio a los clientes, estos últimos llegan solicitando ser servidos, si el servidor se encuentra desocupado un cliente es atendido, de lo contrario los clientes que llegan se unen a una línea de espera o cola, adicional a esto el servidor se toma un tiempo en atender cada cliente.

Partiendo de un modelo de cola M/M/1, los clientes llegan siguiendo un proceso de Poisson con una rata

$$\lambda$$

el tiempo que se toma en servir cada cliente es una variable aleatoria exponencial con parámetro

$$\mu$$

es decir, los clientes tienen un tiempo de servicio exponencial. Se supone que los tiempos de servicio son mudamente independientes e independientes de los tiempos de llegada.

Se tiene que la media entre llegadas exponenciales es

$$1/\lambda$$

y los tiempos de servicio exponencial tiene una media

$$1/\mu\ .$$

Se requiere que

$$\rho = \lambda / \mu < 1$$

de lo contrario la longitud de la cola colapsara.

Donde

$$\rho$$

es la fracción de tiempo que el servidor trabaja.

Se establece que el número medio de clientes en el sistema es:

Y que el número medio de tiempo de servicio y espera, es decir, tiempo en el sistema es:

Para el número medio de clientes en la cola:

Y el tiempo medio de espera E(W) puede expresarse como sigue:

$$E(W) = \frac{\rho / \mu}{1 - \rho} \quad (11)$$

Figura 7. Cola con múltiple servidores [5]

1.5 COLA CON MÚLTIPLES SERVIDORES

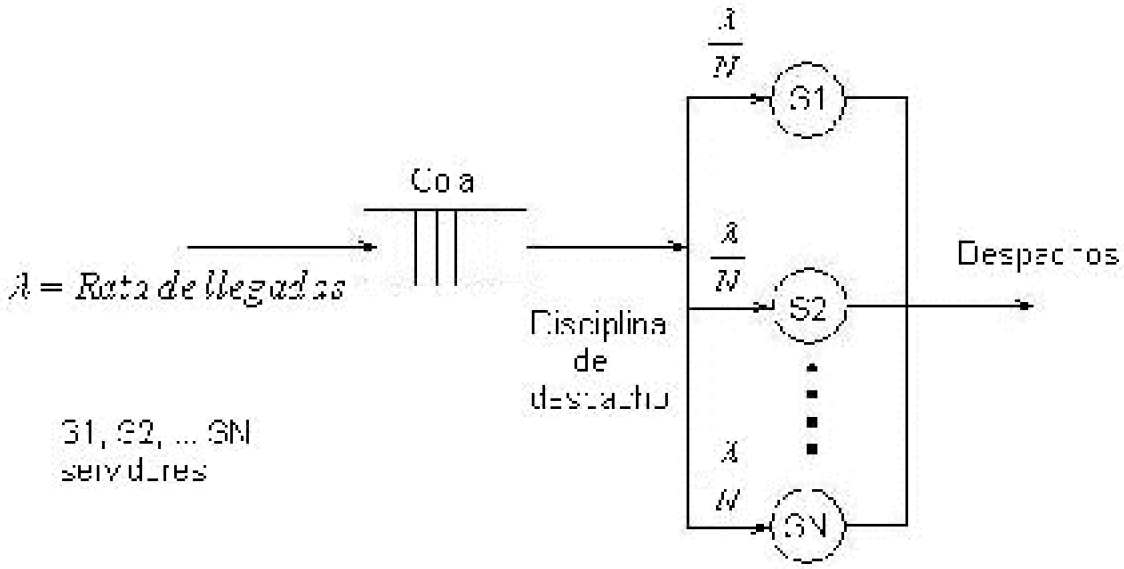


Figura 7. Cola con múltiple servidores [5]

De la figura 5 se observa que se tiene una cola que comparte múltiples servidores, si llega un cliente y existe un servidor libre, este es atendido, pero si todos los servidores se encuentran ocupados se inicia la formación de una línea de espera o cola. Se puede asumir que los servidores en paralelo son idénticos, los clientes son servidos en orden de llegada. Según la notación de Kendall se tiene un modelo M/M/c, con comportamiento exponencial entre llegadas, tiempo de servicio exponencial y numero finito de servidores c.

Donde la rata de ocupación del servidor es

$$\rho = \frac{\lambda}{c\mu} < 1$$

Se tiene que el número medio de clientes en la cola es:

Donde n es el número de clientes en el sistema.

Si se define:

$$P_W = \frac{(c\rho)^c}{c!}\left((1-\rho)\sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!}\right)^{-1} \quad (13)$$

Se establece que el tiempo medio de espera es:

$$E(W) = P_W\left(\frac{1}{1-\rho}\right)\frac{1}{c\mu} \quad (14)$$

1.6 Formula de Little

Un interesante resultado obtenido por Little, es el que tiene que ver los tiempos de respuesta y los tiempos de espera.

Si dejamos que

$$\lambda > 0$$

rata de llegada al sistema de colas en estado estacionario, es decir, los clientes que llegan se mantienen en el sistema por un tiempo determinado (tiempo de espera) y luego salen de este. Se define a E(S) como el tiempo medio de espera en el sistema (tiempo de espera más tiempo de servicio) y a E(L) como el número medio de clientes en el sistema entonces se tiene la formula de Little:

$$E(L) = \lambda E(S) \quad (15)$$

La formula cobra interés pues a menudo se conocen E(L) y

$$\lambda$$

básicamente establece que el numero medio de clientes en el sistema de colas en estado estacionario es igual a la rata de llegada de los clientes al sistema por el tiempo medio de espera en el sistema.

1.7 Prioridad de colas

La presencia de clientes con diferentes exigencias crea la necesidad de pensar en priorizar el servicio de atención de estos, se revisara este tópico usando el modelo de cola con un solo servidor. Se requiere entonces servir a los clientes en orden de prioridad debiendo ser atendidos primero aquellos con una prioridad mayor. En un sistema de colas con prioridad los clientes son divididos en K clases, con i=1,2....K, la menor prioridad, el mayor numero de clase, es decir, clientes con prioridad i se les da preferencia sobre los clientes con prioridad j si i<j. Los clientes llegan de acuerdo a un proceso de Poisson con rata

$$\lambda_i$$

el tiempo de servicio es independiente con una función de distribución

$$G_k(x), \text{media } 1/\mu_k$$

según notación de Kendall el modelo es M/G/1. Se asume que los tiempos de servicio y los tiempos de llegada son independientes e idénticamente distribuidos. Adicionalmente se tiene que

$$\rho_i = \frac{\lambda_i}{\mu_i}$$

es la intensidad de tráfico de la clase i.

De la figura 6 se asume que en cada clase de cola los clientes son servidos en el orden primero en ingresar - primero en salir y existe una cola por cada clase.

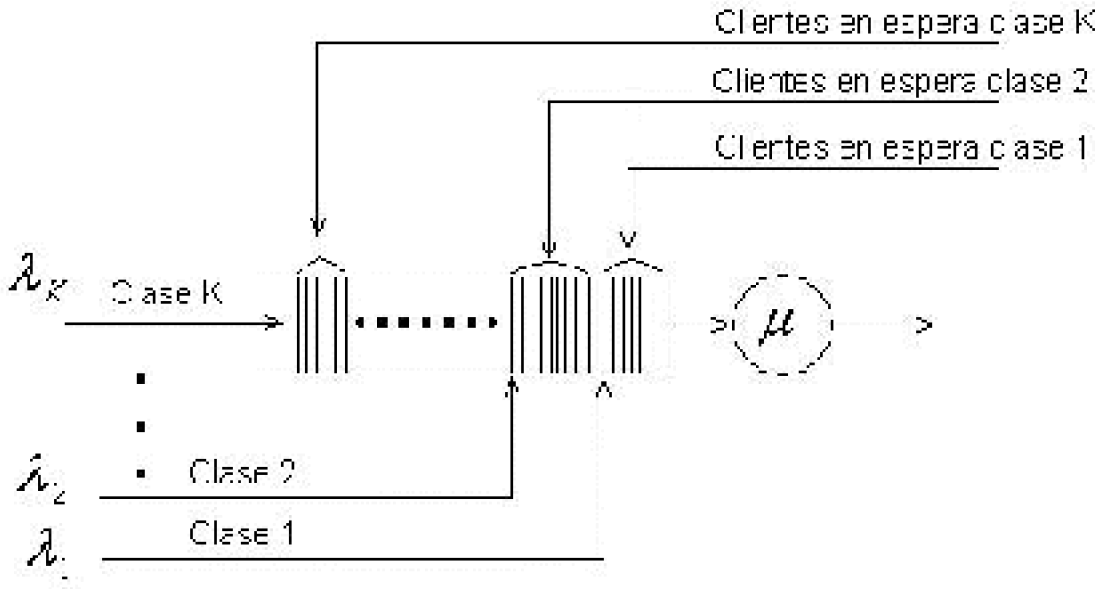


Figura 8. Cola de un solo servidor con prioridad [6]

Se consideran dos tipos básicos de prioridad, la primera es prioridad de no interrupción donde los clientes de mayor prioridad no pueden interrumpir el tiempo de servicio de un cliente de menor prioridad debiendo esperar a que termine de ser servido para poder ser atendidos, la segunda es prioridad de interrupción – reinicio aquí las interrupciones son permitidas para dar servicio a los clientes de mayor prioridad debiendo reiniciar el tiempo servicio después de la interrupción en el punto donde fue interrumpido, existe también la opción de una prioridad de interrupción-repetición en este caso particular el cliente que fue interrumpido cuando de nuevo es servido, su servicio vuelve a empezar como si hubiese acabado de ingresar al servidor. Estos dos tipos de prioridad también son conocidos como no absoluta y absoluta respectivamente.

En el esquema de prioridad de no interrupción el tiempo de cola promedio de un cliente depende de la rata de llegada de los clientes de mas baja prioridad, para el caso

de prioridad de interrupción los clientes de prioridad i no son afectados por los clientes de prioridad mas baja.

1.8 Redes de Colas

En ambientes distribuidos no solo existen colas aisladas, si no que estas se encuentran interconectadas formando una red colas, esta situación lleva a que el análisis sea de mayor complejidad y dos elementos que contribuyen a esto son:

- ·La división y mezclado del tráfico
- ·La existencia de colas en tandem o serie

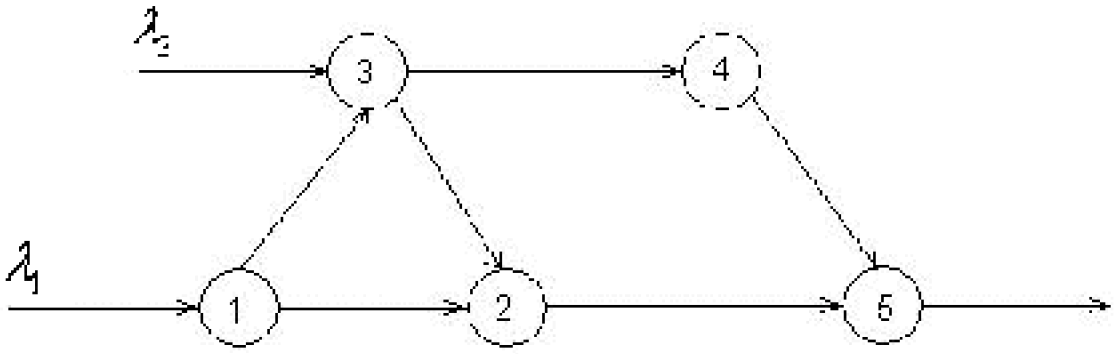


Figura 9. Gráfico de una red de colas [5]

De la figura 9 se tiene que los nodos representan colas y las líneas son el flujo de tráfico, aquí en el nodo 5 muestra un mezclado de tráfico, el nodo 1 división de tráfico y los nodos 3-4 y 1-2 ilustran colas en serie.

Básicamente cuando se habla de redes de colas se presentan dos modelos las redes de colas abiertas y las redes de colas cerradas y asociado a este tema de redes está presente el teorema de Jackson y las redes de Jackson.

1.8.1 Teorema de Jackson.

El teorema de Jackson es utilizado para realizar análisis de colas en redes de colas, alrededor de este tema se habla de redes de Jackson, para este caso una red de Jackson es una red de k colas que satisface tres propiedades:

Los servidores en cada una de las k colas son independientes unos de otros y tiene un tiempo de servicio

$$\mu_i$$

con i=1,2....K,

Las llegadas externas en cada cola siguen un proceso independiente de Poisson con ratas

$$\lambda_i \geq 0$$

para i=1,2....K,

Las redes utilizan un enrutamiento aleatorio

El teorema de Jackson establece:

Las colas en una red de Jackson son colas independientes tipo M/M/1

El numero de paquetes y los retrasos en cada cola son independientes de estos en todas las otras colas

Si a probabilidad $P(n_1, n_2, \ldots n_K)$ es igual a a probabilidad n_1 paquetes en la cola Q_1, n_2 paquetes en la cola Q_2 n_K paquetes en la cola Q_K entonces se tiene:

$$P(n_1, n_2, \ldots n_K) = \rho_1^{n_1}(1 - \rho_1)\rho_2^{n_2}(1 - \rho_2)\ldots\rho_K^{n_K}(1 - \rho_K) \quad (16)$$

Donde $\rho_i = \frac{\lambda_i}{\mu_i}$ con λ_i como la rata de llegada de todos los paquetes a la cola Q_k para $k = 1,2,\ldots,K$

Figura 10. Ejemplo de una Red de Colas Abierta

1.8.2 Red de Colas Abierta.

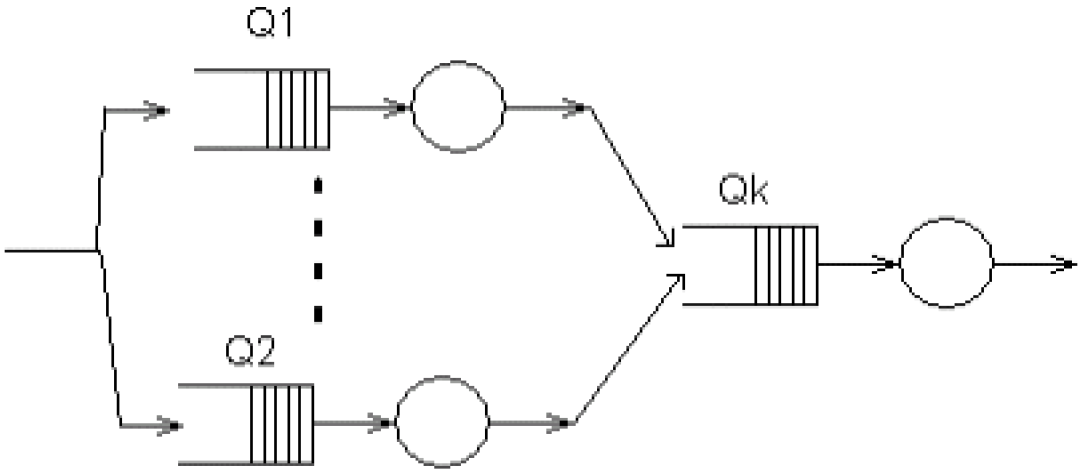


Figura 10. Ejemplo de una Red de Colas Abierta

En una red de colas abierta los clientes llegan del exterior del sistema, circulan por las colas para ser servidos y abandonan el sistema, si se tiene una red abierta la cual consiste de k modelos de cola tipo M/M/1, entonces los clientes que llegan al sistema del exterior se unen a la cola i de acuerdo a un proceso de Poisson con una rata

$$\lambda_i^0$$

después de ser servidos en la cola i según una distribución exponencial con parámetro

$$\mu_i$$

el cliente deja el sistema con una probabilidad p_{i0} o se dirigen a otra cola j con probabilidad p_{ij}.

Se asume que el ingreso a cada cola es primero en ingresar primero en salir (FIFO), para cada cliente el tiempo servicio es independiente del tiempo de servicio en otras colas, para un cliente que sale de una cola i este puede elegir aleatoriamente ir a la siguiente cola j con una probabilidad

p_{ij} o salir de la red con una probabilidad p_{i0} (enrutamiento probabilístico)

Figura 11. Ejemplo de una Red de Colas Cerrada [4]

1.8.3 Red de Colas Cerrada.

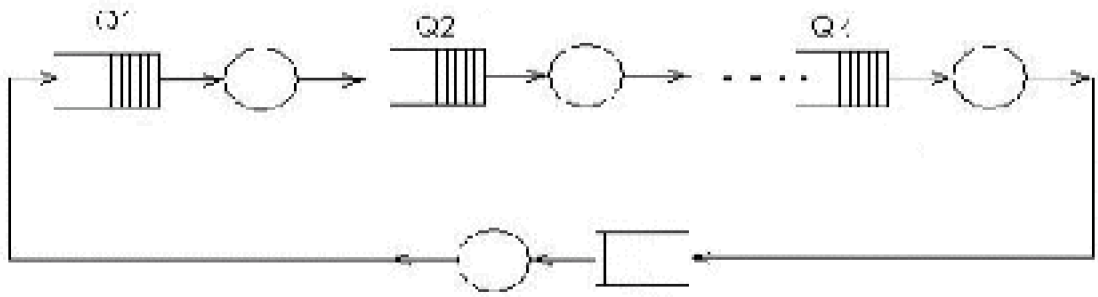


Figura 11. Ejemplo de una Red de Colas Cerrada [4]

Una red de colas es cerrada si un número fijo de clientes circulan indefinidamente y no existe una fuente o sumidero externo, es decir, ningún cliente entra del exterior y ningún cliente abandona la red. Básicamente se tiene una red de M nodos con K clientes en la red, para cada nodo i se presenta un comportamiento FIFO, donde el tiempo de servicio es independiente y sigue una distribución exponencial con parámetro

$$\mu_i$$

ur cliente abandona una cola i e ige una cola j con una probabilidad p_{ij}

Figura 12. Esquema simple de red

1.8.4 Redes de Computadores y las Colas.

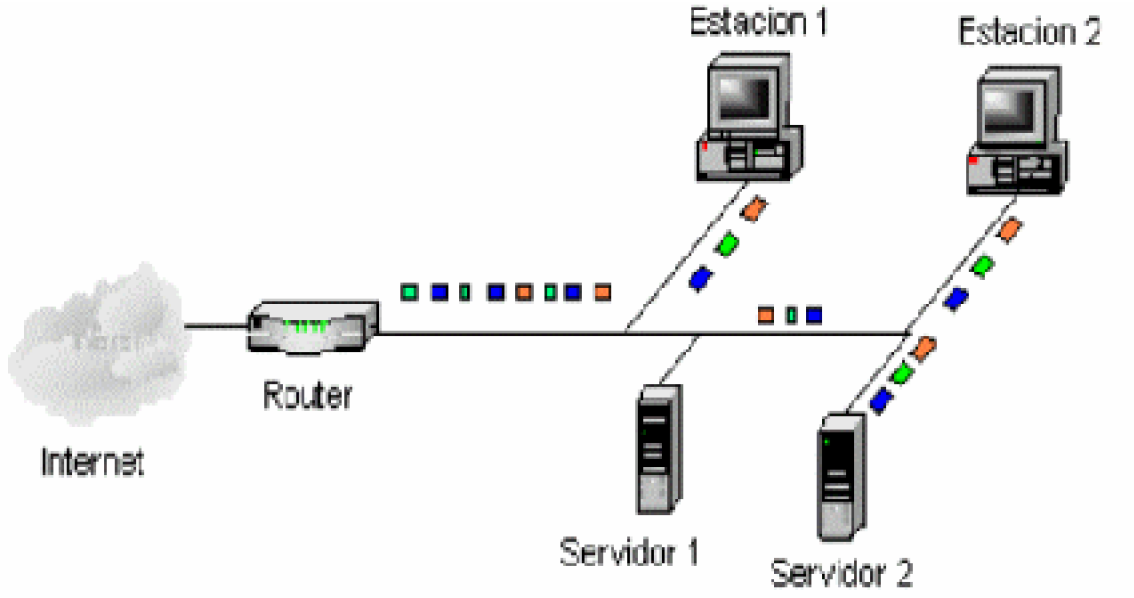


Figura 12. Esquema simple de red

La figura 12 presenta una red LAN que se conecta a través de un router a Internet, se puede apreciar como las estaciones y los servidores generan tráfico tanto al interior de la red como hacia Internet este tráfico debe ser manejado ya sea por las estaciones, servidores y router, y en cada una de estos equipos se generan colas de tramas de datos, es decir, se tiene una red de colas, las cuales deben ser manejadas de alguna forma, la mas típica es tipo FIFO, pero como se verá mas adelante existen muchas otras formas de atender este tráfico, lo importante aquí es la presencia de colas a lo largo de la red, ya sea al interior de una estación debido a que las diferentes aplicaciones que corren solicitan servicio de procesamiento, en los servidores pues deben servir las solicitudes de las estaciones y en el caso del router este debe atender todo el tráfico saliente y entrante de la LAN, en general se dispone de una red de colas, la comprensión de los elementos básicos relacionados con las colas permite entender como se puede controlar el tráfico de datos en una red, pensando en definir algún tipo de prioridad para los diferentes clases de flujo de datos según sus características, es decir, definidos por origen, destino, tipo de aplicación, transferencia de datos, correo electrónico, navegación, consultas a bases de datos, gestión remota de equipos, etc., todos estos elementos pueden ser tenidos en cuenta para realizar el control de dicho tráfico, pues al considerar temas como tamaño de cola, tipo de servicio, tiempo de servicio, mecanismo de atención y prioridad elementos característicos de una cola es posible desarrollar algoritmos que consideren todos estos criterios.

2. Control de tráfico

Se pretende en esta sesión introducir el término de control de tráfico, justificar su uso, identificar ventajas y desventajas de usarlo e introducir algunos conceptos usados en control de tráfico.

2.1 ¿Que es control de tráfico?

Control de tráfico es el nombre dado al conjunto de sistemas de encolado y mecanismos por los cuales los paquetes son recibidos y transmitidos en un enrutador. Esto incluye la toma de decisión de cuales paquetes aceptar y a que rata en la entrada de una interface y determinar cuales paquetes transmitir en que orden y a que rata a la salida de una interface. Dado que los enlaces de red transmiten los datos en una forma serial, una cola es requerida para manejar los paquetes de datos salientes.

En el caso de un PC de escritorio y un servidor WEB compartiendo el mismo enlace hacia Internet, puede ocurrir la siguiente contención de ancho de banda: El servidor WEB puede ser capaz de llenar la cola de salida en el router más rápido de que los datos puedan ser transmitidos a través del enlace. En este punto el router empieza a descartar paquetes (su buffer está lleno). Al mismo tiempo el PC (con un usuario de una aplicación interactiva) se puede enfrentar a pérdida de paquetes y una alta latencia. Si se separan las colas internas que sirven estas dos clases de aplicación, habrá mejor compartimiento

del recurso entre las dos aplicaciones.

El control de tráfico es el grupo de herramientas que permiten a un administrador tener un control granular sobre estas colas y los mecanismos de encolamiento de un dispositivo en red. El término calidad de servicio es usualmente usado como sinónimo de control de tráfico.

2.2 ¿Por que usar CONTROL DE TRÁFICO?

Las redes de conmutación de paquetes difieren de las redes basadas en circuitos en un muy importante asunto: una red de conmutación de paquetes no tiene por si misma estado, una red basada en circuitos debe tener estado dentro de la red. Una red IP es de conmutación de paquetes por diseño y esa “falta de estados“ es una de las fortalezas del protocolo IP, la debilidad de estas redes IP es la falta de diferenciación entre tipos de flujos.

En términos simples el control de tráfico permite a un administrador encolar paquetes de manera diferente, basado en los atributos del paquete. Incluso se puede usar para simular el comportamiento de una red basada en circuitos. Existen muchas razones prácticas para considerar control de tráfico y muchos escenarios en los cuales usarlo tiene sentido. A continuación se relacionan algunos ejemplos de problemas comunes los cuales se resuelven o al menos se amainan mediante el uso de control de tráfico, esto puede dar una idea del tipo de problemas que pueden ser solucionados usando estas técnicas y maximizar el uso de una conexión de red:

- Limitar el ancho de banda a un valor conocido.
- Limitar el ancho de banda a un usuario, servicio o cliente determinado.
- Maximizar el troughput de un enlace asimétrico, priorizar la transmisión de paquetes y ACK.
- Reservar ancho de banda para un usuario o aplicación particular.
- Dar prioridad al tráfico sensitivo a la latencia.
- Permitir distribución equitativa de ancho de banda no reservado.
- Asegurar que un tipo particular de tráfico sea rechazado.

Dentro de las ventajas que ofrece el control de tráfico cuando se emplea apropiadamente, está el lograr tener un uso más predecible de los recursos de la red y una contención menos volátil de los mismos. La red cumple entonces con las metas de la configuración del control de tráfico, para caso de tráfico basura se puede reservar una cantidad razonable de ancho de banda, aun cuando el tráfico interactivo esté simultáneamente servido, incluso al tráfico de baja prioridad como el correo electrónico se le puede reservar ancho de banda sin afectar otros tipos de tráfico. Además si la configuración de control de tráfico representa una política que ha sido comunicada a los usuarios, entonces ellos sabrán que esperar de la red.

En lo que respecta a las desventajas, la complejidad es fácilmente la más significante al usar control de tráfico, sin embargo hay formas de familiarizarse con las herramientas asociadas, las cuales facilitan la curva de aprendizaje acerca de control de tráfico y sus mecanismos. Sin embargo identificar una mala configuración de control de tráfico puede ser todo un reto.

Los recursos computacionales requeridos en un router para soportar control de tráfico, deben ser capaces de manejar el incremento en el costo de mantener las estructuras de control de tráfico, que afortunadamente es un pequeño costo incremental, pero se puede volver más significante a medida que la configuración crece en tamaño y complejidad.

2.3 Colas

Las colas son la base de todo el control de tráfico y son el concepto integral detrás del Scheduling, una cola es una locación (o buffer) conteniendo un número finito de ítems esperando por una acción o servicio. En redes, una cola es el sitio donde los paquetes o unidades esperan a ser transmitidos por el hardware (el servicio). Sin otros mecanismos una cola en si no ofrece ninguna promesa para el control de tráfico, esta se vuelve mas interesante cuando se acopla con otros mecanismos los cuales pueden retrasar, reorganizar, descartar y priorizar paquetes en múltiples colas.

Desde la perspectiva de la capa más alta de software, un paquete es simplemente encolado para transmisión y el sistema de control de tráfico aparece como una simple cola.

2.4 Flujos

Un flujo es una conexión o conversación entre dos hosts, un único set de paquetes entre dos hosts puede ser considerado como un flujo. Bajo el concepto TCP una conexión con una dirección y puerto fuente y una dirección y puerto destino, representa un flujo.

Los mecanismos de control de tráfico frecuentemente separan el tráfico en clases o flujos los cuales pueden ser agregados y transmitidos como un flujo agregado. Los flujos se vuelven importantes cuando se pretende dividir el ancho de banda equitativamente entre un grupo de flujos que compiten por él, especialmente cuando algunas aplicaciones deliberadamente construyen un gran número de flujos.

2.5 Tokens y Buckets

Dos de los puntos claves de los mecanismos de formateo de tráfico son los conceptos interrelacionados de Tokens y Buckets. Para controlar la rata de desencolamiento, una implementación podría contar el número de paquetes o bytes desencolados cada vez que un ítem es desencolado, sin embargo esto requeriría un uso complejo de timers y mediciones muy precisas, en vez de esto un método usado ampliamente en control de tráfico es generar tokens a una rata deseada y solo desencolar paquetes o bytes si un token está disponible.

Considérese la analogía de una diversión de un parque, donde una cola de gente esperando para el paseo, supóngase que hay una pista donde los carts van llegando a la cabeza de la cola a una rata fija. Para poder disfrutar del viaje, cada persona debe esperar por un cart disponible, el es análogo a un Token y la persona es análoga a un paquete. Este es un mecanismo de limitación de velocidad, pues solo cierto número de personas pueden experimentar el viaje en un periodo particular.

Para extender la analogía, supóngase una línea vacía para la diversión del parque y un gran número de carts esperando en la pista listos para llevar gente, si un gran número de gente llega a la línea juntos, muchos (tal vez todos) de ellos podrán experimentar el viaje, ya que los carts están disponibles y esperando. El número de carts disponibles es un concepto análogo al bucket. Un bucket contiene un número de tokens y se pueden usar todos los tokens en un bucket sin preocuparse por el paso del tiempo. Para completar la analogía, los carts (tokens) arriban a una tasa fija y solo están disponibles hasta el tamaño del bucket. Así que el bucket se llena con los tokens de acuerdo a la tasa, y si los tokens no son usados, el bucket se puede llenar.

En el caso de que una cola no necesite tokens inmediatamente, los tokens pueden ser guardados hasta que se necesiten. Guardar tokens indefinidamente, negaría el beneficio de formateo de tráfico, así que estos se guardan hasta que cierto número de tokens ha sido alcanzado. Ahora la cola tiene tokens disponibles para un gran número de paquetes o bytes que necesitan ser desencolados. Estros tokens intangibles se almacenan en un intangible bucket, y el número de tokens que puede ser almacenado depende del tamaño del bucket. Esto significa que un bucket lleno de tokens puede estar disponible en cualquier momento. Un tráfico predecible puede ser manejado por buckets pequeños y los buckets grandes se requieren para tráfico tipo ráfaga, a continuación una definición menos intuitiva de este par de conceptos:

Un flujo token bucket se define por (r,b), r denota la rata a la cual los tokens (créditos) se acumulan y b es la profundidad de la “piscina” de tokens (en bytes).

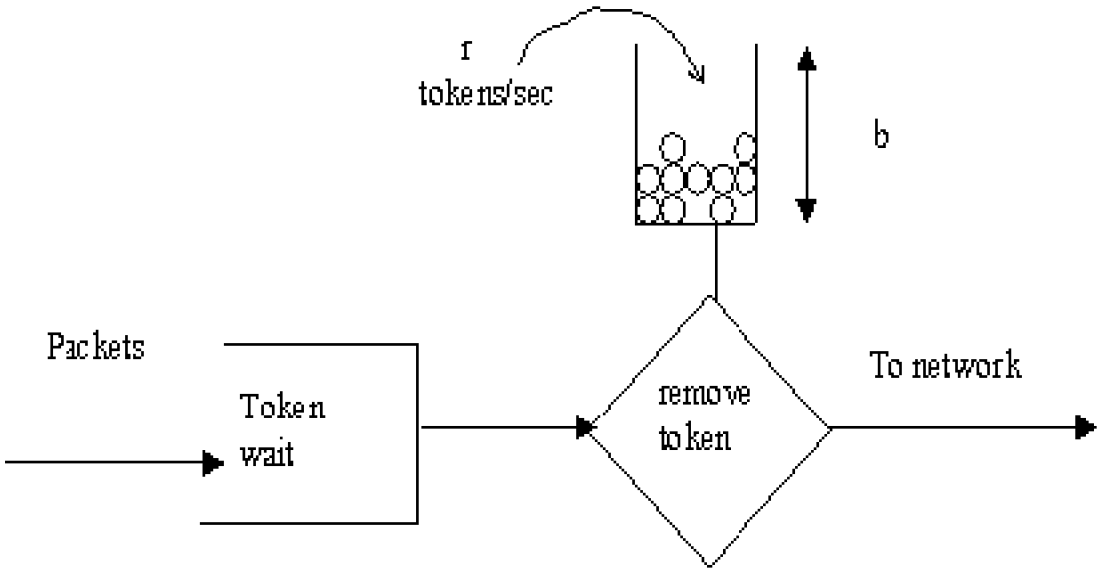


Figura 13. El concepto token y bucket [19]

Como se muestra en la figura anterior, los nuevos token se adicionan al bucket a una rata de r tokens/sec, el máximo de tokens que se puede acumular es de b bytes. Si el bucket está lleno, los tokens que entren serán arrojados afuera. El perfil Token Bucket contiene tres parámetros: una rata promedio, una rata máxima y un tamaño de ráfaga.

Rata promedio: Puede ser deseable limitar la rata promedio a largo plazo a la cual los flujos de paquetes pueden ser enviados a la red. El punto importante aquí es el intervalo de tiempo sobre el cual la rata promedio será controlada. Por ejemplo, un flujo el cual tiene 10000 paquetes por segundo es más flexible que uno que tenga 10 paquetes por milisegundo. Aunque ambos tienen la misma rata promedio, sus comportamientos son diferentes. El flujo que tiene 10000 paquetes por segundo puede enviar 100 paquetes en un milisegundo a largo plazo, pero el flujo que tiene 10 paquetes por milisegundo, no podrá enviar 100 paquetes en un milisegundo a largo plazo.

Rata máxima: Define la máxima rata a la cual los paquetes pueden ser enviados en intervalos cortos de tiempo. En el ejemplo ya mencionado, aunque la rata promedio es de 10000 paquetes por segundo, la rata máxima puede estar limitada a 200 paquetes por milisegundo

Tamaño de ráfaga: Es el máximo número de paquetes que pueden ser transmitidos en un intervalo corto de tiempo. En otras palabras, cuando el intervalo de tiempo se aproxima a cero, el tamaño de ráfaga limita el número de paquetes que puede ser transmitido a la red. Cuando no se considera que el intervalo es cero, el tamaño de ráfaga depende del tamaño del bucket, b, y de la máxima velocidad del enlace de salida.

Uno de los potenciales problemas con Token Bucket es que permite ráfagas y si otra ráfaga llega durante este intervalo de tiempo, no podrá ser manejada, o en el caso de que las ráfagas se vuelvan muy frecuentes, la velocidad de salida será la máxima para el periodo de la ráfaga, lo cual puede causar congestión. Este problema puede ser reducido mediante la adecuada escogencia de los parámetros.

Esencialmente, la red debe asegurarse de que no se deben enviar mas paquetes o bytes de los que se está permitido. Dependiendo de los acuerdos, los paquetes en exceso pueden ser descartados, remarcados, etc.

La operación de Token Bucket en un esquema de diferenciación de servicios puede definirse como sigue:

El arribo de paquetes de L bytes es aceptado (inmediatamente procesado) si hay al menos L tokens en el bucket. Si el número actual de tokens acumulados -b- es menor que el número de paquetes que arriban –L- entonces L – b paquetes no son aceptados (no se procesan inmediatamente), se permiten paquetes a la rata promedio en ráfagas de hasta el tamaño de ráfaga tanto como no se exceda la rata máxima, en cuyo momento se hace un “drenado” del bucket. Si no hay paquetes para ser transmitidos, los tokens se pueden acumular hasta el tamaño del bucket –b-, los otros tokens se desecharán.

Las funciones de aceptación y no aceptación dependerán de los SLA (Service Level Agreement), si los paquetes no son aceptados, las siguientes acciones pueden ser tomadas:

- El paquete puede ser desechado.
- Puede ser marcado en alguna forma particular.
- El paquete puede ser colocado en un buffer (insertando dicho buffer en el medio del punto de entrada y el punto de decisión) y no ser liberado hasta que haya suficiente número de tokens en el bucket.

Estas funciones son aplicadas a diferentes clases en formas diferentes.

2.6 Paquetes y tramas

La palabra trama es típicamente usada para describir una unidad de datos de capa 2 (enlace de datos) que debe ser reenviada al próximo recipiente, interfaces como Ethernet, PPP y E1 denominan su unidad de datos de capa 2 trama, de hecho la trama es la unidad en la cual el control de tráfico es realizado.

Un paquete en cambio, es un concepto de una capa superior, representa unidades de la capa 3 (red), es normal que en documentación sobre control de tráfico se haga referencia a paquetes, aun cundo esto es ligeramente impreciso.

2.7 Traffic Shaping (Ajuste de tráfico)

Una de las desventajas de TCP/IP es la falta de habilidad para garantizar un ancho de banda límite para ciertos servicios y/o usuarios, o en otras palabras: falta de una verdadera funcionalidad de calidad de servicio. A pesar de que hay protocolos como

DiffServ y otras soluciones que han intentado ofrecer QoS en grandes redes, ninguna de las soluciones ha alcanzado un nivel de estandarización tal que los lleve a una utilización a gran escala. Otro hecho es que la mayor parte de las soluciones de QoS son basadas en aplicación, es decir, trabajan con las aplicaciones proveyendo a la red la información de QoS. Desde el punto de vista de seguridad, es inaceptable, que las aplicaciones (los usuarios) decidan la prioridad de su propio tráfico.

La forma mas simple de obtener calidad de servicio en una red, desde perspectivas tanto de seguridad como de funcionalidad, es hacer que los componentes de la red y no las aplicaciones, sean los responsables del control de tráfico en puntos muy bien definidos.

El ajuste de tráfico trabaja midiendo y encolando paquetes IP en tránsito, de acuerdo a un número configurable de parámetros, las acciones que ejecuta son:

- Aplicar límites de ancho de banda encolando paquetes que podrían exceder los límites configurados y enviándolos después de que la demanda momentánea por BW sea menor, es decir, realiza un proceso de retraso de paquetes antes de que salgan para hacer que el tráfico sea conforme a una tasa máxima configurada, este es un proceso que se realiza durante la salida («egress»).
- Descartar paquetes si el buffer está lleno. El paquete a ser descartado debe ser escogido entre aquellos que están causando “el trancón”.
- Priorizar el tráfico de acuerdo a la escogencia del administrador; si el tráfico de mayor prioridad se incrementa mientras una línea de comunicaciones está llena, entonces el tráfico de menor prioridad debe ser temporalmente limitado para hacer espacio al tráfico de alta prioridad.
- Proveer el ancho de banda garantizado. Esto típicamente es alcanzado tratando cierta cantidad de tráfico (la cantidad garantizada) como de alta prioridad, y el tráfico excedente del garantizado como de la misma prioridad del resto, haciendo que este tenga que competir con el resto de tráfico no priorizado.

Los ajustadores de tráfico bien construidos normalmente no trabajan encolando grandes cantidades de datos para después organizar tráfico priorizado y enviarlo antes del no priorizado, mas bien, tratan de medir la cantidad de tráfico priorizado y limitar el tráfico no priorizado dinámicamente, de manera que no interfiera con el troughput del tráfico priorizado.

Los shapers intentan limitar o racionar el tráfico para no exceder una velocidad configurada (frecuentemente medida en paquetes por segundo o bits/bytes por segundo), pero como un efecto secundario pueden suavizar el tráfico por ráfagas, siendo una de las mayores ventajas, la habilidad de controlar la latencia de los paquetes. La gran ventaja del ajuste de tráfico es que puede ser implementado en un dispositivo insertado en cualquier parte del flujo de datos, de manera que en una empresa se podrían usar técnicas de ajuste de tráfico, aun si no se está controlando el core de la red (para suavizar el flujo de tráfico de hacia su ISP, por ejemplo, o para reservar ancho de banda entre aplicaciones o departamentos.

Hay múltiples técnicas que pueden usarse en ajuste de tráfico:

- CBQ (Encolamiento basado en clases) es una forma de setear las ratas máximas convenidas por tipo de aplicación, llamadas clases. CBQ divide el ancho de banda disponible en subcanales, basado en la información de los encabezados IP, por lo tanto los flujos de datos toman ancho de banda solo dentro de su canal. Algunas implementaciones pueden temporalmente distribuir el exceso de capacidad total entre los subcanales. CBQ es de alguna manera similar en su operación a la reservación dinámica de ancho de banda de colas, pero típicamente es empleada en puntos de entrada a la red, en vez de routers de backbone. Es también análoga al mecanismo CIR (Rata de Información convenida) de Frame Relay, excepto que trabaja con tráfico IP en la capa 3 y no en la capa 2 como lo hace Frame Relay.
- Control de velocidad, trabaja directamente con TCP o UDP para manipular las velocidades de transmisión. Para TCP, manipula los ACK cuando pasan a través del dispositivo de control de velocidad. Ajusta el tamaño de la ventana del ACK, inserta nuevos ACK. En UDP los paquetes no usan ACK o ventanas de control de flujo, por lo que un control de flujo con “leaky bucket” puede ser usado para recoger tráfico UDP a medida que arriba y alimentarlo a una rata constante.
- Ajuste de tráfico con Frame Relay: opera en la capa 2 y puede ser un complemento de los mecanismos de capas superiores tanto de ajuste de tráfico como de encolamiento. Se pueden forzar velocidades configurando el CIR a un valor mas bajo que la velocidad del enlace, de manera que cuando hay varios circuitos virtuales en un enlace de acceso Frame Relay, se puede usar el CIR para reservar ancho de banda entre los VC´s.

Que escoger? Si la WAN transfiere video o audio en tiempo real junto con otros datos, entonces hay dos opciones: ATM o una red basada en routers utilizando WFQ.

Si la red lleva mayormente tráfico transaccional, entonces se debe usar una WAN basada en IP con priorización en los routers y clasificación de paquetes basada en señalización implícita, es decir, técnicas de QoS basadas en información ya incluida en los paquetes

2.8 Scheduling (Organizadores o encoladores)

Los encoladores organizan o reorganizan los paquetes para su salida, es decir, Scheduling es el mecanismo mediante el cual los paquetes son organizados o reorganizados entre la entrada y la salida de una cola particular. Desde una perspectiva más grande, cualquier set de mecanismos de control de tráfico en una cola de salida, puede ser considerado como un Scheduler, ya que los paquetes son organizados para su salida. El scheduler más común es el FIFO, aunque existen otros, que buscan compensar varias condiciones de la red, como SFQ, WRR, etc., en la secciones mas adelante se hablará ampliamente de ellos.

2.9 Classifying (clasificadores)

Identificar y categorizar los tipos de tráfico que compiten por un ancho de banda limitado es el primer paso hacia la comprensión y la solución de los problemas de rendimiento de una red. La clasificación es crucial – los controles de rendimiento son útiles solo si se pueden aplicar al tráfico preciso que se tiene en mente.

La clasificación es el proceso mediante el cual los paquetes son separados para diferente tratamiento, posiblemente en diferentes colas de salida. Durante el proceso de aceptar, enrutar y transmitir un paquete, un dispositivo de red puede clasificar el paquete de un número de maneras diferentes y esa clasificación puede incluir el marcado del paquete, lo cual usualmente pasa en los límites de una red que esté bajo un solo control administrativo o la clasificación puede ocurrir en cada salto individualmente.

La clasificación necesita usar un descriptor de tráfico para categorizar un paquete dentro de un grupo específico y hacerlo accesible para el manejo QoS de la red. Usando clasificación, se puede particionar el tráfico de red en varios niveles de prioridad o clases de servicios. Cuando los descriptores son usados para clasificar el tráfico, la fuente acepta unos ciertos términos pactados y la red le promete una cierta calidad de servicio, tanto los traffic policers como los traffic shapers se basan en la clasificación de los paquetes para asegurar adherencia a los términos pactados.

La clasificación de paquetes es un pivote para las técnicas de control que seleccionan paquetes que atraviesan un elemento de red o una interfaz particular para diferentes tipos de servicios QoS. Los métodos de clasificación estuvieron limitados al contenido del encabezado del paquete, sin embargo hoy día, los métodos de marcado para clasificación permiten poner información en los encabezados de las capas 2, 3 y 4 o incluso en el payload del paquete. Los criterios para la clasificación pueden ser tan amplios como “tráfico destinado a la subred X” o tan selectivo como un simple flujo.

Precedencia IP permite especificar la clase de servicio (CoS) de un paquete. Se usan los 3 bits de precedencia en el encabezado IP V4 del campo ToS y se pueden definir hasta 6 clases de servicio. Entonces otros componentes de la red pueden usar estos bits para determinar como tratar el paquete de acuerdo al tipo de servicio garantizado y pueden asignar las políticas adecuadas para el manejo de tráfico, incluyendo manejo de congestión y reserva de ancho de banda. Por ejemplo, aunque la precedencia IP no es un método de encolamiento, varios métodos que si lo son como WRR y WRED, pueden usar la precedencia IP del paquete para priorizar tráfico. Usando los niveles de precedencia en el tráfico entrante y combinando esto con las disciplinas de colas, se pueden crear servicios diferenciados de una manera muy flexible, se pueden asignar precedencias basadas en aplicaciones de usuario o por subred fuente o destino.

Para que cada elemento subsiguiente de red pueda proveer servicio basado en una política o regla determinada, la precedencia IP es usualmente implementada tan cerca del límite de la red o del dominio administrativo como sea posible; se puede ver la

precedencia IP como una función en el límite que permite a las características de QoS, reenviar tráfico basado en CoS. La precedencia IP también puede ser configurada en el host o cliente de red, pero puede ser eventualmente sobrescrita por las políticas y controles de red

Se pueden clasificar los paquetes de acuerdo a políticas basadas en puertos físicos, direcciones fuente o destino IP o MAC, puertos de aplicación, tipo de protocolo IP u otro criterio especificado mediante listas de acceso. Incluso se pueden clasificar paquetes por categorías que son externas a la red, tales como por ejemplo por usuario, de manera que la clasificación de los paquetes puede ser:

- Puertos, rangos de puertos lista de puertos TCP y UDP de capa 4.
- Aplicaciones de capa 7.
- Capa 3: Direcciones IP, rangos de direcciones, subredes, direcciones MAC o listas de hosts.
- Marcas de QoS: DiffServ, IP –ToS, MPLS, etc.
- Sub-clasificación HTTP: por URL, URL wildcard, tipo de contenido, tipo de browser

Después de que un paquete ha sido clasificado, una red puede o bien aceptar o bien sobrescribir y reclasificar el paquete de acuerdo a una política determinada.

2.10 Policing (Control)

Los controladores miden y limitan el tráfico en una cola particular, es el mecanismo mediante el cual el tráfico puede ser limitado. Es más frecuentemente usado en el borde de la red para asegurar que un enlace no consume mas del ancho de banda que tiene asignado. El controlador acepta tráfico a una cierta velocidad y entonces realiza una acción sobre el tráfico que excede dicha velocidad, la solución mas rústica es la de descartar el tráfico aunque este puede ser también reclasificado en vez de descartado.

Y es que los sistemas operativos de equipos de enrutamiento mas comunes usan típicamente dos tipos de mecanismos de regulación: policing y shaping (control y ajuste). Esto a través de características como CAR (committed acces rate o rata de acceso convenida) y Traffic Policing, en el caso de la funcionalidad para policing, mientras que para la funcionalidad de shaping se hace a través de GTS (Generic Traffic Shaping), CBS (Clas- based Shaping), DTS (Distributed Trafic Shaping) y FRTS (Frame Rrelay Traffic Shaping).

Ambos mecanismos (Policing y Shaping) responden de manera diferente ante los paquetes que tienen que afectar, a pesar de que el controlador usa internamente un mecanismo de token bucket, no tiene la capacidad de retrasar un paquete, como si lo hace el mecanismo de ajuste.

Un controlador (policer) típicamente descarta tráfico (Por ejemplo, el controlador CAR de control de velocidad descartará un paquete o rescribirá su precedencia IP, reseteando

el TOS en el encabezado del paquete). Por el contrario el Shaper retrasa el paquete usando un buffer o mecanismo de encolamiento, para retener los paquetes cuando la velocidad de la fuente es mayor que la esperada.

2.11 Disciplinas de colas en proceso de control de tráfico

En las redes de computación uno de los problemas que se presentan es el manejo que se le da a una cantidad de recursos compartidos frente a usuarios, aplicaciones y clases de servicios que compiten por dichos recursos.

Una disciplina de colas, permite manejar el acceso a una cantidad fija de BW de un puerto de salida, mediante la selección del siguiente paquete que será transmitido a dicho puerto. Hay diferentes disciplinas de colas, y cada una apunta a encontrar el balance entre complejidad, control y equilibrio, buscando darle manejo a la congestión. Es así como cada dispositivo de red puede contar con una disciplina de colas asociada, lo cual no es mas que el algoritmo que controla las colas, es decir, como los paquetes encolados en el dispositivo son tratados.

La disciplina de colas mas simple puede consistir solo de una cola, donde todos los paquetes se almacenan en el orden en el que llegaron y la cual se vacía tan rápido como el dispositivo al que sirve lo puede hacer. Disciplinas de colas mas elaboradas pueden usar filtros para distinguir entre diferentes clases de paquetes y procesar cada clase de una forma diferente, por ejemplo, dándole prioridad de alguna clase sobre otras.

Las disciplinas de colas y las clases están íntimamente ligadas: la presencia de clases y su semántica son propiedades fundamentales las disciplinas de colas. En contraste, los filtros se pueden combinar arbitrariamente con las disciplinas de colas y las clases, tanto como la qdisc (queueing discipline) tenga clases para mapear los paquetes. Pero la flexibilidad no termina allí, las clases normalmente no se preocupan de guardar ellas mismas los paquetes, sino que usan otra qdisc que se encargue de eso. Esa qdisc puede ser arbitrariamente escogida de un conjunto de qdisc disponibles y puede tener también clases, las cuales a su vez usan qdisc.

Los paquetes son encolados de la siguiente manera: cuando la función encolar de una qdisc es llamada, revisa los filtros hasta que uno de ellos indica una correspondencia al identificador de clase. Entonces encola el paquete de la clase correspondiente, lo cual usualmente implica el llamado a la función encolar de la qdisc “dueña” de dicha clase. Los paquetes que no concuerdan con los filtros son atribuidos típicamente a alguna clase por defecto.

Típicamente cada clase tiene una cola, pero también es posible que varias clases compartan la misma cola o aun que una sola cola sea usada por todas las clases de la qdisc respectiva. Sin embargo debe notarse que los paquetes no cargan ninguna indicación explícita de a cual clase fueron atribuidos. Por lo tanto, las qdisc que cambian

la información por-clase cuando desencolan paquetes (ej. CBQ), no trabajarán apropiadamente si las colas “internas” son compartidas, al menos que puedan o bien repetir la clasificación o pasar de otra manera el resultado de la clasificación desde encolamiento hasta desencolamiento.

Las tareas básicas que debe cumplir una disciplina de colas son:

- Soportar la distribución del BW a cada uno de las diferentes clases de servicios que compiten por el BW del puerto de salida. En el caso de que algún servicio requiera mas ancho de banda, se le puede asignar mas peso.
- Protección entre clases de servicios, de manera que un pobre desempeño de una clase de servicio en una cola, no afecte a otras clases de servicios en otras colas.
- Hacer manejo inteligente del ancho de banda, es decir, permitir el uso del BW asignado a un tipo de servicio, cuando este no lo usa, a otro clase de servicio.

Las disciplinas de colas tradicionales son:

- Colas FIFO (First In First Out).
- Colas de prioridad (PQ, de Priority Queuing).
- Colas justas (FQ de Fair Queuing).
- Colas justas por pesos (WFQ de Weighted fair Queuing).
- Colas por pesos round-robin o colas basadas en clases (WRR o CBQ).
- DWRR.

2.11.1 Colas FIFO

Es la disciplina de colas más básica, todos los paquetes son tratados igualmente, colocándolos en una cola simple y procesándolos en el mismo orden en que fueron puestos en la cola.

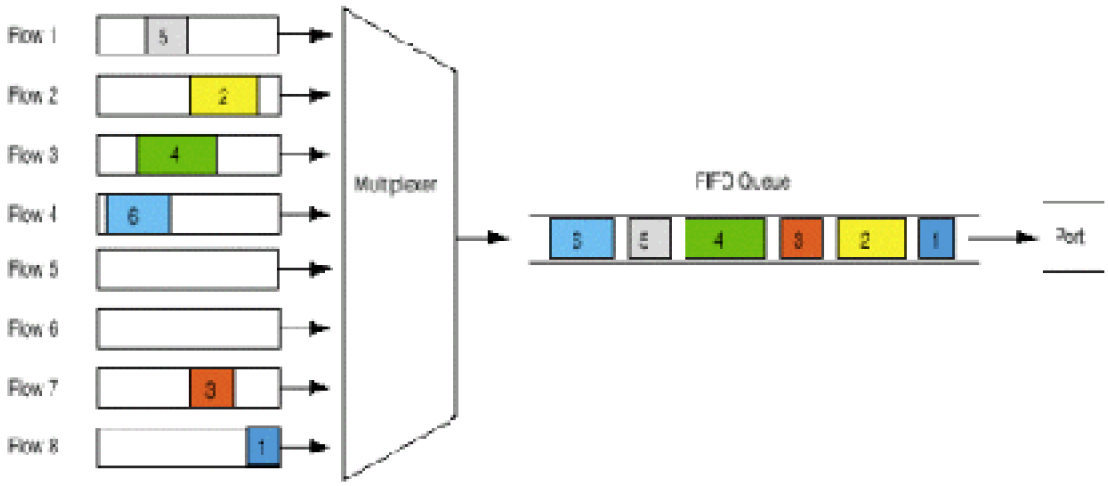


Figura 14. Cola tipo FIFO [9]

Beneficios Colas FIFO:

- Presentan baja carga computacional, comparada con otras disciplinas mas elaboradas.
- Su comportamiento es predecible, los paquetes no son reordenados y el máximo retardo está dado por el largo de la cola.
- Mientras el largo de la cola sea corto, provee una solución simple de contención para los recursos de la red.

Limitaciones colas FIFO:

- Una sola cola FIFO no permite diferenciar una clase de tráfico de otra.
- Impacta todos los flujos igualmente, incrementado su retardo, a medida que la congestión crece. Por lo tanto puede ocasionar perdida de datos de aplicaciones de tiempo real.
- En periodos de congestión, beneficia al tráfico UDP por encima del tráfico TCP, dado que este ultimo baja su rata de transmisión para adaptarse a las condiciones de la red. Por lo tanto, se incrementa el retardo y se disminuye el BW consumido por las aplicaciones TCP que pasan por la cola FIFO.
- Una ráfaga puede consumir todo el espacio de buffer de la cola FIFO, lo que redunda en altos retardos y pérdidas de otros flujos TCP y UDP.

2.11.2 Colas con Prioridad (PQ)

Proveen un método simple para soportar clases de servicios diferenciados. Los paquetes son primero clasificados por el sistema y luego son colocados en diferentes colas de prioridad. Los paquetes son despachados desde la cabeza de una cola determinada si todas las colas de mayor prioridad están vacías. Dentro de cada cola de prioridad, los paquetes son despachados en orden FIFO.

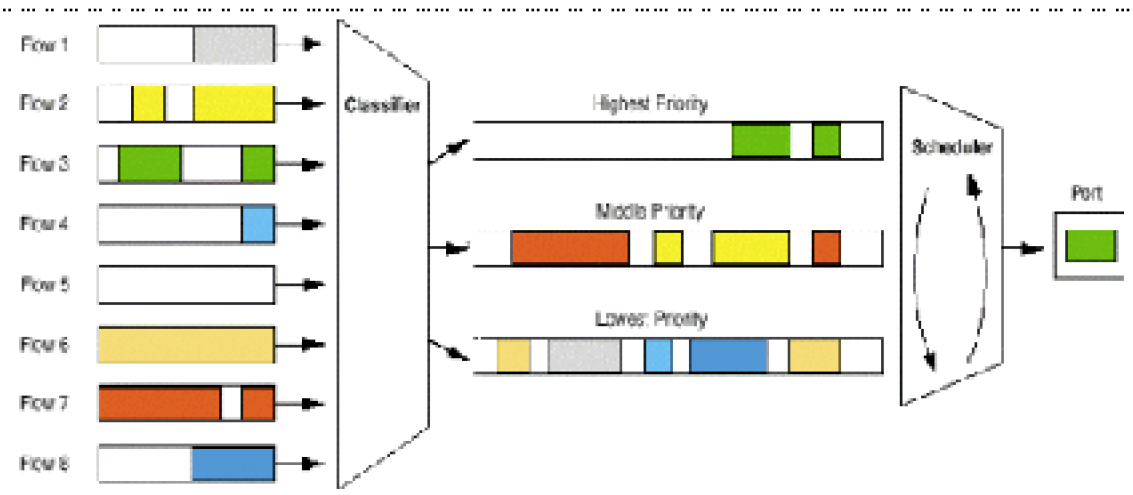


Figura 15. Colas con Prioridad [9]

Beneficios colas con prioridad:

- Presentan baja carga computacional, comparada con otras disciplinas mas elaboradas.
- Permiten dar servicios a un tipo de tráfico de forma diferente a los otros tipos.

Limitaciones:

- Si la cantidad de tráfico de alta prioridad no está controlado mediante políticas o condicionado en los límites de la red, entonces el tráfico de baja prioridad puede experimentar retardos exageradamente altos.
- Si el volumen de tráfico de alta prioridad se vuelve excesivo, entonces el tráfico de baja prioridad puede ser descartado, ya que su buffer se llena.
- PQ no es una solución a la limitación de las colas FIFO, donde se favorecía el tráfico UDP sobre el TCP, en periodos de congestión. Si se trata de usar una cola de mas alta prioridad para el tráfico TCP que para el UDP, entonces el manejo de ventanas TCP y mecanismo de control de flujo terminarán consumiéndose todo el BW disponible del puerto de salida, y por lo tanto dejando sin recursos al tráfico UDP definido como de mas baja prioridad.

Típicamente, se usa PQ en dos modos:

- PQ Estricto: Asegura que los paquetes en una cola de alta prioridad son siempre entregados antes que los paquetes en colas de mas baja prioridad
- PQ con control de rata: Asegura que los paquetes en una cola de alta prioridad son siempre entregados antes que los paquetes en colas de más baja prioridad, solamente si la cantidad de tráfico de alta prioridad se mantiene por debajo de un nivel de umbral configurado.

Existen dos aplicaciones principales de PQ en los límites y core de una red:

- PQ puede mejorar la estabilidad de la red durante periodos de congestión, permitiéndole al administrador asignar protocolo de enrutamiento y otros tipos de control de tráfico de la red a la cola de más alta prioridad.
- PQ soporta la entrega de tráfico que requiera bajos retardo y pérdidas, por lo que se presta a para aplicaciones de tiempo real como voz o video interactivos o emular circuitos TDM.

2.11.3 Colas Justas (FQ)

Están diseñadas para asegurar que cada flujo de datos tenga un acceso justo a los recursos de la red y prevenir que un flujo tipo ráfaga consuma mas de lo justo del ancho de banda disponible. Los paquetes son clasificados en flujos por el sistema y entonces se

asignan a una cola que está específicamente dedicada a ese flujo. Las colas son atendidas entonces un paquete a la vez y en orden round-robin (por turnos) donde las colas vacías son obviadas. Las colas justas también se conocen como colas por flujo.

El principal beneficio de usar este tipo de colas es que un flujo de datos tipo ráfaga no degrada la calidad del servicio de otros flujos, dado que cada uno está aislado en su propia cola y en el caso de que un flujo trate de ocupar mas de su ancho de banda justo, su propia cola será la única afectada y finalmente no habrá impacto sobre el desempeño de las otras colas en el puerto de salida compartido.

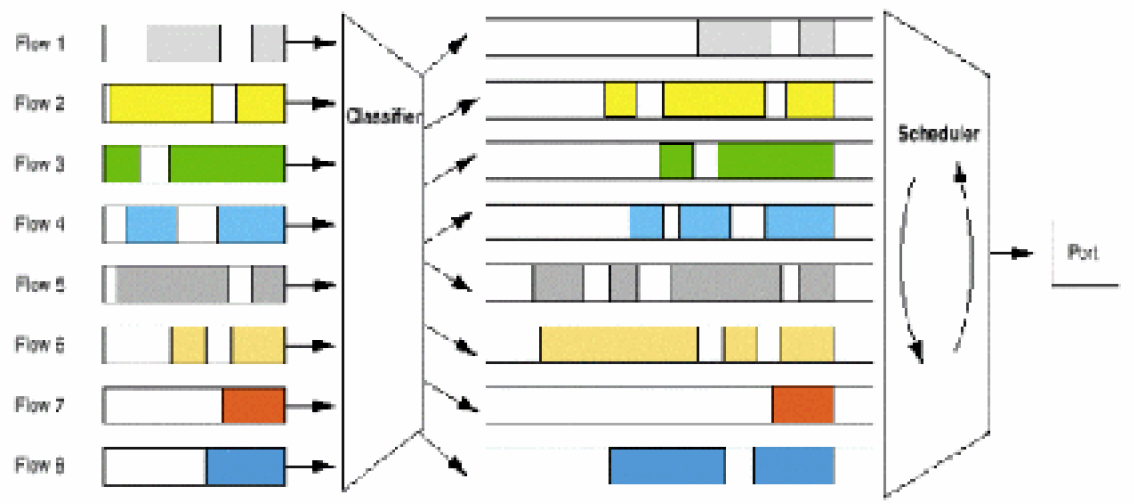


Figura 16 Colas Justas (FQ) [9]

Limitaciones:

- FQ reserva el mismo BW para cada flujo de datos y no soporta un número de flujos con diferentes requerimientos de ancho de banda.
- FQ proporciona el mismo ancho de banda a cada flujo, solo si el tamaño de los paquetes en cada cola, es el mismo. Normalmente un flujo con un tamaño de paquete mayor, obtiene un mayor ancho de banda del puerto de salida que los flujos con menores tamaños de paquetes.
- FQ es sensible a orden de arribo de los paquetes. Si un paquete arriba a una cola vacía inmediatamente después de que ha sido examinada, este paquete deberá esperar en su cola, hasta que las otras colas sean servidas.
- FQ no soporta aplicaciones de tiempo real como VoIP.
- Dependiendo del mecanismo usado para clasificar los paquetes en colas, FQ generalmente, no puede ser implementado en enrutadores de core debido a que en cada puerto se necesitarían miles de colas.

FQ es implementado normalmente en los límites de la red, donde los usuarios se conectan con los proveedores de servicios. Normalmente se clasifican los paquetes en 256, 512 o 1024 colas, mediante una función Hash que se calcula con el par de direcciones fuente / destino, puertos UDP /TCP fuente /destino y el byte ToS. Si se tiene n colas, a cada una le corresponderá 1/n del ancho de banda del puerto de salida.

En colas justas con clases, el puerto de salida es dividido en un número de clases de servicios y cada clase se le asigna un porcentaje del BW total disponible. A cada clase de servicio se le aplica FQ. En la figura 17 se muestra una división en dos tipos de servicios: VoIP (se lea signa el 20%) y tráfico IP (80%). A cada flujo de VoIP, se le asigna un BW igual del 20% disponible y a los flujos IP un BW del otro 80% y no se interfieren entre ellos.

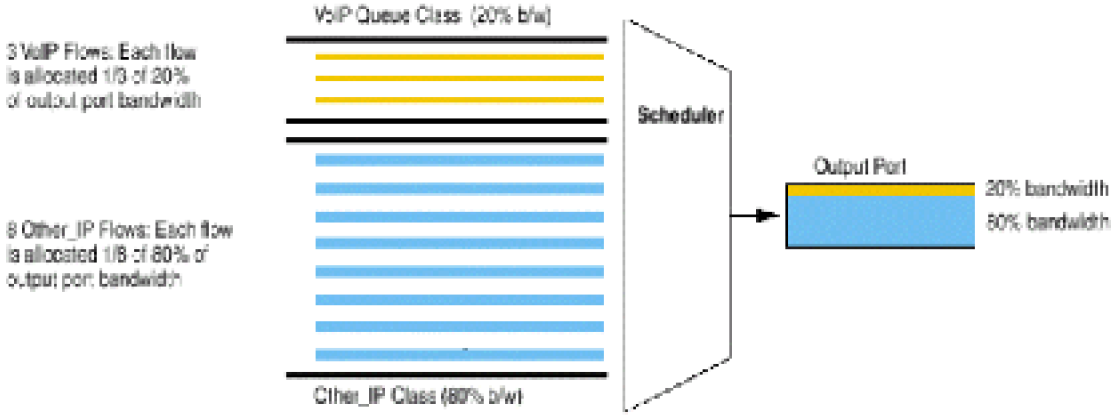


figura 17. FQ Con Clases [9]

2.11.4 FQ con pesos (WFQ, Weighted Fair Queuing).

WFQ soporta flujos con diferentes requerimientos de ancho de banda, dándole a cada cola un peso que le asigna un diferente porcentaje del BW del puerto de salida, también soporta paquetes de longitud variable. Utiliza un esquema donde se hace la transmisión de bits de cada cola de acuerdo al peso dado a la misma.

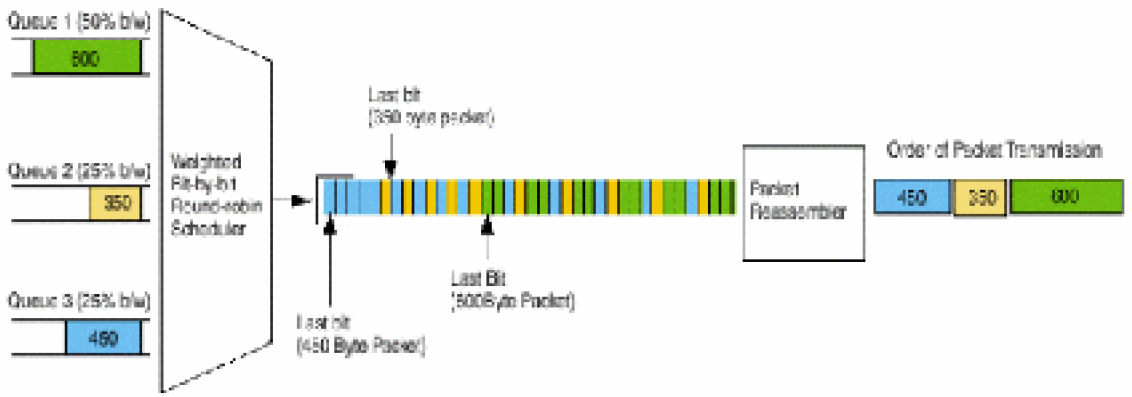


Figura 18. WFQ [9]

Beneficios:

- Provee protección a cada tipo de servicio, pues le garantiza un mínimo nivel de BW del puerto de salida, independiente del comportamiento de las otras clases de servicios.
- Si se combina con condicionamiento de tráfico en los límites de la red, WFQ

garantiza una distribución justa del BW del puerto a cada clase de servicio con un retardo controlado.

- Limitaciones:
- Implementa un algoritmo complejo que requiere el mantenimiento de una cantidad significante de estados de clase y revisiones iterativas de los estados de cada paquete cuando arriba y es despachado.
- Alta complejidad computacional.

WFQ es empleado en los límites de la red para proveer una distribución justa de BW entre un número de diferentes clases de servicios, pudiéndose configurar:

- Para clasificar paquetes en número relativamente grande de colas usando una función hash con los direcciones fuente /destino, los puertos UDP / TCP fuente / destino y el byte ToS de la trama IP.
- Para permitir al sistema manejar un número limitado de colas que llevan flujos de tráfico agregado. Los paquetes se asignan a las colas de acuerdo a políticas de calidad de servicio (QoS) o los 3 bits de menor peso del byte ToS. A cada cola se le asigna un BW diferente, de acuerdo al peso por cada clase de servicio. Esto finalmente permite al sistema reservar diferentes BW a cada cola, de acuerdo a la política de QoS del grupo o reservar cantidades de BW a cada cola, que se incrementa al incrementarse el parámetro del byte ToS.

2.11.5 WRR (Weighted round-robin).

Diseñada para solucionar las limitaciones de los modelos FQ y PQ, pues supera las limitaciones el modelo FQ, soportando flujos de tráfico con anchos de banda significativamente diferentes, de manera que a cada cola se le puede asignar un porcentaje diferente del ancho de banda del puerto de salida y corrige las limitaciones del modelo PQ, asegurando que a las colas de baja prioridad no se les niegue el acceso a espacio de buffer y BW de salida. EN WRR al menos un paquete se mueve de cada cola durante una ronda de servicio.

Los paquetes son primero clasificados en varias clases de servicio y se asignan a colas que están exclusivamente dedicadas a cada clase de servicio.

Soporta la reserva de diferentes cantidades de ancho de banda para cada clase de servicio mediante dos mecanismos:

1. Permitiendo a las colas de alta prioridad enviar mas de un paquete cada vez que es servida en una ronda o

2. Permitiendo enviar un solo paquete en cada servicio a cada cola, pero las colas de alta prioridad son servidas más de una vez en una ronda.

En la figura 19 se aprecia el esquema, en donde al tráfico para transferencia de archivos se le garantiza el 50% del BW, sirviendo su cola dos veces en cada ronda.

Los siguientes son los parámetros que se configuran para regular la cantidad de recurso de red reservados a cada clase de servicio:

El retardo experimentado por los paquetes en una cola dada, está determinado por una combinación de la rata en que los paquetes son puestos en la cola, el largo de la misma, la cantidad de tráfico removido de la cola en cada ronda de servicio y el número de colas restantes servidas

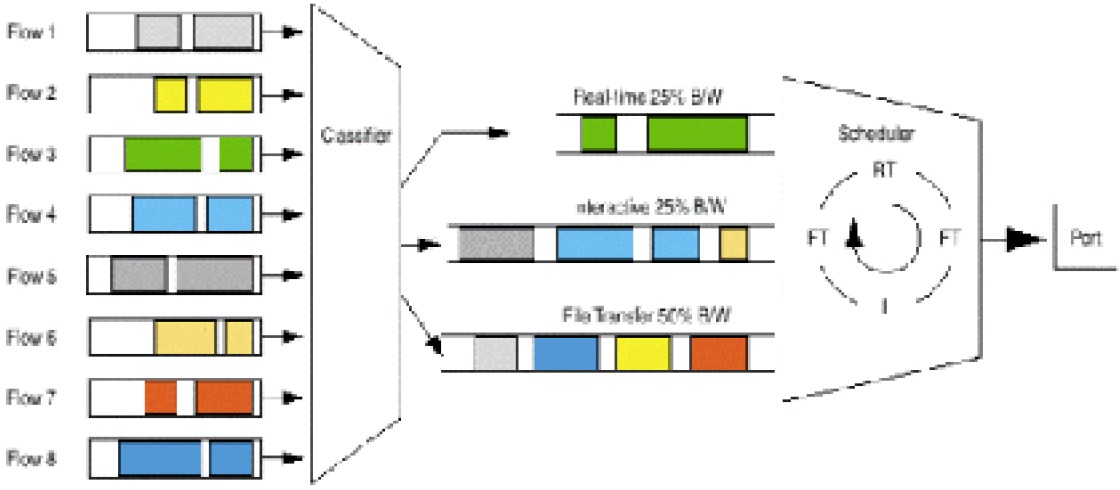


Figura 19 WRR Queuing. [9]

El jitter (o desfase) experimentado por los paquetes en una cola dada está determinado por las variabilidades en los retrasos en todas las colas y la variabilidad en el intervalo entre las rondas de servicio.

La cantidad de paquetes perdidos en cada cola está determinado por una combinación de la rata en que los paquetes son puestos en la cola, el largo de la misma, la agresividad de los perfiles de red configurados en la cola y la cantidad de tráfico servido en cada ronda.

WRR provee un grueso control sobre el porcentaje de BW de salida asignado a cada clase de servicio, asegura que todas las clases de servicio tengan acceso a por lo menos alguna cantidad configurada de BW, provee un mecanismo eficiente para soportar la entrega de servicios diferenciados a un numero razonable de flujos de tráfico agregado y la clasificación del tráfico en clases provee un manejo mas equitativo y estable par aplicaciones de red que el uso de prioridades o preferencias. WRR asume que la reducción de recurso es más conveniente que negación de los mismos para controlar la congestión.

La limitación principal de WRR es que funciona bien solo si el tamaño de los paquetes en todas las colas es el mismo, o cuando el principal tamaño de paquete es conocido previamente. Cuando WRR se enfrenta a tamaños variables de paquetes, no mantiene la distribución configurada de BW de salida.

2.11.6 DWRR (Deficit Weighted Round Robin).

Diseñada para superar las limitaciones de WRR, esta soporta de manera precisa la distribución de BW en colas que contienen paquetes de longitud variable, también supera las limitaciones del modelo WFQ, ya que tiene una baja complejidad computacional, cada cola es configurada con una serie de parámetros:

- Un peso (weight) que define el porcentaje del ancho de banda del puerto de salida reservado para la cola.
- Un Contador de déficit (DeficitCounter) que define el número de bytes que la cola tiene permitido transmitir cada vez que es visitado por el “despachador”. En caso de que el paquete en la cabeza de la cola sea más grande que el valor del contador, no se permite su transmisión y deberá esperar hasta la próxima ronda.
- Un quantum de servicio que es proporcional al peso de la cola y se expresa en bytes. El contador de déficit de una cola se incrementa en un valor igual al quantum cada vez que la cola es visitada por el “despachador”.

El “despachador” visita cada cola no vacía y determina el número de bytes en el paquete en la cabeza de la cola. La variable DeficitCounter se incrementa en por el valor del quantum. Si el tamaño del paquete en la cabeza de la cola es mayor que la variable DeficitCounter, entonces se pasará a atender la próxima cola, por el contrario si el tamaño de dicho paquete es menor o igual que DeficitCounter, entonces la variable DeficitCounter se reduce por el número de bytes en el paquete y este es transmitido en el puerto de salida. El despachador continúa desencolando paquetes y decrementa la variable DeficitCounter en el valor del tamaño del paquete transmitido, esto hasta que o bien el tamaño del paquete en la cabeza sea mayor que el DeficitCounter, o que la cola este vacía. Cuando la cola está vacía, DeficitCounter se pone en cero.

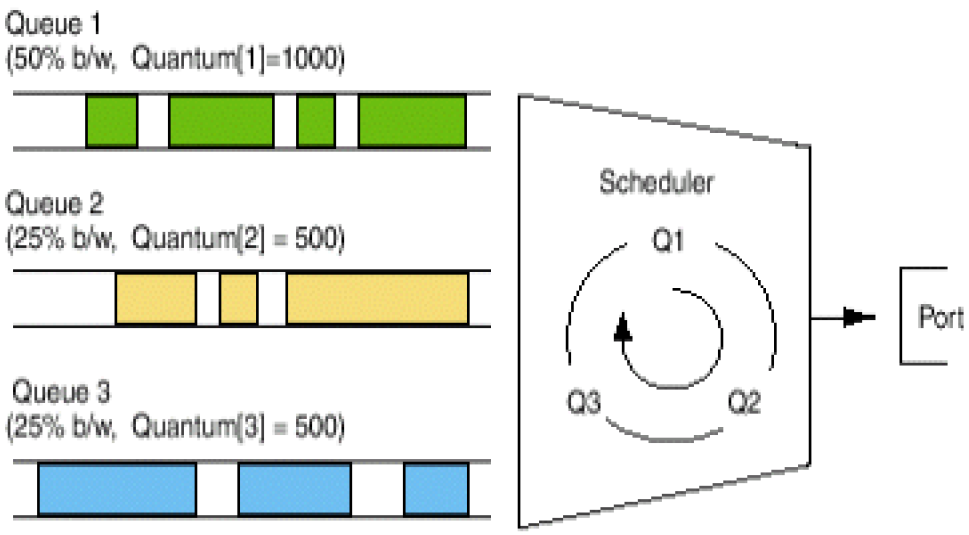


Figura 20. DWRR [9]

Beneficios y limitaciones:

- Provee protección entre flujos de datos, un pobre desempeño de uno de ellos no

afecta a los demás.

- Supera las limitaciones de WRR, al proveer control sobre los ancho de banda de cada cola, incluso cuando manejan tamaños de paquetes diferentes.
- Supera las limitaciones del modelo PQ, asegurando que todas las clases de servicios tengan acceso al menos a su cantidad de ancho de banda configurado.
- Implementa un algoritmo relativamente simple y poco costoso, desde un punto de vista computacional, que no requiere del mantenimiento de un número significante de estados por servicio.
- Esta disciplina de cola, no provee un retardo end to end garantizado tan preciso como otras disciplinas de colas y además puede no ser tan precisa como otras disciplinas, sin embargo la precisión y la distribución del ancho de banda en interfaces de alta velocidad, no es tan crítico como en interfaces de baja velocidad.

3. ENRUTAMIENTO AVANZADO CON LINUX

En los capítulos anteriores se han tratado temas como las colas y el control de tráfico en redes de computadores en general, en el presente capitulo se abordará la temática relacionada con las capacidades de enrutamiento avanzado del sistema operativo Linux. Específicamente se hablará de la versión 2.4.x.+. Se entiende entonces que el concepto de enrutamiento avanzado está relacionado con la posibilidad de explotar al máximo las herramientas con que cuenta Linux para efectuar tareas de gestionar tráfico entre diferentes redes, es decir, que este sistema operativo cuenta con todo lo necesario para realizar tareas similares e incluso superiores a un enrutador comercial, tales como tener varias tablas de enrutamiento, reservar ancho de banda, brindar calidad de servicio (QoS), balancear carga, túneles, cortafuegos, multicasting, etc.

Con Linux a nivel de enrutamiento es posible realizar tareas tales como:

- Controlar el ancho de banda de algunos computadores
- Controlar el ancho de banda hacia algunos computadores
- Compartir el ancho de banda de manera justa
- Proteger la red de ataques de negación de servicios (DoS)
- Proteger a Internet de sus clientes
- Balancear la carga de múltiples servidores

- Restringir el acceso a sus computadores
- Restringir el acceso de sus usuarios a otros computadores
- Enrutamiento basado en la identificación del usuario, dirección MAC, dirección fuente, puerto, tipo de servicio, hora del día y contenido.

Todas estas posibilidades que brinda el kernel de Linux lo hacen viable para considerarlo como una opción cuando se trata de ofrecer soluciones a nivel de redes de computadores, pues dentro del conjunto de sistemas operativos se convierte en un fuerte competidor, gracias a las características de enrutamiento, clasificación y filtrado.

3.1 Enrutamiento

En principio cuando se habla de enrutamiento se piensa en tablas para enrutar el tráfico entre diferentes redes, tradicionalmente enrutar tráfico IPv4 es un proceso de dirigir los paquetes basados en su destino, para ello Linux cuenta con dos herramientas básicas para realizar dicha tarea que son ifconfig y route, estas basan su funcionamiento en la dirección destino del paquete, es decir, se verifica cual es su destino para, basados en la tabla de enrutamiento, reenviar el paquete a su destino, con estos comandos se busca configurar las interfaces e inicializar las tablas de enrutamiento.

Se tratará brevemente estas dos utilidades para mostrar posteriormente la potencia de la utilidad iproute2, el interés está en mostrar que cuando se busca enrutar paquetes diferentemente, se requiere pensar en políticas de enrutamiento, las cuales están relacionadas con enrutar los paquetes no solo basados en su dirección destino sino también se pueden considerar la dirección fuente, el protocolo IP, los puertos del protocolo de transporte o el contenido del paquete, es decir, pensar en políticas de enrutamiento es proveer capacidad de enrutamiento basados en una o todas las facetas de un paquete.

El núcleo de la política de enrutamiento está cimentada en tres elementos, estos son dirección, ruta y regla.

- Dirección: define la localización del servicio
- Ruta: establece la ubicación de la dirección
- Regla: describe la localización de la ruta

Lo interesante de definir una política de enrutamiento es emplear estos tres elementos de manera combinada, ofreciendo así mayor flexibilidad y posibilidad de abordar tareas más complejas.

La dirección se refiere a la ubicación del servicio o conjunto de servicios, la ruta tiene que ver con cómo alcanzar un destino, cuando se considera en términos de políticas de enrutamiento, tiene características avanzadas, finalmente regla es relacionada con la definición de listas de acceso a rutas, es decir permite definir filtros a los cuales están

sometidos los paquetes.

Hablar de Linux en términos de enrutamiento avanzado, se refiere a definir estructuras de políticas de enrutamiento a través de la base de datos de políticas de enrutamiento (RPDB – Routing Policy DataBase), que es un conjunto de rutas, tablas de rutas y reglas, RPDB es parte integral de Linux desde el kernel 2.2, y es una lista lineal de reglas ordenadas por un valor numérico que define la prioridad, básicamente provee una estructura interna y un mecanismo para implementar las reglas en la política de enrutamiento.

RPDB permite verificar características de los paquetes tales como dirección fuente, dirección destino, interface de entrada, TOS y emplear valores fwmark para verificar protocolos IP y puertos de transporte.

En RPDB cada regla de enrutado de la política está formada por un selector y una acción, lo que se hace al interior de la base de datos es revisar en orden de incremento de la prioridad con el selector de cada regla aplicada a la dirección fuente, dirección destino, interface de entrada, TOS y fwmark, si el selector halla la coincidencia entonces la acción es realizada, de lo contrario se evalúa la siguiente regla.

RPDB puede contener reglas de los siguientes tipos:

- Unicast: Retorna la ruta encontrada en la tabla de enrutamiento referenciada por la regla.
- Blackhole: Rechaza un paquete silenciosamente.
- Unreachable: Genera un error red inalcanzable.
- Prohibit: Genera un error comunicación administrativamente prohibida.
- Nat: traduce la dirección fuente del paquete IP a otro valor.

Otra característica de RPDB es su compatibilidad con las utilidades estándares de red tales como ifconfig y route, pero no se requiere hacer uso de estas, pues basta la utilidad ip que es superior a estas. En Linux se puede hacer uso de características extendidas de la política de enrutamiento: basta con compilar el kernel para que de soporte a NETLINK y RT_NETLINK.

La RPDB constituye el núcleo que provee la funcionalidad alrededor de la cual la política de enrutamiento y las características de enrutamiento avanzado pueden ser construidas.

Para el caso de ifconfig, esta se viene empleando desde los sistemas operativos UNIX, para Linux no es una excepción, su sintaxis en general es como sigue:

ifconfig {interface} {IP-address} {netmask} broadcast {broadcast}

Lo que hace ifconfig es hacer accesible la interface a la capa del kernel relacionada con el manejo de red, esto involucra activar la interface, asignarle una dirección de red y otros parámetros.

Considerando route, este comando lo que hace es agregar o retirar rutas de la tabla de enrutamiento del kernel, su sintaxis general es:

route add -net(-host) {IPv4 Address} netmask {netmask} gw {gateway} dev {interface}

Con estas dos utilidades no es posible llevar a cabo tareas mas interesantes tales como las anteriormente mencionadas, por lo tanto, para hacer uso de las características con que cuenta Linux es necesario mencionar, netfilter con su utilidad iptables e iproute2 con sus utilidades tc e ip.

A continuación de manera breve se expondrán los temas de enrutamiento avanzado con Linux, tales como cortafuegos (firewall), túneles GRE, balanceo de carga, tablas de enrutamiento múltiple, multicasting y reserva de ancho de banda.

3.2 IPTABLES

Iptables es una utilidad de Netfilter usada para establecer reglas para filtrado de paquetes, definición de cortafuegos (firewalls), está basada en un filtrado puro de paquetes

Haciendo uso de la utilidad iptables de Netfilter, es posible filtrar los paquetes o marcarlos, los paquetes pueden ser etiquetados con un número, en concreto con iptables es posible filtrar el tráfico por:

- Dirección fuente
- Dirección destino
- Puerto fuente
- Puerto destino
- Identificación de protocolo

La sintaxis típica para generar una regla con iptables es:

iptables [-t table] command [matches] [target/jump]

Tablas en iptables son:

- mangle: Usada para el marcaje de los paquetes, para cambiar su contenido, cambiando sus encabezados.
- filter: Empleada para filtrado de paquetes
- nat: Para realizar traducción de direcciones

Los Comandos son:

- -A, --append: para agregar una regla.
- -D, --delete: para borrar una regla.
- -R, --replace: para reemplazar una entrada en una cadena

- -I, --insert: para insertar una regla en algún lugar de la cadena
- -L, --list: lista todas las entradas en una cadena especifica
- -F, --flush: limpia todas las reglas de una cadena especificada
- -Z, --zero: pone los contadores a cero en una cadena específica o en todas las cadenas.
- -N, --new-chain: le dice al kernel cree una cadena de un nombre especifico en la tabla especificada
- -X, --delete-chain: borra la cadena especificada de la tabla, para poder hacer uso de este comando no debe existir ninguna regla que se refiera a dicha cadena.
- -P, --policy: le dice al kernel que ponga una etiqueta especifica o política por defecto sobre una cadena
- -E, --rename-chain: para cambiar el nombre de una cadena

Junto a estos comandos existen unas opciones empleadas para mostrar o expandir algunas características de los comandos.

Las igualaciones o matches tienen que ver con que condiciones el tráfico es igualado para verificar si cumple o no la regla, veamos algunas:

- -p, --protocol: chequeo por protocolo
- -s, --src, --source: igualación por dirección fuente
- -d, --dst, --destination: por dirección destino
- -i, --in, --in-interface: por interface de entrada
- -o, --out, --out-interface: por interface de salida
- -sport, --source-port: puerto fuente
- -dport, --destination-port: puerto destino

Las etiquetas o saltos (targets/jumps), le dicen al kernel que debe hacer con un paquete cuando este coincide con la sección match de la regla, por ejemplo se tiene:

- ACCEPT
- DROP
- REJECT
- MARK
- TTL
- TOS
- SNAT
- DNAT
- ULOG
- MASQUERADE

Finalmente nos falta nombrar las cadenas con que cuenta iptables:

- TCP
- UDP
- ICMP
- INPUT
- OUTPUT
- FORWARD
- PREROUTING
- POSTROUTING

En general para formar la cadena su sintaxis es como sigue:

iptables [-P {chain} {policy}]

La política es usada por defecto cada vez que un paquete no encuentra una regla aplicable en la cadena.

Veamos un ejemplo:

iptables -A PREROUTING -i eth0 -t mangle -p tcp –dport 25 –j MARK \

--set-mark 1

La regla nos dice que se realizo un marcado de todos los paquetes destinados al puerto 25.

En este caso la etiqueta MARK es utilizada para marcar valores asociados a los paquetes y es válida solamente con la tabla mangle, esta puede ser usada en unión con iproute2 para aprovechar características de enrutamiento avanzado de Linux, por ejemplo para enviar diferentes paquetes a través de diferentes rutas, usando diferentes disciplinas de colas. Para el ejemplo se debe anotar que el registro de la marca está asociado dentro del kernel al paquete, esta no está dentro del paquete, para lograr que viaje con el paquete se debe entonces marcar el campo TOS del paquete, dicho campo consiste de 8 bits, esto nos permite definir diferentes servicios y sobre estos servicios crear reglas.

Lo importante es que al usar la tabla mangle de iptables es posible hacer uso de características avanzadas para el enrutamiento de tráfico, las etiquetas válidas únicamente con esta tabla son: TOS, TTL y MARK, la etiqueta TOS como se explicó arriba brinda la posibilidad de actuar sobre el campo TOS (Type of Service) del paquete, permitiendo definir políticas para el enrutamiento de los paquetes, tales como minimizar el retraso, maximizar la velocidad, minimizar el costo, servicio normal, maximizar la transferencia, maximizar la fiabilidad, la etiqueta TTL se emplea para modificar el campo TTL (Time To Live) del encabezado IP.

Iptables es una utilidad que permite definir un conjunto de reglas para filtrar el tráfico que circula por lo menos entre dos redes, permitiendo con este filtrado proteger la máquina que hace esta labor y las redes detrás de esta, adicional a esto combinado con otras utilidades como ip y tc de iproute2, es posible desarrollar tareas de enrutamiento

avanzado, al marcar los paquetes para luego ser tratados con las herramientas mencionadas.

Para seguir revisando las otras posibilidades de enrutamiento a continuación se hablará de iproute2.

3.3 IPROUTE2

La posibilidad de disponer de la herramienta iproute2 en el sistema operativo Linux, en especial en versiones de kernel 2.4.x.+ , brinda opciones como túneles, multicasting, control del tráfico, múltiples tablas de enrutamiento, balanceo de carga, etc.

Se mostrarán las posibilidades de iproute2 con sus utilidades ip y tc para tareas de túneles, múltiples tablas de enrutamiento y se trata en mas detalle lo relacionado con el control de tráfico, que es el interés del presente trabajo.

Gracias a que a partir de la versión de kernel 2.2, se cuenta con el socket NETLINK, que en principio permite desarrollar en espacio de usuario código para gestión de paquetes y tráfico IP, llevó a que Alexey Kuznetsov desarrollara el código iproute2, el cual permite hacer entre muchas cosas las siguientes:

- Unificar los comandos relacionados con la gestión del tráfico IP
- Monitoreo de los periféricos, direcciones y rutas
- Gestión de tablas ARP
- Uso de tablas de enrutamiento múltiples
- Creación de túneles IP
- Reserva de ancho de banda

Inicialmente se hará referencia a la utilidad ip, entendido como un comando dentro de iproute2, siendo su sintaxis:

ip [OPTIONS] OBJECT [COMMAND [ARGUMENTS]]

OPTIONS se refiriere a un conjunto de modificadores que afectan el comportamiento y la salida de la utilidad ip, todas las opciones empiezan con el carácter “-“ y pueden ser usados en forma larga o corta. Las opciones disponibles son:

- -V, -Version: imprime la versión de la utilidad ip.
- -s, -stats, -statistics: entrega como salida mayor información.
- -f, -family {inet, inet6, link}: forza el uso de una familia de protocolos.

OBJECT tiene que ver con el tipo de objeto con el que se desea operar u obtener información, estos son:

- link: dispositivo de red físico o lógico.

- address: direccion IPv4 o IPv6 sobre el dispositivo.
- neighbour: entrada cache ARP o NDISC.
- route: entrada a tabla de enrutamiento.
- rule: regla en la RPDB.
- maddres: direccion multicast.
- mroute: entrada al caché del enrutamiento multicast.
- tunnel: tunel sobre IP.

COMMAND especifica la acción a realizar sobre el objeto, el conjunto de acciones depende del objeto sobre el que se actúa, los más típicos son add, delete y show, no todos los objetos permiten todas las acciones anteriores y tienen comandos adicionales. El comando help está disponible para todos los objetos el cual muestra los comandos disponibles y la sintaxis.

ARGUMENTS es una lista de opciones de comando específica del comando, estos argumentos dependen del comando y del objeto, existen dos tipos de argumentos: flags (consisten de una sola palabra clave), parameters (consisten de una palabra clave seguida de un valor).

Se presentan enseguida algunas estructuras de uso de la utilidad ip, para cualquier ampliación de manejo de los diferentes objetos, remitirse a la bibliografía.

ip link: utilizado para realizar la configuración del dispositivo de red, aquí link se refiere al dispositivo de red, permite con el correspondiente comando ver y manipular el estado de los dispositivos de red, los comandos para el objeto link son set y show. Con set se cambian los atributos del dispositivo, mientras que con show se presentan sus atributos.

Los argumentos para set son:

- dev NAME: aquí NAME especifica el dispositivo de red sobre el que se opera.
- up / down: cambia el estado del dispositivo.
- arp on / arp off: cambia las banderas de status del ARP sobre el dispositivo.
- multicast on / multicast off: cambia la bandera de estado de multicast en dispositivo.
- dynamic on / dynamic off: cambia la bandera DYNAMIC sobre el dispositivo.
- name NAME: cambia el nombre del dispositivo
- txqueuelen /txqlen NUMBER: cambia la longitud de la cola de transmisión del dispositivo.
- mtu NUMBER: cambia el MTU del dispositivo.
- address LLADDRESS: cambia la dirección de estación del dispositivo.
- broadcast LLADDRESS, brd LLADDRESS, peer LLADRESS: cambia la dirección broadcast nivel de enlace o la dirección peer en caso de una interface punto-punto.

Los argumentos de show:

- dev NAME (default): NAME especifica el dispositivo de red a mostrar, si este argumento es omitido se presentan todos los dispositivos de red.
- up: presenta solo el dispositivo que este corriendo.

ip address: protocolo para gestión de dirección, aquí address se refiere a un protocolo (IPv4, IPv6) de dirección ligado al dispositivo de red, cada dispositivo debe tener por lo menos una dirección para usar el correspondiente protocolo, es posible tener múltiples direcciones ligadas a un dispositivo. El comando ip address permite sobre una interface ver sus direcciones y sus propiedades, agregar o borrar direcciones en cualquier orden.

Los comandos permitidos sobre el objeto address son add, delete, flush, y show. Con add se establece una nueva dirección, delete borra una dirección, flush borra las direcciones y show lista las direcciones del protocolo.

Argumentos del comando add y delete:

- dev NAME: nombre del dispositivo al que se le agrega una dirección
- local ADDRESS (default): direccion de la interface.
- peer ADDRESS: direccion de un extremo remoto para interfaces punto-punto.
- broadcast ADDRESS: dirección broadcast de una interface.
- label NAME: cada dirección puede ser etiquetada con una etiqueta string.
- scope SCOPE_VALUE: área de alcance dentro de la cual la dirección es válida.

Para el comando show los argumentos son:

- dev NAME: nombre del dispositivo.
- scope SCOPE_VALUE: lista las direcciones con el alcance definido.
- to PREFIX: lista solo las direcciones que coincidan con el prefijo
- label PATTERN: lista solo las direcciones con las etiquetas que coincidan con el patrón.
- dynamic / permanent: solo IPv6, lista únicamente direcciones instaladas debido a una configuración no orientada al estado o lista las direcciones permanentes (no dinámicas).
- tentative: solo Ipv6, lista las direcciones que pasaron la detección de duplicadas.
- deprecated: solo IPv6, lista solo las direcciones caducadas.
- primary / secondary: lista las direcciones primarias o secundarias.

Para el caso del comando flush este tiene los mismos argumentos que show, este borra todas las direcciones seleccionadas según un criterio, la diferencia radica en que este no opera sin un argumento, los comandos flush son peligrosos pues nunca preguntan confirmar el comando, llevando a posibles errores de borrado total sin posibilidad de recuperación.

ip neighbour: Los objetos de la tabla neighbour establecen enlaces entre las

direcciones en cada protocolo y las direcciones a nivel de enlace para los hosts que comparten el mismo enlace físico. Los objetos neighbour son organizados en tablas, en IPv4 estos objetos son conocidos bajo otro nombre como tabla ARP. Los comandos permiten ver la tabla neighbour, enlaces y propiedades, agregar o borrar entradas a la tabla, estos son add, change, replace, delete, flush y show.

ip route: para gestión de tablas de enrutamiento, gestiona entonces las rutas de entrada dentro de las tablas de enrutamiento de kernel, estas tablas mantienen la información acerca de la rutas de los protocolos a otros nodos de la red.

Cada ruta de entrada tiene una llave formada por el prefijo del protocolo, el cual está formado por el par dirección de red y longitud de mascara de red y opcionalmente el valor TOS. Un paquete coincide con una ruta si los bits superiores de la dirección destino del paquete son iguales al prefijo de la ruta o al menos el prefijo de la longitud y si el campo TOS de la ruta es cero o igual a el TOS del paquete, en resumen una ruta es identificada por la triada prefijo, TOS y preferencia.

Cuando existan múltiples rutas con las que coincide el paquete, se siguen las siguientes reglas de corte, seleccionando la mejor de ellas:

1. Se seleccionan aquellos paquetes cuyo prefijo sea el mas largo, el resto es descartado.

2. Si el campo TOS de aquellas rutas con el prefijo más largo tienen es igual al campo TOS del paquete, las rutas con diferente campo TOS son descartadas.

3. Si no existe una coincidencia exacta del campo TOS y existen rutas con el campo TOS igual a cero, el resto de rutas son descartadas, de lo contrario la búsqueda de ruta falla.

4. Si se mantienen muchas rutas después de haber aplicado los pasos 1 a 3, entonces las rutas con la mejor preferencia son seleccionadas.

5. Si aun existen muchas rutas entonces la primera de ellas es seleccionada.

Las rutas presentan atributos, que son requeridos como información de enrutamiento para enviar los paquetes IP, estos atributos son: dispositivo de salida, enrutador en próximo salto, MTU y dirección fuente preferida para comunicación con el destino, estos atributos dependen del tipo de ruta, la mas importante es la unicast, pero existen otros tipos veamos:

- unicast: la entrada a la ruta describe los caminos a los destinos cubiertos por el prefijo de ruta.
- unreachable: define destinos inalcanzables, los paquetes son descartados y se genera un mensaje ICMP de error “host unreachable”.
- blackhole: para destinos inalcanzables, los paquetes son descartados silenciosamente.
- prohibit: para destinos a los que nos se puede llegar, los paquetes son descartados y se genera un mensaje ICMP “communication administratively prohibited”.
- local: los destinos son asignados a este host, los paquetes están en lazo cerrado y

son enviados localmente.

- broadcast: los destinos son direcciones broadcast, los paquetes son enviados como enlace broadcastthrow: ruta de control especial usada junto con las reglas de la política, si una ruta es seleccionada como throw entonces la búsqueda en esta tabla en particular es terminada, pretendiendo que la ruta no fue encontrada.
- nat: ruta especial NAT, los destinos cubiertos por el prefijo son considerados como direcciones dummy (externas), lo cual requiere de una traducción a una real (interna) antes del reenvío.
- anycast: no implementada, los destinos son direcciones anycast asignadas para el host, son equivalentes a una dirección local con la diferencia que no pueden ser utilizadas como una dirección fuente de ningún paquete.
- multicast: es un tipo especial empleada para enrutamiento multicast.

Linux brinda la posibilidad de tener múltiples tablas de enrutamiento identificadas por un numero que va del 1 al 255 o por nombre tomado del archivo /etc/iproute2/rt_tables, por defecto normalmente todas las rutas son insertadas en la tabla main (ID 254) y el kernel usa esta tabla cuando se trata de calcular la rutas.

Los comandos aceptados por el objeto route son: add, replace, change, delete, show, flush y get.

ip rule: gestiona la base de datos de enrutamiento de políticas, las reglas en este base de datos controlan la selección del algoritmo de ruta. Los algoritmos de enrutamiento en Internet basan sus decisiones únicamente en la dirección destino y teóricamente en el campo TOS, es deseable de manera diferente, haciendo uso de los campos del paquete como dirección fuente, protocolo IP, puertos del protocolo de transporte o contenido del paquete, esto es lo que se conoce como política de enrutamiento.

Cuando arranca el kernel este configura un RPDB consistente de tres reglas:

1. Prioridad 0: aquí se tiene un selector con coincidencia cualquiera, lo que hace es buscar en la tabla de enrutamiento local (ID 255), esta tabla es especial pues en ella se tiene el control de rutas de alta prioridad para direcciones locales y broadcast. La regla 0 es especial y no puede ser borrada o sobrescrita.

2. Prioridad 32766: el selector es cualquier coincidencia, lo que realiza es buscar en la tabla main (ID 254), esta tabla tiene que ver con el enrutamiento normal y contiene las rutas sin políticas, esta regla puede ser borrada o sobrescrita.

3. Prioridad 327667: el selector es cualquier coincidencia, la acción a realizar es buscar la tabla de enrutamiento default (ID 253), esta tabla está vacía y es reservada para post-procesamiento, si el paquete no se selecciona por las reglas previas.

Las reglas pueden apuntar a diferentes tablas de enrutamiento o muchas reglas pueden referirse a una tabla de enrutamiento y algunas tablas es posible que no tengan reglas que apunten a ellas. En general si se borran las reglas que apuntan a una tabla esta continúa existiendo, para borrar una tabla se deben borran todas las rutas que contiene.

El tipo de reglas que contiene el RPDB son:

- unicast
- blackhole
- unreachable
- prohibit
- nat

Con las reglas de enrutamiento aparece el concepto de dominios (realms) que básicamente se aplica cuando se tienen ambientes de enrutamiento complejo, se habla de entonces de meta-rutas que describen a gran escala grupos de destino, la palabra clave en la utilidad ip es realm. Realm se entiende como la definición de un conjunto de rutas seleccionadas por la lógica humana, básicamente permite clasificar un conjunto de rutas para comprenderlas mejor.

Los comandos aceptados por el objeto rule son: add, delete y show.

ip tunnel: permite la configuración de túneles ip, básicamente encapsulando paquetes dentro de paquetes IPv4 y enviándolos sobre la infraestructura IP. Los comandos para este objeto son: add, delete y show

ip monitor: con este objeto es posible monitorear el estado de los dispositivos, direcciones y rutas continuamente, este tiene el siguiente formato de sintaxis:

ip monitor [file FILE] [all OBJECT-LIST]

Aquí OBJECT-LIST es la lista de los objetos tipo los cuales se desean monitorear.

Es el momento de hacer referencia a tc (Traffic Control), esta utilidad de iproute2 interactúa con el kernel para la creación, borrado o modificación directa de estructuras de control de tráfico, su sintaxis es como sigue:

tc [OPTIONS] OBJECT { COMMAND | help }

Donde OBJECT se refiere a qdisc, class o filter, OPTIONS define { -s[tatistics] | -d[etails] | -r[aw] }

Para el manejo de la utilidad tc se requiere conocer los parámetros particulares de cada qdisc, class o filter, que pueden diferir de una a otro en cada objeto.

Por qdisc se entiende el conjunto de disciplinas de colas que Linux puede manejar, class define las clases para las disciplinas que soportan clases y filter brinda la posibilidad de clasificar paquetes.

Con tc es posible definir estructuras para gestionar el ancho de banda de un enlace y cuenta con unas reglas para especificación del ancho de banda:

mbps = 1024 kbps = 1024 x 1024 bps (bytes/s)

kbps = 1024 bps = 1024 (bytes/s)

kbit = 1024 (bits/s)

mbit = 1024 kbits (kilobits/s)

mb = 1024 kb = 1024 x 1024 b (byte)

Para la definición de tiempo se tiene ms, msec o msecs se refiere a milisegundos y us, usec o usecs define microsegundos.

Se debe tener en cuenta que cuando se maneje temas relacionados con memoria como tamaño de cola, buffer se habla de unidades en bytes y para casos de ancho de banda y velocidad las unidades son en bits.

Disciplina de colas (qdisc): se entiende como aquello que permite encolar o desencolar los paquetes en un tiempo determinado con unos algoritmos específicos. La sintaxis para manejar las disciplinas de colas es:

tc qdisc [add | del | replace | change | get |] dev STRING [handle QHANDLE]

[root | ingress | parent CLASSID] [estimator INTERVAL TIME_ CONSTANT]

[[QDISC_KIND] [help [OPTIONS]]]

Como se puede observar la definición de una disciplina de cola es una línea de comandos con una lista de parámetros.

Donde:

- add: agrega una cola al dispositivo.
- del: borra la qdisc del dispositivo.
- replace: reemplaza la qdisc con otra.
- dev: define el dispositivo de red al cual se vincula la qdisc
- handle: define el manejador asignado a la disciplina de cola, consta de un número que es asignado por usuario, el menor valor es cero, su forma es major:minor. Dos disciplinas de colas no pueden tener el mismo manejador.
- change: permite cambiar algún parámetro en la disciplina de cola.
- root: indica a tc que la disciplina de cola es tipo egress(salida) también existe la ingress que puede ser asignada al mismo dispositivo de red. Indica que la disciplina de cola está en la raíz del enlace compartido jerárquicamente y es propietaria de todo el ancho de banda, solo puede haber una qdisc root por dispositivo.
- ingress: indica a tc que la disciplina de cola es de ingreso y por ahora solo permite controlar la entrada de paquetes.
- parent: representa el manejador de la disciplina de cola padre.
- estimator: utilizado para determinar si los requerimientos de la cola han sido satisfechos. Con este estimador se estima el ancho de banda usado por cada clase sobre el intervalo, permitiendo establecer si cada clase a recibido el enlace compartiendo el ancho de banda. Donde INTERVAL define el intervalo entre mediciones y TIME_CONSTANT es una constante de tiempo promedio.

Clases (class): Este concepto es aplicable a aquellas disciplinas de clases que soportan tener clases, estas no son mas que una característica de estas para posibilitar dividir el tráfico por tipos, dentro de la misma disciplina de cola. Veamos su sintaxis:

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

Donde:

- classid: representa el manejador que está asignado a la clase por el usuario, este consta de un número mayor y un número menor separados por dos puntos.
- root: indica que la clase representa la clase raíz en la jerarquía en el enlace compartido.
- handle: se refiere al manejador del padre de la disciplina de cola.

Filtros (filter): Usado para clasificar los paquetes basados en ciertas propiedades del paquete. Las disciplina de colas utilizan los filtros para asignar los paquetes que ingresan a una clase en particular, estos pueden ser mantenidos por clases o por disciplinas de colas depende del diseño de la disciplina de cola. Existe la posibilidad de definir listas de filtros que son manejados por orden de prioridad en orden ascendente. Su sintaxis es como sigue:

tc filter [add | del | change | change | get] dev STRING [pref PRIO]

[protocol PROTO] [estimator INTERVAL TIME_CONSTANT] [root | classid CLASSID]

[handle FILTERID] [[FILTER_TYPE] [help | OPTIONS]

Donde:

- pref: define la prioridad asignada al filtro.
- protocol: usado para que el filtro identifique los paquetes según el protocolo al que pertenecen, dos filtros no pueden tener la misma prioridad y campo de protocolo.
- root: indica que el filtro es la raiz de la jerarquia del enlace compartido.
- estimator: cumple idéntica función a como se explico para las clases.
- handle: representa el manejador por el cual el filtro es identificado de manera única, su formato varía según el clasificador, el tipo de filtro puede ser u32, fw, route, rsvp, tc_index, etc.
- classid: define el manejador de la clase a la cual el filtro es aplicado.

Algunos ejemplos simples de uso de ip y tc son, para el caso de túneles Linux cuenta con tres tipos: IP sobre IP, túneles GRE y túneles que se realizan fuera del kernel, básicamente se trata de encapsular un paquete IP sobre otro.

Empleando la utilidad ip un túnel tendría la siguiente forma:

ip tunnel add tunelA mode sit remote 192.31.7.104 local 192.203.80.142

Lo que se hizo fue crear un túnel llamado tunelA y tipo sit que es un túnel IPv6 sobre IPv4, el cual debe ir a (remote) 192.31.7.104 y sale desde (local) 192.203.80.142

Revisando el tema de múltiples tablas de enrutamiento, lo que se plantea es la posibilidad de definir varias tablas de enrutamiento, lo que lleva a decidir que tabla utilizar

para un determinado tráfico IP, clasificando el tráfico y asignándole diferentes reglas para su enrutamiento, como base existen tres tablas por defecto main, local y default, cada una con prioridades diferentes y que se aplican a todo el tráfico.

Un ejemplo de generación de una tabla es:

```
# echo 200 tablaprueba >> /etc/iproute2/rt_tables
```

```
# ip rule add from 192.168.30.0/25 table tablaprueba
```

Lo que se hizo fue definir una tabla llamada tablaprueba y se asigno una regla a esta tabla, que consistió en asignar un rango de IP que se enrutan según la tabla.

En lo referente al control de tráfico con iproute2 se requiere utilizar la utilidad tc (Traffic Control) para realizar gestión del ancho de banda, veamos una línea de un script donde se hace manejo de ancho de banda:

```
# tc qdisc add dev ppp0 root tbf rate 220kbit latency 50ms burst 1540
```

Revisando esta línea se observa que se definió una disciplina de cola tbf, asignada al dispositivo ppp0 con una velocidad de 220 kbit/s rata de salida de los paquetes, tiempo máximo que puede pasar un paquete en el tbf aquí de 50ms y tamaño máximo del buffer hasta de 1540 bits, la línea de comando anteriormente definida para tc tiene relación directa con la disciplina de cola trabajada en este caso TBF (Token Bucket Filter) la cual tiene sus parámetros propios de configuración.

Lo que hasta aquí se ha tratado tiene que ver con las posibilidades de Linux para realizar enrutamiento avanzado, se presento de manera breve las herramientas iptables, ip y tc con ejemplos muy sencillos de lo que se puede hacer a nivel filtrado de paquetes, enrutamiento con múltiples tablas, control de tráfico regulando el ancho de banda, en lo que sigue se dará explicación de como Linux aborda el tema de control de tráfico y como se logra esto usando iproute2 con sus utilidades.

De la figura 21 se puede apreciar de manera gráfica que le ocurre a un paquete dentro del kernel, se establecen diferencias según sea procesado por la utilidad iptables (ipchains) o iproute2, para el caso de control de tráfico se deben observar los bloques delineados en rojo, lo cuales son controlados por las utilidades ip y tc, donde la referencia a QOS se asocia a control de tráfico, se ve existencia de un tratamiento a la entrada y a la salida y está presente la RPDB o PDBB (Routing Policy DataBase).

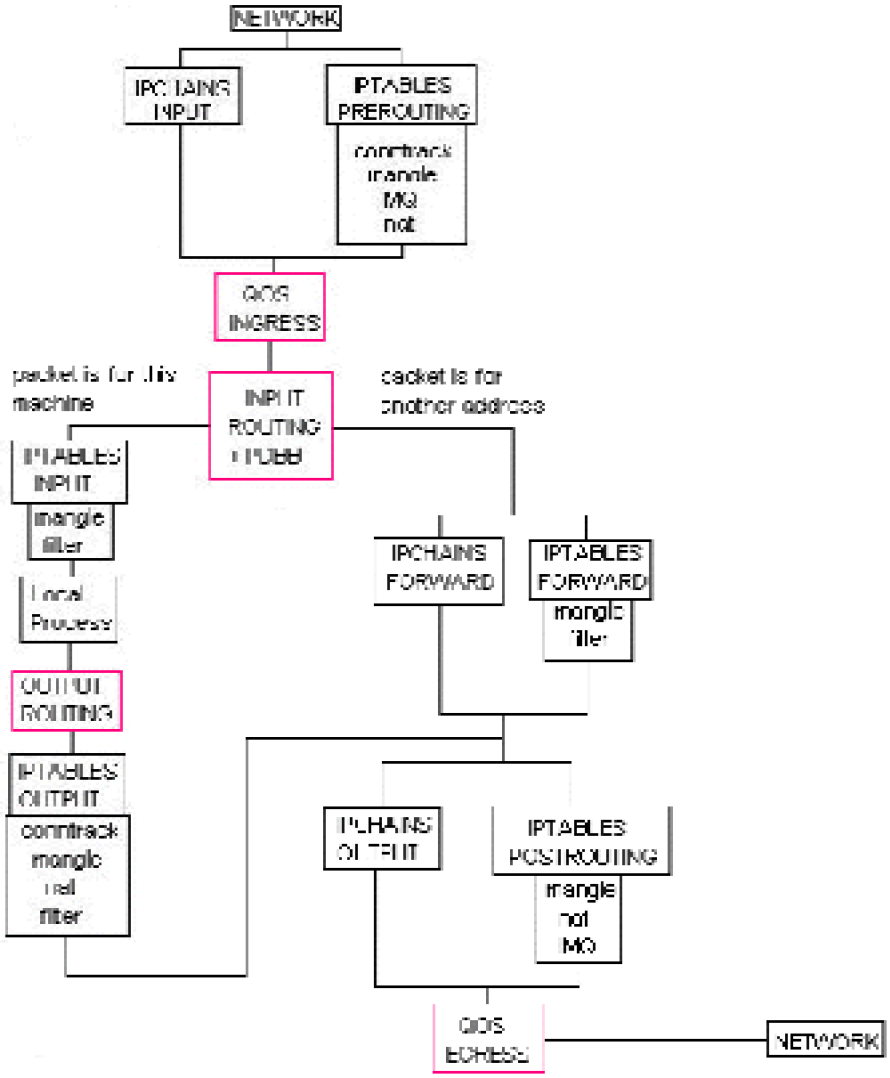


Figura 21. Diagrama sobre como viaja un paquete en el kernel [10]

4. Linux y el control de TRÁFICO

El sistema operativo Linux a partir de la versión de kernel 2.2.x cuenta con un conjunto de herramientas o utilidades que permiten realizar tareas de enrutamiento avanzado, esto consiste en poder efectuar la gestión y manipulación del los paquetes que se transmiten.

El control de tráfico hace referencia a un conjunto de sistemas de colas y los mecanismos bajos los cuales los paquetes son recibidos y transmitidos por un router, buscando decidir cuales paquetes y a que rata se les permite la entrada a través de una interface y también se debe determinar que paquetes serán transmitidos y en que orden por la interface de salida.

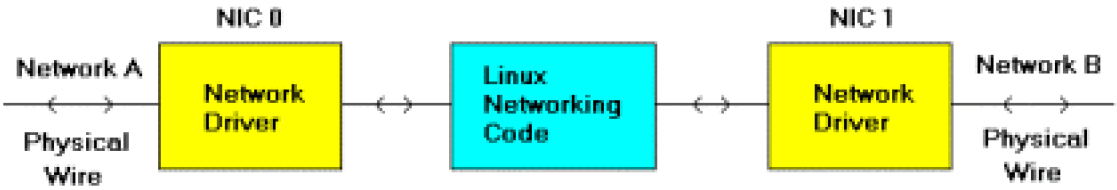


Figura 22 Operación de Linux como un Router [20]

Se tiene entonces que control de tráfico es un conjunto de herramientas que permiten tener un control fino sobre las colas y los mecanismos de encolamiento de un dispositivo de red.

Teniendo un maquina con sistema operativo Linux operando como un router puro (figura 22), es decir, realizando el reenvío de paquetes de una interface a otra, sin

generar o consumir paquete alguno, entonces lo que ocurre es que unos paquetes pueden estar entrando desde la red A por la interface de red NIC 0 allí el controlador (Network driver) se encarga de entregar los paquetes al código de red de Linux (Linux Networking Code) este realiza el reenvío de estos paquetes, solicitando a la interface de red NIC 1 enviar estos paquetes a la red B.

Por medio de la figura 23 se aprecia como una maquina Linux puede regular de alguna manera el tráfico, esta nos muestra el proceso que realiza el kernel para los paquetes que ingresan y los paquetes que genera para ser enviados a la red, se tiene un de-multiplexor a la entrada que determina si los paquetes van destinados al nodo local , si es así estos son enviados a las capas superiores para su procesamiento, de lo contrario estos van al bloque de reenvío el cual recoge los paquetes generados localmente y los externos para reenviarlos, se presenta entonces la posibilidad de que se pueda realizar un control de tráfico de estos paquetes, y es aquí donde el control de tráfico de Linux juega un rol importante, pues gracias a esto se puede realizar un conjunto de disciplinas de colas, clases y filtros mediante los cuales es posible efectuar un control sobre los paquetes que van a salir.

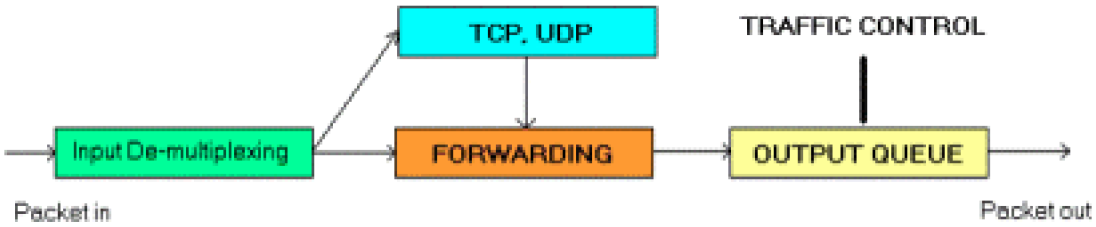


Figura 23. Como regular el tráfico [20]

Como se dijo antes lo interesante está en que la máquina Linux opere como un router, realizando tareas de reenvío o rechazo de paquetes, veamos con más detalle como es posible llevar a cabo el control de tráfico gestionando las colas de paquetes, empleando la figura 24 que sigue:

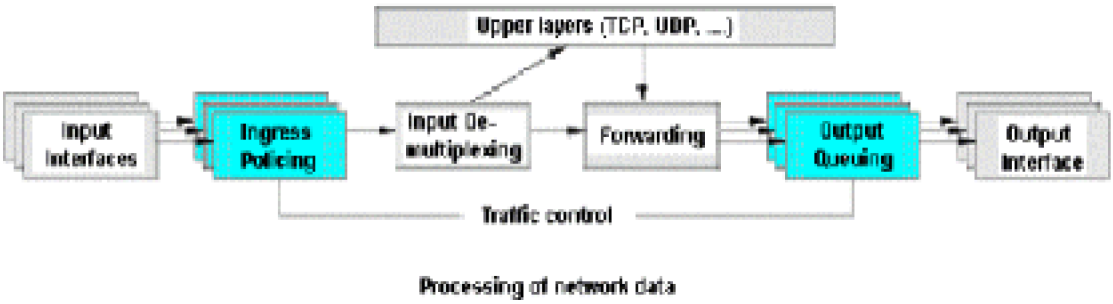


Figura 24. Control de tráfico con Linux [20]

Se puede a preciar que los paquetes que ingresan por las interfaces de entrada están sometidos a lo que el kernel puede decidir mediante una regulación de ingreso (Ingress Policing), es decir, permitir su entrada o descártarlos debido a que están llegando demasiado rápido, después de ello interesa los que son reenviados a la cola de salida (Output Queuing) donde nuevamente el kernel decide a que cola deben ir los

paquetes, por cual interface van a salir, si los paquetes serán encolados o no, como serán enviados los paquetes según un orden, y si es posible retrasar su envío. Después de que el kernel realiza su control los paquetes son enviados a las interfaces de salida. En resumen lo que se tiene son dos etapas donde se realiza el control del tráfico por parte del kernel de Linux: una en la entrada Ingress Policing y la otra a la salida con las colas Output Queueing.

Entonces a nivel de su estructura en lenguaje C, el kernel de Linux para realizar el control de tráfico se basa en los siguientes conceptos:

- Queueing disciplines
- Classes
- Filters y Policers

Mediante el empleo de las características del kernel de Linux para realizar control de tráfico, es posible llevar a cabo tareas tales como:

- Limitar el ancho de banda a un valor conocido
- Limitar el ancho de banda de un usuario o servicio
- Reservar ancho de banda para una aplicación en particular
- Priorizar el tráfico
- Permitir equilibrar el uso del ancho de banda entre diferentes usuarios
- Realizar balanceo de carga entre varios servidores
- Rechazar cierto tráfico en particular

A continuación se desarrollan conceptos tales como queuing discipline, classes y filters/policing.

4.1 Disciplinas de Colas

Las disciplinas de colas (queuing disciplines) son algoritmos que se encargan de gestionar las colas en todo el proceso de ingreso (ingress) y salida (egress) de los paquetes sobre un dispositivo de red, estas son la base sobre la que se fundamenta la gestión de tráfico en Linux, cada dispositivo de red tiene asociado un proceso de ingreso y de egreso.

Existen dos grupos de disciplinas de colas, unas con clases y las otras sin clases. Aquellas con clases permiten al usuario crear subdivisiones para realizar diferenciación en el tratamiento del tráfico, mientras que en las disciplinas de colas sin clases esto no es posible. Se debe tener en cuenta que Linux define el concepto root para referirse a la etapa egress, es decir, tráfico saliente sobre el cual es posible aplicar toda la potencia del control de tráfico y está el concepto ingress que se refiere al tráfico que ingresa, y su única utilidad es limitar el tráfico entrante. Linux soporta varias disciplinas de colas y son

las siguientes:

- CBQ: Class Based Queue
- TBF: Token Bucket Filter
- CSZ: Clark-Shenker-Zhang
- FIFO: First In First Out
- TEQL: Priority Traffic Equalizer
- SFQ: Stochastic Fair Queuing
- ATM: Asínchronous Transfer Mode
- RED: Random Early Detection
- GRED: Generalized Random Early Detection
- DSMARK: Diff-Serv Mark
- HTB: Hierarchical Token Bucket

Las funciones que soportan las diferentes disciplinas de colas son:

- Enqueue: Permite encolar los paquetes a la disciplina de cola.
- Dequeue: Desencola los paquetes para que sean enviados.
- Requeue: Reencola un paquete para que sea transmitido, esta se utiliza en el caso de que al ser desencolado el paquete este no es transmitido por alguna razón, este entonces se vuelve a encolar a la misma posición donde fue desencolado.
- Drop: Descarta los paquetes de la cola.
- Init Inicializa y configura los parámetros de una disciplina de cola cuando es creada.
- Reset: Reinicializa la disciplina de cola a su estado inicial.
- Destroy: Usada para remover una disciplina de cola con sus clases y filtros asociados.
- Dump: Empleada para realizar diagnóstico de los datos de una disciplina de cola. Cada disciplina mantiene datos de diagnóstico que son volcados cuando esta función es invocada.

4.2 Clases

Las disciplinas de colas que permiten definir clases (Classes) en su interior brindan al usuario la posibilidad de establecer configuraciones para diferentes tipos de tráfico a los cuales se les da tratamiento según la clase. Cada clase posee una cola que por defecto es una tipo FIFO, cuando la función de encolamiento de la disciplina de cola es llamada, la disciplina de cola aplica los filtros para determinar a que clase pertenece el paquete.

Existen dos maneras mediante las cuales se puede identificar una clase, una es por

medio de la identificación asignada por el usuario y la otra es interna y es realizada por kernel, esta última se conoce como identificador interno (internal ID) y es único, el otro se conoce como identificador de clase (class ID). El class ID es un dato tipo u32, mientras que el interno es un entero largo sin signo, la mayoría de las funciones sobre las clases usan el internal ID para identificar la clase, aunque existen funciones como get change que utilizan el class ID también. La estructura del class ID es major mumber : minor number, donde el major number corresponde a la instancia de la disciplina de clase y el minor number identifica la clase dentro de esa instancia.

Las funciones que se pueden aplicar a las clases son:

- Graft: Utilizada para ligar una nueva disciplina de cola a la clase, como se dijo antes la disciplina de cola por defecto atada a una clase es la FIFO, pero esta se puede cambiar invocando esta función.
- Get: Empleada para obtener el internal ID dando su class ID. Get incrementa el contador de uso de la clase.
- Put: Invocada cuando una clase que ha sido previamente referenciada mediante la función get se le retira la referencia, esto hace que put decremente el contador de uso de la clase.
- Change: Esta función cambia las propiedades asociadas con una clase, inclusive se usa a veces para crear clases.
- Delete: Función usada para borrar una clase, esta determina el uso de la clase al verificar el contador de referencia, si está en cero, procede a remover y desactivar la clase.
- Walk: Usada para iterar sobre todas las clases de una disciplina de cola e invoca una función de retorno para cada una de las clases, se usa habitualmente para diagnóstico de datos para todas las clases de una disciplina de cola.
- Tcf_chain: Retorna el puntero de una lista de filtros que están asociados a una clase, cada clase es asociada a una lista-filtro que contiene una lista de filtros que usados para identificar los paquetes que pertenecen a una clase en particular.
- Bind_tcf: Esta función es usada para unir una instancia de un filtro a una clase.
- Unbind_tcf: Utilizada para remover una instancia de una filtro unida a una clase.
- Dump_class: Empleada para volcar los datos de diagnóstico de una clase.

4.3 Filtros

Para que Linux pueda realizar tareas de control de tráfico, cuenta en su kernel con utilidades que permiten tratar de manera diferente los paquetes, en este caso se explicará lo relacionado con los filtros (Filters) y los reguladores (Policers).

Esta explicación va dirigida a entender dichos conceptos relacionados de manera

general con el control de tráfico.

Los filtros están asociados a las disciplinas de colas pues es a través de estos que se define a que clase serán asignados los paquetes que ingresan, un filtro básicamente se usa para clasificar los paquetes basados en propiedades del campo TOS, la dirección IP, el número de puerto, etc.

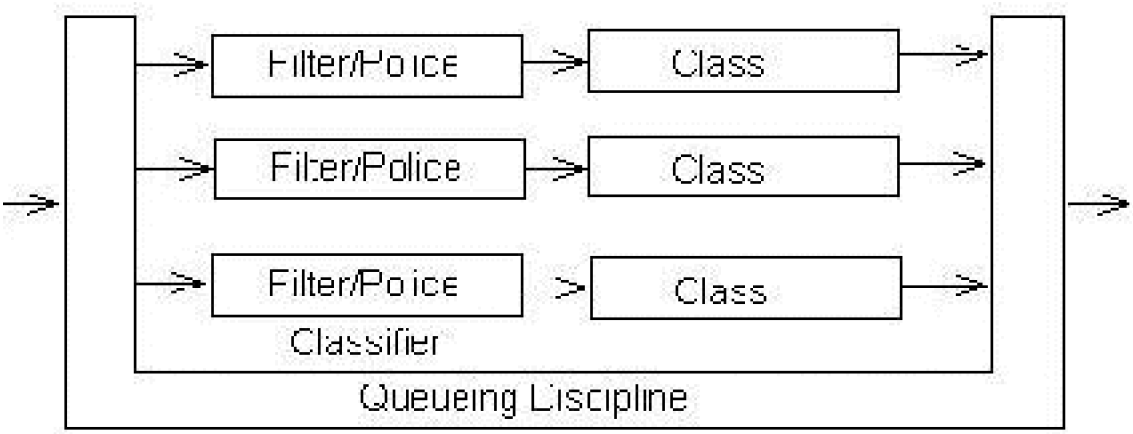


Figura 25. Modelo para control de tráfico del kernel de Linux [17]

De la figura 25 se puede apreciar como el kernel de Linux maneja el tráfico donde la disciplina de cola tiene asociada un filtro/regulador y unido a esto puede venir una clase.

Con los filtros se forman listas de estos que son mantenidos por la disciplina de cola o la clase, según sea el diseño de la disciplina de cola, estas listas son ordenadas por prioridades de forma ascendente, adicional a esto un filtro puede tener una estructura interna de elementos que son referenciados por un manejador (handle). Estos manejadores son identificados por un número de 32 bits, un filtro con manejador 0, se refiere al filtro en si.

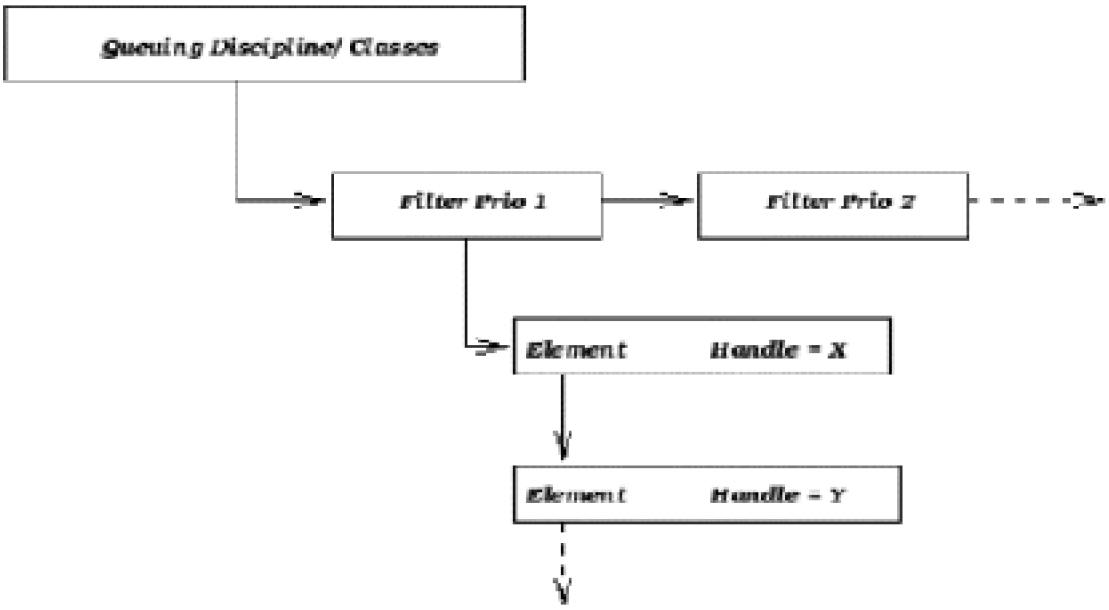


Figura 26. Estructura de los Filtros [17]

De la figura 26 se establece la estructura de filtros y como un filtro contiene elementos asociados con su respectivo manejador. El proceso se realiza como sigue, dentro de cada filtro los elementos son revisados con el objeto de clasificar el paquete, una vez clasificado el paquete este es encolado de acuerdo a la disciplina de cola y su clase asociada si hay alguna.

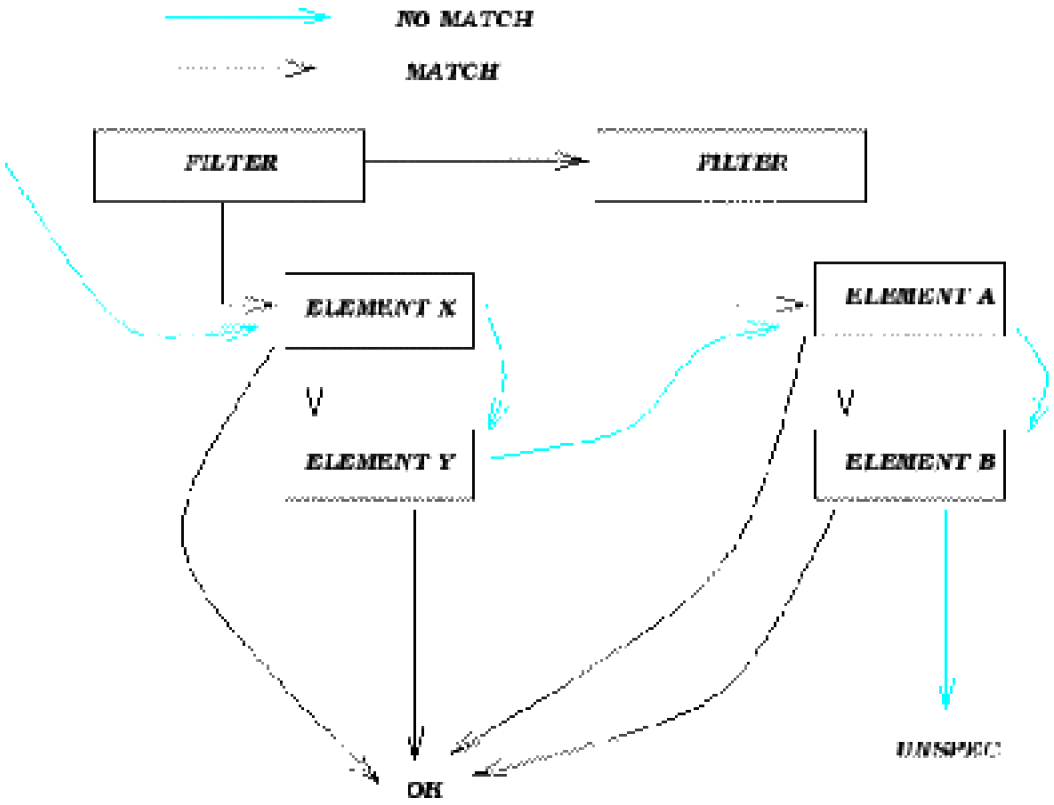


Figura 27. Proceso de selección en un Filtro [17]

Se tiene entonces que para identificar una disciplina de colas y una clase dentro de la estructura de control de tráfico se requiere un único identificador el cual es conocido como manejador (handle) el cual está constituido por dos partes: un número mayor (major number) y un número menor (minor number), estos números son asignados por el usuario de manera arbitraria de acuerdo a la siguientes reglas:

- major: este parámetro es completamente libre de significado para el kernel. Se puede usar un esquema arbitrario de numeración, sin embargo todos los objetos en la estructura de control de tráfico con el mismo parentesco deben compartir un major number.
- minor: este parámetro identifica sin ambigüedad el objeto como una qdisc si tiene un valor de cero, cualquier otro valor identifica el objeto como una clase, todas las clases que comparten un pariente deben tener minor numbers únicos.

Un manejador (handle) especial es ffff:0 que está reservado para el ingress, el manejador

es usado como una etiqueta en classid y flowid de la utilidad tc cuando con esta se define un filtro, básicamente los manejadores (handles) son identificadores externos usados por el usuario, el kernel cuenta con identificadores internos para cada objeto.

Se presentan a continuación las funciones bajo las cuales un filtro puede ser manipulado esto a nivel de su estructura en lenguaje C, se pretende básicamente mostrar lo que hay de detrás de un filtro cuando se utiliza en el proceso de definir la estrategia de control del tráfico, con las herramientas como tc dentro de iproute2.

Classify: función que permite seleccionar un paquete para una clase dada, según las propiedades del mismo.

Init: función empleada para inicializar los parámetros del filtro.

Destroy: función usada para remover el filtro.

Get: gracias a que los filtros tienen una identificación interna (ID) asociada al manejador, mediante esta función es posible obtener el ID interno del filtro.

Put: es utilizado cuando un elemento de un filtro previamente referenciado por la función get no es empleado más.

Change: permite configurar un nuevo filtro o efectuar cambios sobre uno ya existente.

Delete: borra un elemento de un filtro.

Walk: itera sobre todos los elementos de un filtro e invoca una función de retorno para cada uno de ellos, usado para diagnóstico de datos.

Dump: retorna un diagnóstico de datos para un filtro o uno de sus elementos.

Básicamente en Linux se definen dos tipos de filtros: los genéricos y los específicos y su diferencia radica en el alcance que se le da a los paquetes según la instancia de clasificación.

Para los filtros genéricos se requiere una instancia del filtro por disciplina de cola para clasificar los paquetes para todas las clases.

En el caso de los filtros específicos estos necesitan una o más instancias de un filtro o de sus elementos internos por clase para identificar la pertenencia de los paquetes a una clase. Las figuras 28 y 29 a continuación muestran gráficamente la diferencia entre un filtro genérico y un filtro específico.

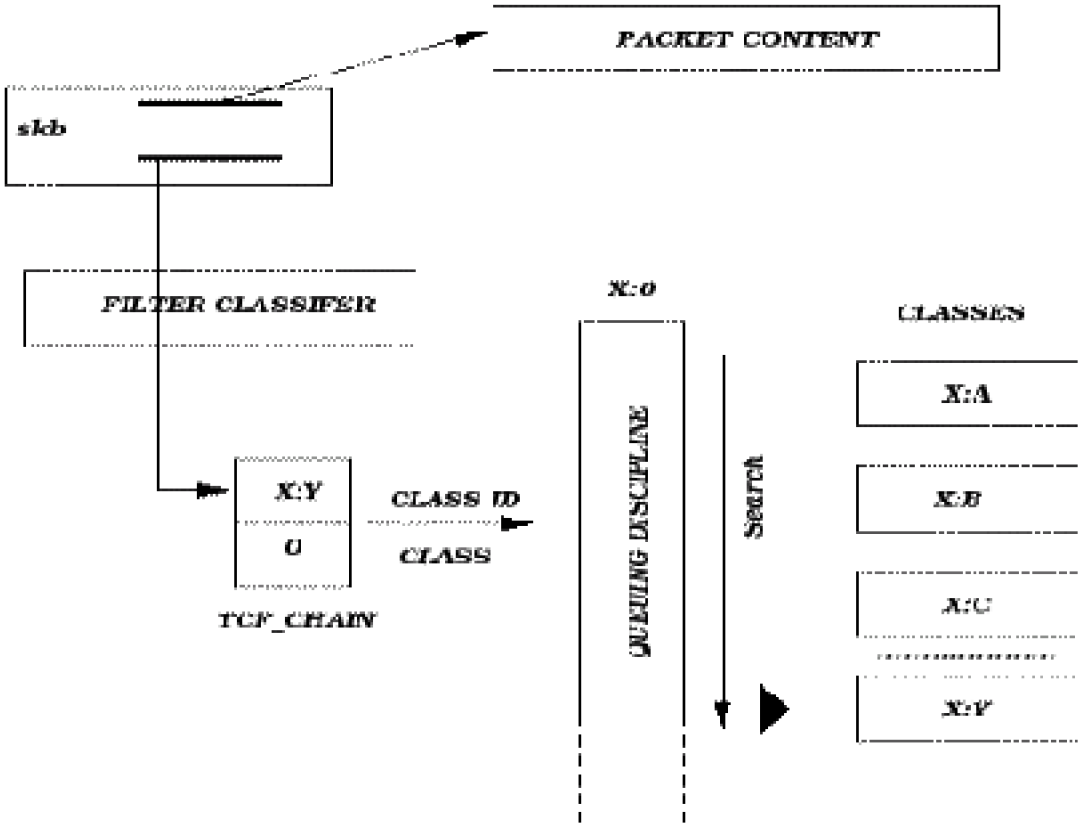


Figura 28. Filtro Genérico [17]

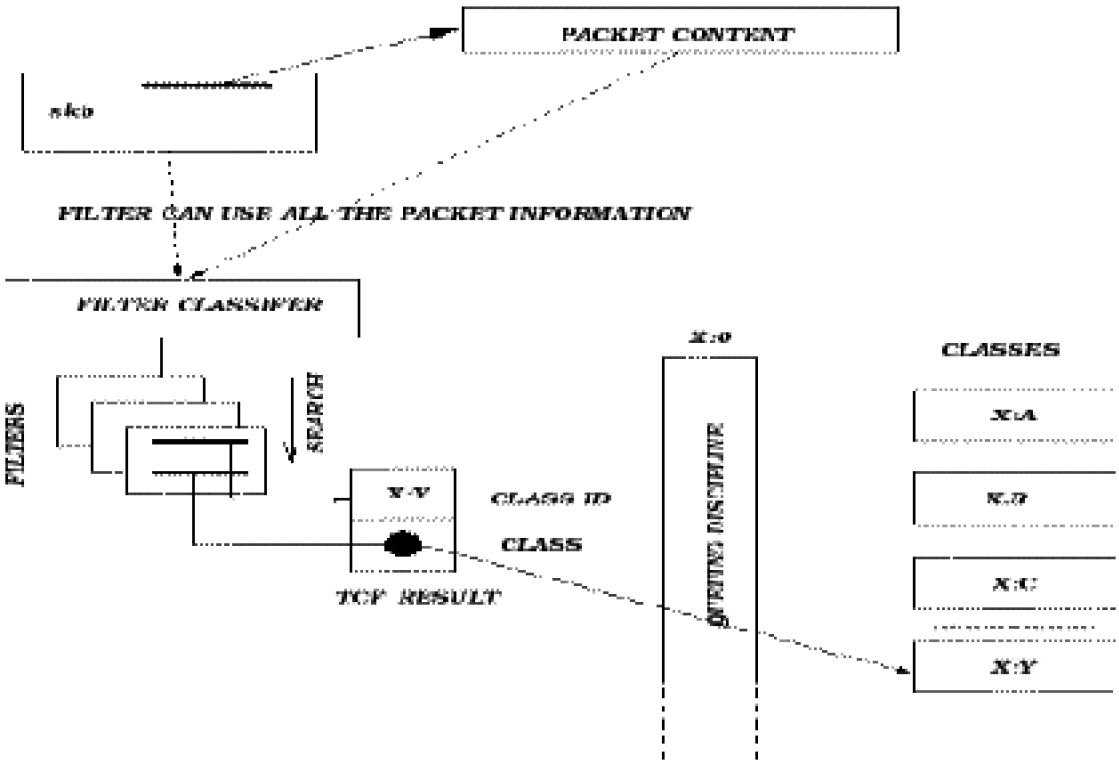


Figura 29. Filtro Específico [17]

4.4 Regulación o Control

Cuando se habla de regulación o control (Policing), para efectuar control de tráfico, se refiere a regular el tráfico de los paquetes para que estos no excedan un límite.

En principio se consideran cuatro tipos de mecanismos de regulación o control:

- Decisiones de regulación por filtros
- No aceptar encolar un paquete
- Rechazo de un paquete desde una cola interior
- Rechazo de un paquete cuando se encola uno nuevo.

Cuando se habla de decisiones de regulación por filtros, son estos últimos los que deciden que hacer con el paquete, ya sea no realizar nada, reclasificarlo si este excede ciertos limites o simplemente descartarlo por que el paquete violó los límites.

El caso de no aceptar encolar un paquete, básicamente se presenta cuando la disciplina de cola falla en encolar el paquete y este es descartado, algunas disciplinas de colas cuentan con mecanismos de realimentación que permiten volver a llamar la disciplina de cola, dando así una segunda oportunidad al paquete para que sea encolado.

El mecanismo de rechazo de un paquete desde una cola interior, es aplicado si la disciplina de cola opta por rechazar un paquete desde una disciplina de cola interior después de que el paquete ha sido encolado, para crear espacio a paquetes de clases más importantes.

Finalmente en el mecanismo de rechazo de un paquete cuando se encola uno nuevo, este descarta los paquetes que han sido exitosamente encolados, si la función de encolamiento de la disciplina de cola establece que hay un nuevo paquete más importante que alguno más antiguo, descartando el antiguo y encolando el nuevo.

En general la regulación busca medir y limitar el tráfico en una cola en particular, esta como elemento del control de tráfico se convierte en un mecanismo para limitarlo, la regulación es usada al extremo de la red para asegurar que no se exceda un ancho de banda asignado, el tema de regulación (Policing) de tráfico en Linux está directamente relacionado con los filtros, pues son parte de estos.

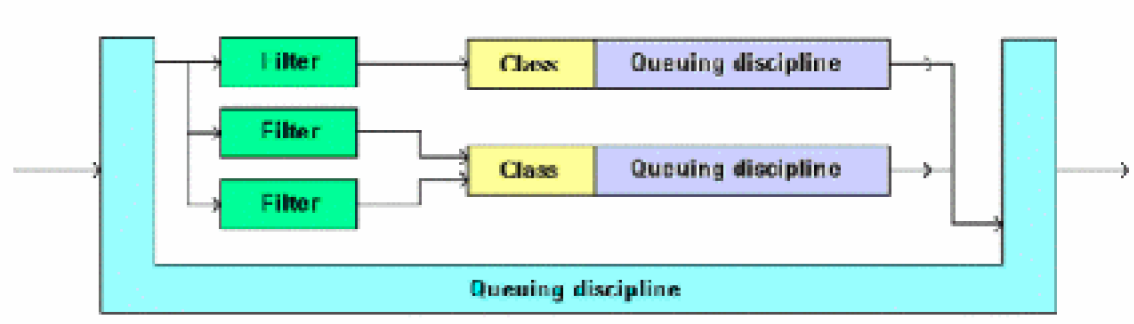


Figura 30. Manejo de disciplinas de colas con clases y filtros. [20]

La figura 30 resume los temas tratados hasta el momento, disciplina de colas, clases y filtros, Linux dispone entonces de un conjunto de componentes para realizar el control de tráfico, que tienen su análogo en los elementos tradicionales de control de tráfico, a continuación se listan:

Tabla 2. Comparación entre control de tráfico tradicional y Linux [18]

Elementos Tradicionales	Componentes de Linux
shaping (ajuste)	class, ofrece capacidades de ajuste.
Scheduling (ordenación o gestión)	qdisc es en si un scheduler (ordenador de los paquetes para salgan de cierta forma)
Classifying (clasificación)	filter, realiza la clasificación, pues esta es parte del filtro.
policing (regulación o limitación)	policer hace parte de filter, como mecanismo de regulación.
dropping (descarte)	filter con un policer, permite descartar el trafico.
Marking (Marcado)	dsmark, mediante esta disciplina de colas se realiza el marcaje del paquete, es decir, puede ser alterado.

Hasta aquí se ha revisado el tema de control de tráfico con Linux presentándose la estructura general a nivel de kernel, se mostró como se realiza el proceso, que objetos utiliza Linux para llevar acabo dicha tarea que funciones están asociadas, interesa ahora es cual es la interface de usuario disponible para llevar a cabo el control de tráfico, en este trabajo se revisó la herramienta IPROUTE2 con su utilidad tc (Traffic Control) que gracias a Alexey Kuznetsov y la existencia de NETLINK, pues por medio de este IPROUTE2 y sus utilidades se comunican con el kernel, lo que sigue es aplicar los conceptos ya vistos usando las utilidades de IPROUTE2, para ello se revisarán algunas disciplinas de colas mostrando como es su configuración con tc, lo primero que se debe saber es que la configuración del kernel debe brindar soporte al control del tráfico, se presenta a continuación lo requerido para esto:

```
#
CONFIG_NET_SCHED=y
CONFIG_NET_SCH_CBQ=m
```

```
CONFIG_NET_SCH_HTB=m
CONFIG_NET_SCH_CSZ=m
CONFIG_NET_SCH_PRIO=m
CONFIG_NET_SCH_RED=m
CONFIG_NET_SCH_SFQ=m
CONFIG_NET_SCH_TEQL=m
CONFIG_NET_SCH_TBF=m
CONFIG_NET_SCH_GRED=m
CONFIG_NET_SCH_DSMARK=m
CONFIG_NET_SCH_INGRESS=m
CONFIG_NET_QOS=y
CONFIG_NET_ESTIMATOR=y
CONFIG_NET_CLS=y
CONFIG_NET_CLS_TCINDEX=m
CONFIG_NET_CLS_ROUTE4=m
CONFIG_NET_CLS_ROUTE=y
CONFIG_NET_CLS_FW=m
CONFIG_NET_CLS_U32=m
CONFIG_NET_CLS_RSVP=m
CONFIG_NET_CLS_RSVP6=m
CONFIG_NET_CLS_POLICE=y
#
```

Inicialmente se va a revisar el uso de tc con algunas disciplinas de colas, primero sin clases y por ultimo con clases.

4.5 Configuración de disciplinas de cola sin clases y con clases en Linux

4.5.1 Disciplinas de colas sin clases.

Básicamente solo pueden reordenar, retrasar o descartar el tráfico. Existe una disciplina de colas sin clases por defecto para Linux, la cual es utilizada por este si no se ha definido ninguna otra, la cual se llama pfifo_fast, esta básicamente es una FIFO, esta

disciplina tiene tres bandas con prioridades 0, 1, 2 dentro de cada banda se aplica la regla FIFO, los paquetes que vengan con mayor prioridad van a la banda 0, los siguientes a la banda 1 y el resto a la banda 2, ver figura 31

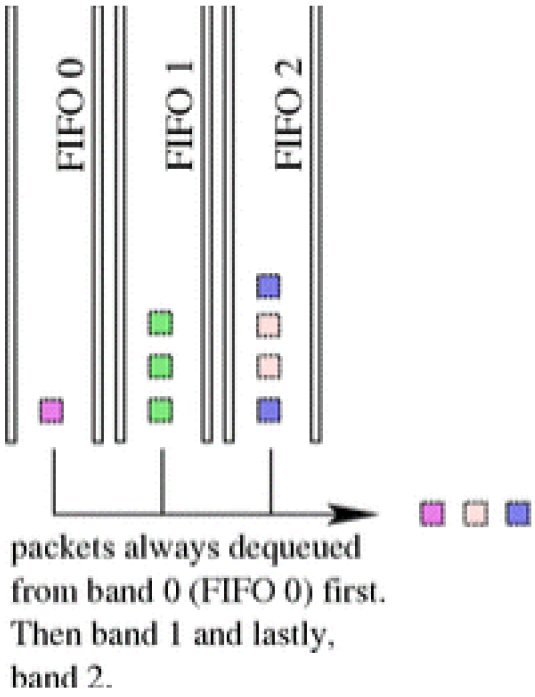


Figura 31. Disciplina de cola pfifo_fast [19]

Esta disciplina de cola no puede ser manipulada por el usuario, pues tiene ya una configuración por defecto, para priorizar el tráfico el kernel se basa en la campo TOS del paquete IP.

4.5.1.1 FIFO.

Linux define dos tipos de disciplinas de colas una PFIFO la otra BFIFO, estas son básicamente First In First Out (FIFO), la primera maneja paquetes la segunda bytes. Configurar una FIFO con tc es muy simple:

tc qdisc add dev eth0 root pfifo limit 10

Lo que se hizo fue agregar “add” una nueva disciplina de cola “qdisc” al dispositivo de red “dev” eth0 (interfaz Ethernet 0), root se refiere a disciplina de cola raíz (egress) se requiere a pesar que pfifo es una disciplina de cola sin clases por el formato de la utilidad tc, pfifo (disciplina de cola PFIFO) y limit establece la longitud de la cola, que es el numero de paquetes a tener en la cola, en este caso 10 paquetes en cola, este parámetro es el único a configurar en esta disciplina de cola.

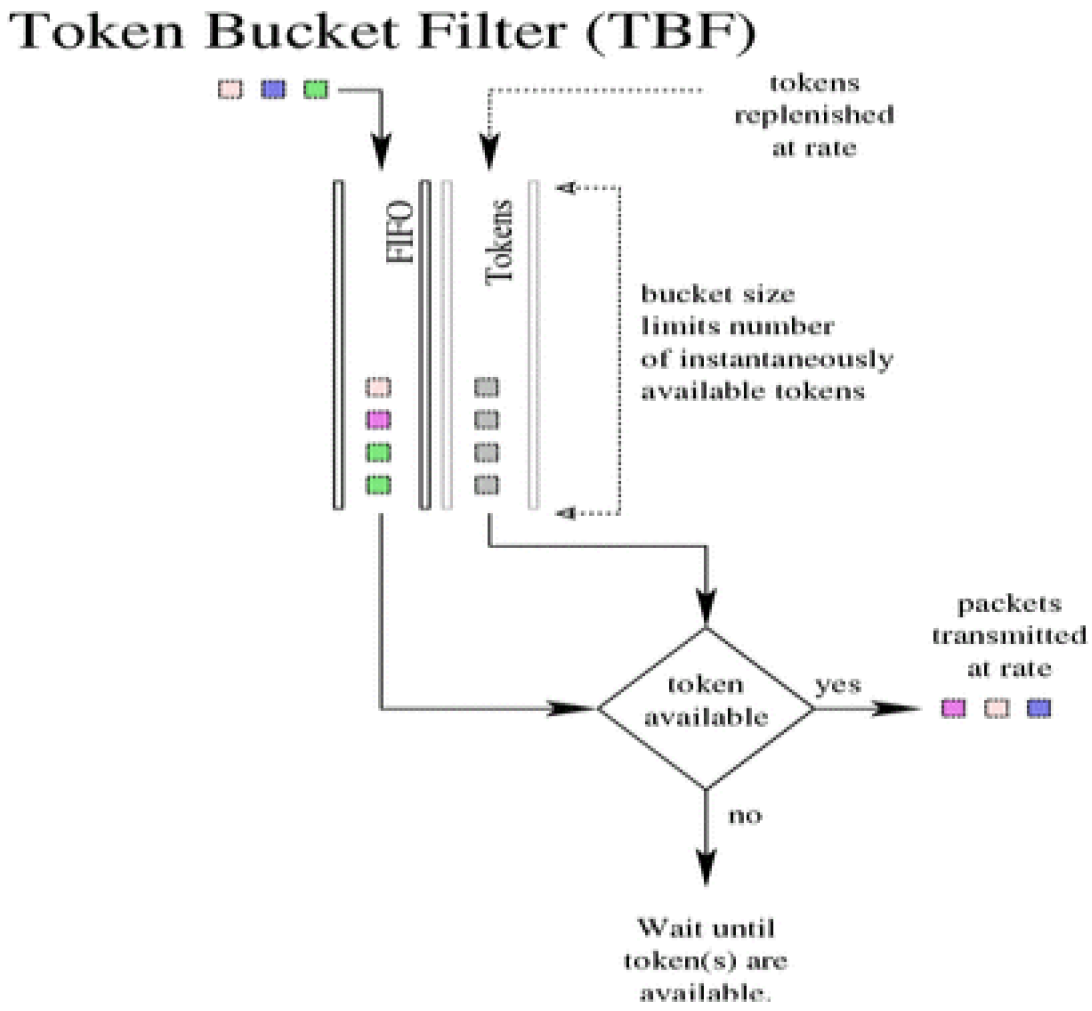


Figura 32. Disciplina de Cola Token Bucket Filter [19]

4.5.1.2 Token Bucket Filter.

Esta disciplina de cola basa su funcionamiento en el concepto bucket que no es más que un buffer al cual llegan los tokens, los paquetes utilizan los tokens para abandonar la interfaz, es decir, deben existir tokens disponibles para que un paquete abandone la cola, esto significa que dependiendo del ritmo de disponibilidad de tokens, será la velocidad con la cual los paquetes salgan de la cola, gráficamente este concepto se aprecia en la figura 32:

Esta disciplina de cola permite que los paquetes salgan a un ritmo previamente configurado e incluso soporta ráfagas de exceso.

El bucket (Buffer) se va llenando con los tokens a un ritmo determinado, aquí lo que importa es el tamaño del bucket, pues determina la cantidad de tokens que puede almacenar, esta disciplina de cola tiene asociado dos flujos uno de tokens y otro de paquetes, lo cual brinda tres posibilidades:

- Los datos llegan a un ritmo que es igual al de llegada de los tokens, permitiendo que cada paquete tenga un token y así este pasa la cola sin retraso.
- Los datos llegan a un ritmo menor que los tokens, significa esto que aun quedan tokens disponibles en el buffer, esto permite el envío de paquetes a una rata superior a la del ritmo de llegada de los tokens, es decir, breves ráfagas de exceso.
- Los datos llegan a una velocidad superior que el ritmo en que ingresan los tokens, esto hace que falten, produciendo un cuello de botella pues no hay disponibilidad de tokens, creando la posibilidad de descarte de paquetes.

Revisemos los parámetros que son configurables de este algoritmo:

- limit o latency: limit es el número de bytes en la cola que pueden estar esperando por tokens disponibles, también es posible definirlo de otra manera, especificando el tiempo máximo (latency) que puede estar un paquete en la TBF, este cálculo tiene en cuenta el tamaño del bucket, su velocidad y velocidad pico (peakrate, si está configurada).
- burst/buffer/maxburst: define el tamaño de la cola de tokens, es decir, el tamaño del bucket en bytes, este es la máxima cantidad de bytes para la cual existen tokens disponibles de manera inmediata.
- mpu: establece el número mínimo de tokens por paquete (minimum packet unit.
- rate: velocidad.
- peakrate: cuando existen tokens disponibles los paquetes que llegan son enviados inmediatamente, pero es posible regular esto si se desea, configurando la velocidad pico (peakrate) que establece la velocidad a la cual el bucket puede vaciarse.
- mtu/minburst: define la unidad máxima de transferencia.

Un ejemplo de configuración puede ser:

```
# tc qdisc add eth0 root tbf rate 250kbit latency 50ms burst 1540
```

Se configura la qdisc tbf como tipo egress y se define como velocidad 250kbit, con un tiempo de permanencia de los paquetes de 50ms y un buffer de 1540 bytes.

4.5.1.3 Stochastic Fairness Queuing.

Con esta disciplina de cola se busca distribuir de manera equitativa los datos transmitidos a la red, basa su funcionamiento en flujos, pues la distribución la realiza generando un número arbitrario de estos, para lograr esto usa una función hash la cual separa el tráfico en colas tipo FIFO que son mantenidas internamente, el tráfico es enviado en modo round-robin (por turnos), gráficamente se tiene:

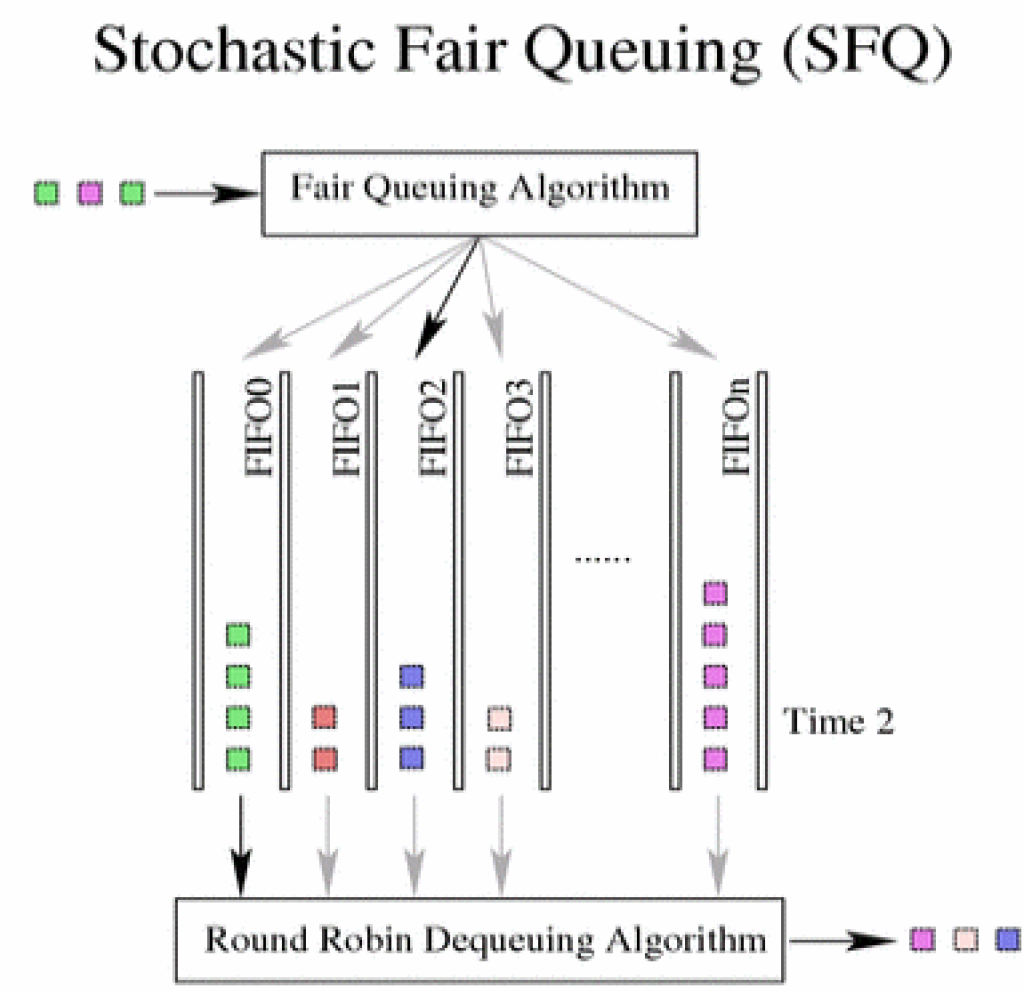


Figura 33. Disciplina de cola Stochastic Fairness Queuing [19]

Parámetros a configurar:

- perturb: permite configurar el tiempo según el cual se recalcula la función hash que se encarga de la distribución de los flujos de manera equitativa, si no se indica el hash no se reconfigura.
- quantum: define la cantidad de bytes de un flujo que se permiten sacar de una cola antes de la siguiente cola en turno. Su valor por defecto es 1 que es el máximo tamaño de paquete (MTU).

Un ejemplo de configuración de esta disciplina de cola puede ser:

```
# tc qdisc add dev ppp0 root sfq perturb 10
```

4.5.2. Disciplinas de colas con clases.

Este tipo de disciplinas brindan un mayor nivel de control y adquieren su valor cuando se trata de gestionar tráfico con diferentes prioridades, pues permiten al usuario configurarlas para dar tratamiento a cada tipo de tráfico. Los paquetes IP cuando llegan a

un tipo de disciplina de cola como esta deben ser enviados a unas clases que la componen, son entonces clasificados por medio de unos filtros que se encargan de decidir a que clases deben ser asignados los paquetes, es posible que el paquete deba ser sometido a nuevas clasificaciones (filtros) y clases hasta alcanzar su clasificación final.

Como se explicó antes las disciplinas de colas cuentan con unos manejadores (handles) que consisten de dos partes un número mayor y un número menor, cuyo formato es numero mayor : numero menor, se le asigna el numero 1:0 a la qdisc raíz o también 1: , podemos representar una jerarquía de clasificación como sigue:

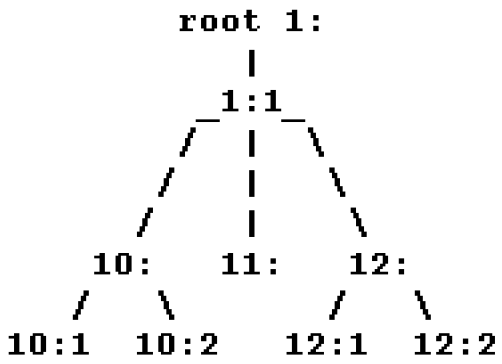


Figura 34. Posible estructura jerárquica de clasificación. [18]

No se debe interpretar que la estructura de clasificación tiene una topología en árbol, la realidad es que el kernel trata directamente con la qdisc raíz (root) pues es allí donde los paquetes entran o salen de la cola, se pueden tener esquemas como 1: > 1:1 > 12: > 12:2, o este 1: > 12:2.

Para enviar los paquetes a la interfaz, la disciplina de colas qdisc 1: recibe la petición de dar salida a los paquetes, esta se pasa a 1:1 que enseguida se pasa a 10: , 11: y 12: los cuales consultan a sus descendientes para desencolar los paquetes.

4.5.2.1 Filtros.

Para el caso de los filtros, lo que se hace es llamar una cadena clasificadora que se encarga de tomar la decisión sobre que clase debe procesar el paquete, esta cadena está formada por todos los filtros asociados a la disciplina de cola. Algunos filtros disponibles son:

- fw: se fundamenta en la manera en que el firewall (cortafuegos) marca el paquete, es sencillo y no requiere el uso de la utilidad tc.
- u32: su funcionamiento está basado en los campos del paquete IP.
- route: basa su funcionamiento por la ruta que tomara el paquete según la tabla de enrutamiento.
- rsvp, rsvp6: la decisión es tomada en las rutas de los paquetes basadas en RSVP.
- tcindex: se utiliza en la disciplina de colas DSMARK.

- Los argumentos más comunes de estos filtros son:
- protocol: define el protocolo que aceptará el filtro.
- parent: define el manejador (handle) al que está asociado el filtro, debe ser una clase ya existente.
- prio: establece la prioridad del filtro, los números menores se verifican primero.
- ·andle: es el manejador y difiere según el filtro.

A continuación solo se analizarán los filtros u32 y route por ser los más significativos:

Filtro U32 : Es el filtro mas avanzado, pues gracias a su versatilidad es posible definir muchos criterios para lograr el filtrado, lo cual lo hace el mas utilizado. Este permite definir estructuras basadas en cualquier conjunto de bits del paquete IP, ya sea en su encabezado o en campo de datos, algunos de los criterios son:

- Dirección IP de origen, dirección de destino del paquete.
- Puerto fuente, puerto destino de todos los protocolos IP.
- Protocolo utilizado: tcp, udp, icmp, gre, ipsec.
- El campo TOS.

La forma más simple del filtro u32 es una lista de registros consistentes en dos campos: un selector y una acción, se realiza entonces una comparación entre los selectores y el paquete IP que está siendo procesado hasta que ocurra una coincidencia y entonces la acción asociada se ejecuta.

Cuando se usa la utilidad tc para definir un filtro, esta consiste de tres partes: especificación del filtro, selector y acción.

tc filter add dev INTF [protocol PROTO] [(preference| priority) PRIO] [parent Q_KIND]

Se tiene entonces que protocol define el protocolo que se aplicará al filtro, preference (priority) establece la prioridad del filtro seleccionado, se puede disponer de varios filtros (listas de reglas) con diferentes prioridades, se pasará por cada lista en el orden en que las reglas fueron agregadas, entonces las listas con menor prioridad (con numero de preferencia mas alto) serán procesadas, el campo parent define la disciplina de cola raíz.

Selector U32: este selector contiene la definición del patrón con el cual será comparado el paquete actualmente procesado, concretamente lo que hace es definir cuales bits han de ser comparados en el encabezado del paquete, un ejemplo:

```
# tc filter add dev eth0 protocol ip parent 1:0 pref 10 u32 \
match u32 00100000 00ff0000 at 0 flowid 1:0
```

Al revisar esta línea desde la palabra clave match, se establece que el selector coincidirá con cabeceras IP cuyo segundo byte sea 0x10(0010), el numero 00ff es la mascara de comparación, que le indica al filtro que bits tiene que comparar, at define desde donde inicia la coincidencia con respecto al desplazamiento especificado en este caso el desplazamiento cero, iniciándose así desde el principio del paquete, la

coincidencia del paquete ocurrirá si su campo TOS tiene los bits de menor retraso activos.

La sintaxis del selector es como sigue:

match [u32 | u16 | u8] PATTERN MASK [at OFFSET | nexthdr + OFFSET]

Aquí u32, u16 y u8 definen la longitud en bits del patrón, PATTERN y MASK deben tener esta misma longitud, at OFFSET establece el ajuste o desplazamiento en bytes, para dar inicio a la comparación y nexthdr + OFFSET define de acuerdo al valor del desplazamiento el inicio del encabezado de la capa superior.

La descripción anterior corresponde a la definición de un selector general, como se puede ver es más complejo, existe el selector especifico que relaciona ítems tales como: dirección fuente/destino, puerto fuente/destino, campo TOS y protocolo.

Filtro Route: Este filtro basa su funcionamiento en los resultados de las tablas de enrutamiento, cuando un paquete que viaja a través de las clases llega a una que tiene marcado con el filtro route, entonces separa los paquetes basado en la información de las tablas de enrutamiento, la configuración de este filtro, se usa en combinación con la utilidad ip con la cual se generan las diferentes tablas de enrutado, la sintaxis es:

tc filter [add | del | change | get] dev STRING [parent PARENTID] [protocol PROTO] [prio PRIORITY] route

Un ejemplo:

tc filter add dev eth0 parent 1:0 protocol ip prio 100 route

Se definió un filtro tipo route, que se agregó al nodo padre con una prioridad 100, básicamente si el paquete llega a este nodo consultará la tabla de rutas y si coincide una, lo enviará a la clase dada con la prioridad especificada.

4.5.2.2 CBQ.

CBQ (Class Based Queueing) esta disciplina de cola con clases es la más compleja y empírica de todas cuando se trata de configurar, pues brinda la posibilidad de definir clases y realizar tareas de ajuste (shaper). Cuando se trabajo CBQ como ajustadora presenta inconvenientes, pues se dificulta determinar ciertos valores para realizar la configuración, por ejemplo si se tiene un enlace de 10 mbit/s y se desea ajustar a 1 mbit/s, se espera que el enlace esté el 90% del tiempo ocioso, pero si no es así se requiere acelerar para que realmente esté el 90% ocioso, esto es difícil de medir, pues CBQ se basa en un algoritmo de estimación del intervalo de tiempo transcurrido entre peticiones consecutivas del hardware para el envió de datos, en ocasiones no se consiguen aproximaciones buenas, perdiendo eficiencia, a continuación se definen algunos conceptos:

-idle: establece la diferencia entre el tiempo deseado y el actualmente medido entre la transmisión mas reciente de paquetes para los dos últimos paquetes enviados desde esta clase.

-avgidle: esta variable es el promedio de ocio (idle) y es calculado usando un EWMA (Exponential Weighted Moving Avarage), cuando avgidle es zero los paquetes llegan

exactamente una vez cada intervalo calculado, si es menor, la clase está sobre limitada (la clase ha estado excediendo su ancho de banda asignado en un corto intervalo de tiempo recientemente).

- maxidle: define el límite superior para avgidle.
- offtime: establece el intervalo de tiempo que un sobre limite debe esperar antes de enviar otro paquete.
- minidle: esta variable da el valor de límite inferior negativo para avgidle.

A continuación algunos parámetros para su configuración:

Operación para ajustar la regulación del ancho de banda: CBQ trabaja ajustando el ancho de banda al provocar que el enlace esté inactivo un porcentaje de tiempo determinado para así regular el ancho de banda al deseado, pero como se explicó el algoritmo no es preciso y requiere de experiencia para lograr buenos resultados, adicionalmente requiere de muchos parámetros que en muchos casos no se entiende que función cumplen, para hacerlo calcula el tiempo entre paquetes promedio, mientras la operación dura, el tiempo de ocio es medido usando EWMA (Exponential Weighted Moving Average) el cual considera que los paquetes recientes son exponencialmente mas importantes que los pasados. La carga media de UNIX se calcula de igual manera.

- avpkt: tamaño promedio de paquete en bytes, necesario para el cálculo del maxidle (relacionado con el tiempo de ocio).
- bandwidth: ancho de banda del dispositivo físico, se requiere para calcular el tiempo muerto entre peticiones.
- mpu: tamaño mínimo de paquete, se requiere porque incluso un paquete de cero bytes da lugar a un paquete de 64 bytes en Ethernet, el cual toma cierto tiempo para transmitirlos.
- rate: ancho de banda regulado con el cual se desea que funcione este disciplina de cola.
- cell: el tiempo que toma un paquete en ser transmitido sobre un dispositivo crece escalonadamente y está basado en el tamaño del paquete, por ejemplo dos paquetes uno de tamaño 800 y otro de 806 pueden tardar lo mismo en ser transmitidos, a menudo se define un valor de 8, este debe ser un múltiplo de potencias de 2.
- maxburst: define el número de paquetes que usan para calcular el maxidle, se usa para definir ráfagas.
- minburst: define el número mínimo de paquetes a enviar en una ráfaga.
- Operación para definir comportamiento como disciplina con clases: CBQ brinda la posibilidad de definir prioridades dentro de las clases que forman su estructura interna, de allí que cada vez que el hardware solicita un paquete, se inicia un proceso weighted round-robin, que empieza con las clases de menor prioridad, los parámetros que definen este proceso son:
- allot: determina la cantidad de datos que se pueden enviar por una interface, esto

ocurre cuando se le solicita a la CBQ externa enviar un paquete por la interface, se revisan las qdisc internas (en las clases) siguiendo el parámetro de prioridad, cuando a una clase le toca el turno solo podrá enviar la cantidad especificada.

- weight: ayuda en el proceso WRR, pues cada vez que una clase tiene el turno para enviar, si existen clases con ancho de banda significativamente superior que otras, tiene sentido permitir a estas enviar mas datos en una ronda que al resto.
- prio: define las distintas prioridades entre las clases que forman la estructura interna de la disciplina de cola.
- Operación para definir compartir o prestar ancho de banda entre diferentes clases: Tomando como punto de partida que CBQ tiene una limitación global al ancho de banda, es posible que entre las diferentes clases se presten ancho de banda, a continuación los parámetros:
- isolated/sharing: cuando se configura una clase con el parámetro isolated, se asegura que esta clase no prestará ancho de banda a sus hermanas, lo contrario se logra con el parámetro sharing, si se indica se asume que este último está activo.
- bounded/borrow: con bounded activo la clase no pedirá prestado ancho de banda a sus hermanas, mientras que borrow permite lo contrario, por defecto se asume que está activo este último.

Un ejemplo de configuración:

```
# tc qdisc add eth0 root handle 1:0 cbq bandwidth 10 Mbit cell 8 \
avpkt 1000
# tc class add dev eth0 parent 1:0 classid 1:1 cbq bandwidth 10 Mbit \
rate 6Mbit weight 0.6 Mbit prio 8 allot 1514 cell 8 maxburst 20 \
avpkt 1000 bounded
```

Inicialmente se agregó la disciplina de cola y se estableció como raíz, con un ancho de banda fijo de 10 Mbit, después se define una clase 1:1 a la cual se limita su ancho de banda a 6 Mbit.

Como se pudo apreciar configurar un disciplina de cola CBQ es mas complejo y requiere el uso de muchos parámetro, algunos con valores empíricos obtenidos como resultado de la experiencia en su uso.

4.5.2.3 HTB.

HTB (Hierarchical Token Bucket) es una disciplina de cola con clases similar a CBQ pero tienen sus diferencias, es mas sencilla de configurar, intuitiva y comprensible, es usada para controlar el ancho de banda dado en un enlace, permite usar un enlace físico para simular varios enlaces y enviar diferentes tipos de tráfico en los distintos enlaces simulados, requiere que se especifique como dividir el enlace físico en los enlaces simulados y decidir cual enlace simulado usar para que un paquete dado sea enviado.

HTB asegura que la cantidad de servicio proporcionado a una clase sea la cantidad

mínima requerida por esta y que efectivamente se le asignó, cuando una clase solicita menos de la cantidad asignada, el ancho de banda restante es distribuido a otras clases las cuales solicitaron el servicio.

HTB basa su principio de funcionamiento en los conceptos de tokens y buckets de un extremo a otro de un sistema basado en clases y filtros lo cual permite un control de tráfico más complejo y fino. Bajo un esquema complejo de préstamo, HTB brinda la posibilidad de realizar sofisticados técnicas de control de tráfico y para las formas fáciles se puede usar HTB para ajustarlo.

Lo que permite hacer esta disciplina de cola al usuario es definir las características de los tokens y buckets a usados y le posibilita anidar los buckets de manera arbitraria, cuando se acompaña de clasificación el tráfico puede ser controlado en forma granular.

El principio de funcionamiento de HTB se comprende mejor si se visualiza un árbol de clases, cada clase tiene su velocidad y prioridad, los paquetes son divididos en flujos los cuales son ligados a las hojas del árbol, cuando se habla de ajuste (shaping), este ocurre en las clases hoja (classe leaf), que están al extremo del árbol, es decir, no se lleva acabo ajuste alguno en las clases interiores o en las clases raíz (root), estas solamente existen para sugerir cómo el modelo de préstamo debe distribuir los tokens disponibles.

El mecanismo de préstamo es fundamental en esta disciplina de cola, pues ocurre que las clases hijas prestan tokens de sus padres una vez ellas han excedido la velocidad (rate), una clase hija continúa intentando prestar hasta que alcance un límite (ceil), en este punto empezará a encolar paquetes hasta que hallan tokens/ctokens disponibles, la siguiente tabla resume el concepto de préstamo:

Tabla 3. El mecanismo de préstamo en HTB [19]

Tipo de Clase	Estado De la Clase	Estado Interno de HTB	Acción Tomada
Leaf	< rate	HTB_CAN_SEND	La clase leaf desencola los bytes encolados hasta los tokens disponibles.
Leaf	> rate, <ceil	HTB_CAN_BORROW	La clase leaf intentara pedir prestado tokens/ctokens de la clase padre (parent), si hay tokens disponibles ellos serán prestados en incrementos según el quantum y las clases leaf desencolan hasta el valor cburst en bytes.
Leaf	> ceil	HTB_CANT_SEND	Los paquetes no son desencolados, lo cual causara retrasos e incrementara los retardos para encontrar la velocidad deseada.
Inner, root	< rate	HTB_CAN_SEND	La clase inner prestara tokens a las hijas.
Inner, root	> rate, < ceil	HTB_CAN_BORROW	La clase inner intentara tomar prestado tokens/ctokens de la clase parent, prestando estos a las hijas que compiten en incrementos quantum por solicitud.
Inner, root	> ceil	HTB_CANT_SEND	La clase inner no pedirá prestado tokens/ctokens de su padre y tampoco prestara a las clases hijas.

En la figura 35 se presenta gráficamente como ocurren los préstamos, además se puede ver la existencia de varias clases root que simulan circuitos virtuales:

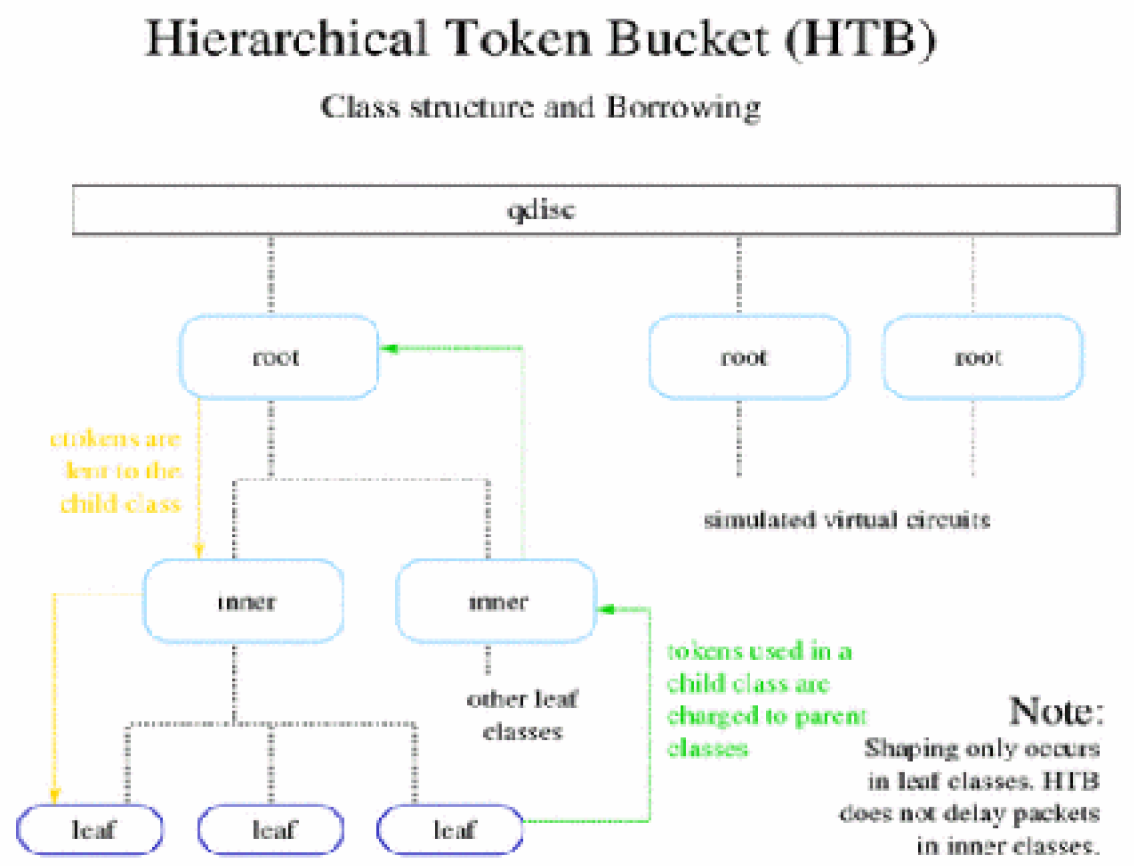


Figura 35. Estructura de clase y préstamo. [19]

Los parámetros de configuración de esta disciplina cola son:

- default: si este parámetro está en default 0 define que por cualquier causa un tráfico sin clasificar abandone la cola a velocidad del hardware, pasando por encima de cualquier clase ligada a la disciplina de cola root, este valor puede apuntar a una clase en particular, que se encargará de sacar dicho tráfico.
- rate: establece la velocidad límite mínima del tráfico transmitido, se puede entender como el ancho de banda garantizado para la clase leaf.
- ceil: usado para configurar la máxima velocidad deseada la cual será el límite del tráfico transmitido.
- burst: define la velocidad del bucket, HTB sacará de la cola burst bytes antes de esperar la llegada de mas ctokens.
- cburst: configura la máxima velocidad del bucket, HTB sacará de la cola cburst bytes antes de esperar la llegada de mas ctokens.
- quantum: este es el parámetro clave que usa HTB para controlar el préstamo, normalmente el valor correcto del quantum es calculado por HTB, si no es especificado por el usuario. Este parámetro es muy delicado pues tiene un gran efecto sobre el mecanismo de préstamo y el ajuste, pues es usado para dividir el tráfico entre las clases hijas sobre la velocidad y para transmitir los paquetes de esas

mismas clases.

- r2q: este valor es calculado por el usuario como pista para ayudar HTB en la determinación de quantum optimo para una clase en particular.
- mtu: define la máxima unidad de transferencia.
- prio: establece la prioridad.

Algunas observaciones a tener en cuenta cuando se use HTB:

- El ajuste con HTB solo ocurre en las clases leaf.
- Debido a que HTB solo realiza ajustes en las clases leaf, la suma de las velocidades (rates) de las clases leaf no debe exceder la velocidad ceil de la clase padre (parent). Idealmente la suma de velocidades (rates) de las clases hijas debería coincidir con la velocidad (rate) de la clase padre, para permitir a la clase parent distribuir el ancho de banda restante (ceil – rate) a la mayoría de clases hijas.
- La cantidad quantum solo es usada cuando la clase está por encima de la velocidad rate y por debajo de ceil.
- Los quantum deberían ser configurados en MTU o mayor, debido a que HTB sacará de la cola al menos un solo paquete por servicio aun si el quantum es demasiado pequeño. En tal caso este no será capaz de calcular de manera precisa el ancho de banda real consumido.
- Las clases padres (parents) prestan tokens a las clases hijas en incrementos de quantum, para maximizar la granularidad e igualmente de manera instantánea el ancho de banda distribuido, el quantum debería ser tan bajo como sea posible mientras no sea menor que el MTU.

Un ejemplo de configuración:

```
# tc qdisc add dev eth0 root handle 1: htb default 11
# 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 80kbps \
ceil 100kbps
# tc class add dev eth0 parent 1:1 classid 1:11 htb rate 20kbps \
ceil 100kbps
```

La primera línea agrega la qdisc HTB al dispositivo de red como root (egress) le asigna el handle 1: que se usa mas abajo como referencia y asigna a la clase 11 todo el tráfico que no sea clasificado, la segunda línea define la clase root 1:1 que está debajo de la qdisc 1: , que es la que permitirá a las clases hijas prestarse entre si, las restantes líneas configurar clases hijas en las que se configura el ancho de banda 80kbps y 20kbps.

4.5.2.4 PRIO.

La disciplina PRIO (Priority) organiza el tráfico por prioridades, es decir no realiza ajustes a este. Por defecto esta cola define tres clases y cada una tiene asociada una nueva disciplina de colas tipo FIFO, estas tienen una prioridad, los paquetes que correspondan a la clase 1 se envían antes que la clase 2 y estos a su vez se envían antes que los de la clase 3.

Pero PRIO ofrece la posibilidad de definir filtros para estas clases, no se está limitado solo a clasificar los paquetes por el campo TOS, es factible realizar clasificaciones mas complejas, si bien las clases tienen una disciplina de cola tipo FIFO es posible definir la disciplina de cola que se quiera.

Los parámetros de esta disciplina son:

- bands: este permite crear el número de bandas que se desean, por defecto son tres.
- priomap: si no sea desea adjuntar ningún filtro, la decisión de clasificación puede estar basada en el campo TOS, este parámetro permite realizar el mapeo entre las prioridades del kernel y clases.

Un ejemplo de configuración de uso de esta disciplina:

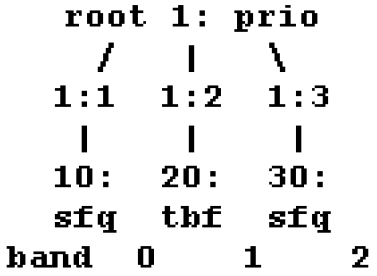


Figura 36. Diagrama Árbol aplicación de la disciplina de cola PRIO [18]

Se tiene el anterior árbol figura 10, en donde el tráfico interactivo va a 10: 0 a 20: , el resto es manejado por 30: .

tc qdisc add dev eth0 root handle 1: prio

tc class add dev eth0 parent 1:1 handle 10: sfq

tc class add dev eth0 parent 1:2 handle 20: tbf rate 20kbit buffer 1600 \

limit 3000

tc class add dev eth0 parent 1:3 handle 30: sfq

La primera línea define le disciplina de cola, y por defecto establece las clases 1:1, 1:2 y 1:3, las siguientes líneas agregan las clases con una disciplina de cola adjunta para controlar el tráfico para cada banda.

5. IMPLEMENTACIÓN BÁSICA DE CONTROL DE TRÁFICO

En los capítulos anteriores se han presentado las bases teóricas y conceptos relacionados con teoría de colas, control de tráfico y enrutamiento avanzado y la teoría de su implementación por medio del sistema operativo Linux.

En este capítulo se desarrollará con un ejemplo práctico el control de tráfico sobre una red, usando una máquina con sistema operativo Linux como el dispositivo que realiza dicha labor de control. La metodología usada será la de una práctica tipo laboratorio en la cual, mediante un esquema sencillo de red, se evidencia el potencial existente en el área de control de tráfico usando LINUX y las diversas herramientas con que cuenta dicho sistema operativo.

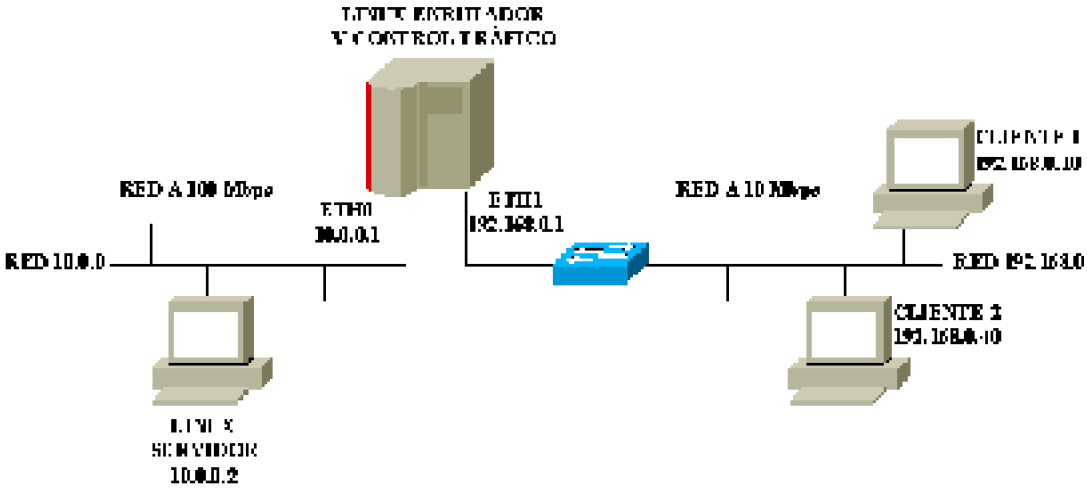


Figura 37 Esquema del montaje para control de tráfico.

En la figura 37 se presenta el esquema de red utilizado: Se tienen dos segmentos de red diferentes, en uno de los cuales está ubicado un Servidor y en el otro se ubican clientes de los servicios implementados. Entre las dos redes se tiene una máquina que sirve de Gateway a ambas redes y que a su vez servirá como dispositivo de control de tráfico. Los resultados que se obtengan a partir de este esquema, a pesar de su sencillez, pueden hacerse extensivos a otro tipo de configuraciones, en donde se tiene un conjunto de clientes, accediendo a servicios ubicados en otra red (P.ej Internet).

Esto se ilustra en la figura 5.2

Para la implementación que se muestra en la figura 37, se hará control de tráfico sobre la interfaz ETH1 de la máquina Linux, es decir, se controlará el tráfico que el servidor Linux en la red a 100 Mbps le envía a los clientes en la red a 10 Mbps. Esto de la siguiente manera:

1.Control de tráfico sobre la Interfaz ETH1, independiente del tipo de tráfico y del destino del mismo.

2.Control de tráfico sobre la Interfaz ETH1, dependiendo del destino del mismo.

3.Control de tráfico sobre la Interfaz ETH1, dependiendo del tipo de tráfico.

4.Control de tráfico sobre la Interfaz ETH1, dependiendo del tipo de tráfico y de su destino.

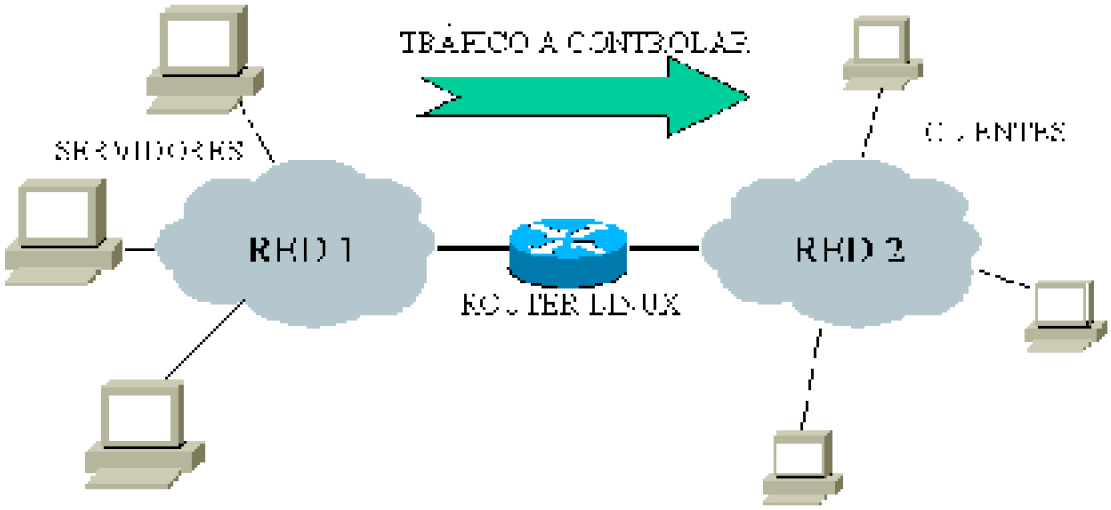


Figura 38 Esquema general para control de tráfico

DESCRIPCIÓN DE LOS COMPONENTES DE LA RED.

Red 10.0.0: Cable cruzado entre el servidor Linux y la interfaz ETH0 del Enrutador Linux.

Red 192.168.0:Switch 10/100. La conexión del Enrutador Linux es a través de la interfaz ETH1 y es a 100 Mbps. La conexión de los clientes se hizo a 10 Mbps.

Servidor Linux:

Hardware:

Procesador Pentium IV 1.6 GHz

256 MB de RAM

Tarjeta de red integrada PRO/100 VE de Intel(R) (ETH0)

Software:

Sistema Operativo RED HAT LINUX 9.0 Versión Kernel 2.4.20-8.

Servidor WEB Apache: HTTPD versión 2.0.40-21

Servidor FTP: VSFTPD versión 1.1.3-8

Servidor SSH: OpenSSH versión 3.5p1.

NTOP: ntop-2.2-0.dag.rh90.

Cliente SNMP: net-snmp-5.0.6-17

Enrutador Linux:

Hardware.

Computador Desktop Procesador AMD Duron 1 GHz.

512 MB de RAM

Tarjeta de Red Integrada 10/100 SIS 900. (ETH0)

Tarjeta de Red 10/100 VIA Technologies. (ETH1)

Software:

Sistema Operativo RED HAT LINUX 9.0 Versión Kernel 2.4.20-8.

IPROUTE2: iproute2-ss010824.

NTOP: ntop-2.2-0.dag.rh90. Herramienta para S.O. LINUX que permite el monitoreo y análisis de tráfico.

Cliente SNMP: net-snmp-5.0.6-17

Cliente 1

Hardware

Computador Portátil con procesador Intel Pentium IV de 1.7 GHz.

256 MB RAM.

Tarjeta Integrada de Red 10/100 3Com 3C920.

Software:

Sistema Operativo Windows 2000 5.00.2195 Service Pack 4

Internet Explorer 6.0.2800.1106CO

WINFTP versión 2.52

SSH Secure Shell versión 3.2.9

Agente de captura SNMP.

WhatsUp GOLD para Windows. Versión 7.03

Cliente 2

Hardware

Computador Portátil con procesador Pentium III de 850 MHz

128 MB RAM

Tarjeta Integrada de Red Intel PRO/100 + MiniPCI

Software:

Sistema Operativo

Internet Explorer

WINFTP versión 2.52

SSH Secure Shell versión 3.2.9

Agente de captura SNMP.

Switch 10/100

Switch nivel 2 de 8 puertos 10/100.

5.2 Configuración y herramientas de Software

A continuación se hace una descripción mas detallada del montaje, incluyendo el software necesario en la red para hacer la toma de datos y verificar los resultados del control de tráfico implementado.

Dado que se hará control de tráfico sobre la interfaz ETH1 del enrutador Linux (Figura 37), se requiere tener en la red una herramienta de monitoreo que esté reportando la cantidad de tráfico que pasa por dicha interfaz.

Igualmente se debe poder monitorear el tráfico en las máquinas clientes, de manera que permita una discriminación de dicho tráfico según parámetros de los paquetes IP, como mínimo: Direcciones IP, Protocolos.

En este caso se escogieron las siguientes herramientas:

5.2.1 NTOP

Aplicación para Linux que permite mostrar el uso de la red, siendo capaz de listar los usuarios de la misma y reportar información concerniente al tráfico (IP y no IP) generado por cada usuario. Dicho tráfico es clasificado por NTOP de acuerdo al usuario y los protocolos. NTOP nació 1998, para satisfacer la necesidad de su creador Luca Deri, quien requería de una herramienta para monitorear el tráfico en una red, pues las que existían no satisfacían sus necesidades. Los protocolos que pueden ser monitoreados son:

TCP/UDP/ICMP

(R) ARP

IPX

DLC

Decnet

Apple Talk

Netbios

TCP/UDP

FTP

HTTP

DNS

Telnet

SMTP/POP/IMAP

SNMP

NFS

X11

Se requiere de un browser web para poder acceder a la información capturada por el NTOP, mientras este se está ejecutando es posible que varios usuarios puedan acceder a la información de tráfico, usando un browser web convencional, si se desea hacer un acceso seguro, NTOP requiere de la librería openSSL.

Para hacer uso de NTOP es necesario de un conjunto de herramientas externas, algunas de ellas son opcionales pero ayudan a mejorar las posibilidades de la aplicación de acuerdo al uso que se le quiera dar.

Las librerías requeridas incluyen:

·libpcap

·gdbm

Librerías opcionales:

·gchart

·gd

·libpng

·rrdtool

·sFlow

La versión utilizada de NTOP para este montaje es ntop-2.2-0.dag.rh90. en general se puede decir que NTOP es un sniffer con características que permiten visualizar las estadísticas de tráfico mediante un navegador y es ideal para tareas donde se requiera conocer el grado de utilización de una red en periodos de monitoreo prolongados, debido a esto se hizo necesario el uso de una herramienta adicional que presentara un gráfico en el tiempo del tráfico total cursado por cada maquina en la red, dicha aplicación es la What's Up que a continuación se describe.

5.2.2 WhatsUp Gold

Sistema de monitoreo para Windows diseñado para redes multiprotocolo, el cual permite monitorear los dispositivos y servicios críticos en una red, generando alarmas visibles y/o audibles cuando se detecta un problema. Dos tipos de datos son guardados por la aplicación: Cambios en el estado de la red (llamados eventos), tales como que un dispositivo falló y estadísticas de poleo de cada dispositivo. Los eventos pueden filtrarse por dirección IP, nombre del dispositivo o descripción del evento y llevarse a gráficas estadísticas que muestran la disponibilidad del dispositivo.

También se cuenta con algunas características adicionales de monitoreo, tales como el monitoreo SNMP, el cual se usa para monitorear la actividad en un dispositivo y sus interfaces, lo que permite mostrar el tráfico en tiempo real.

SNMP (Simple Network Management Protocol) es un estándar de Internet que

permite la gestión de datos en diferentes dispositivos de red para ser leídos y monitoreados por una aplicación. SNMP define un método mediante el cual un usuario remoto puede ver o cambiar la información de gestión de un dispositivo que está en red (Host, Gateway, Servidor, etc). Una aplicación de gestión en un sistema remoto de usuario usa el protocolo para comunicarse con el agente SNMP en el dispositivo en el cual se quiere acceder a los datos. El agente SNMP en cada dispositivo puede proveer información acerca de la configuración y operación del dispositivo de red, tales como: las interfaces de Red del dispositivo, tablas de enrutamiento, paquetes IP enviados, recibidos y perdidos, etc. Esta información, llamada Objetos SNMP es almacenada en un formato estándar definido en la Base de información de gestión (MIB: Management Information Base). En el MIB se definen los objetos SNMP que pueden ser gestionados y el formato de cada objeto.

Se puede usar entonces WhatsUp Gold para ver y monitorear objetos SNMP en los dispositivos que tengan implementado el agente SNMP. Obviamente solo se podrán monitorear aquellos objetos que estén implementados en el agente SNMP remoto.

La versión utilizada para este montaje de WhatsUp Gold es la 7.03

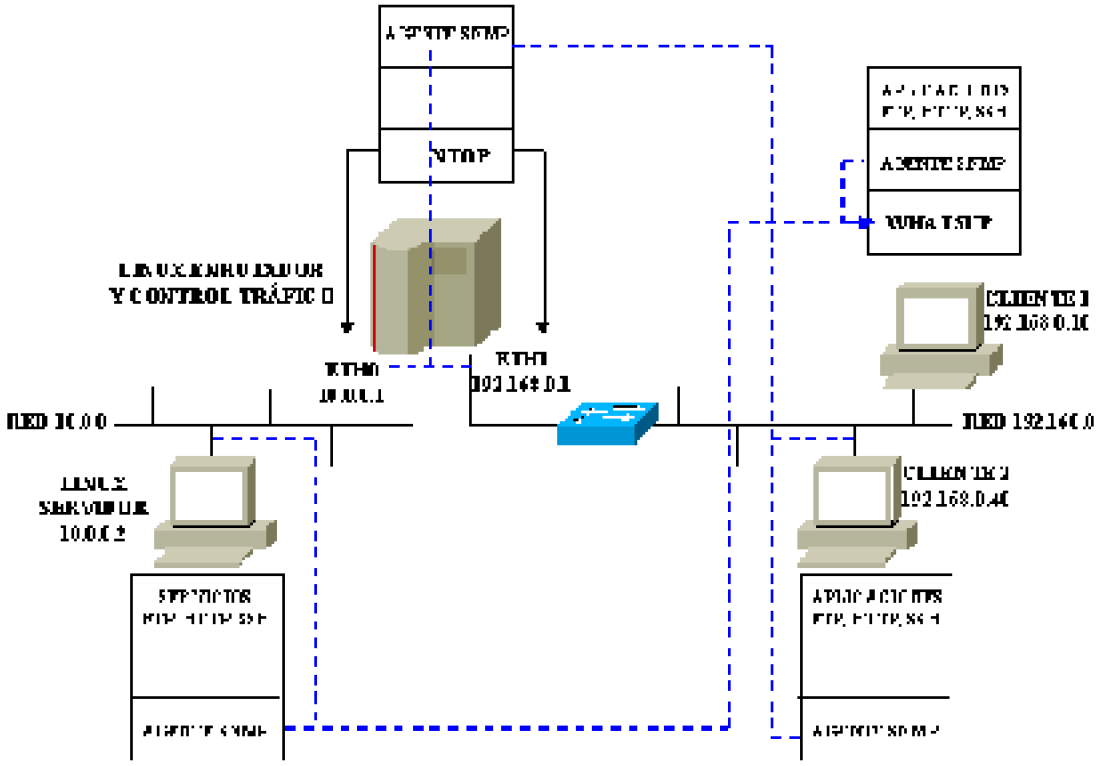


Figura 39 Esquema de tráfico de monitoreo en la red

NTOP y WHATSUP, las dos aplicaciones usadas para las tareas de monitoreo en la red tienen cada uno filosofías y formas de despliegue de datos diferentes. En el la figura 39 se ilustra el flujo de datos relacionado con dichas aplicaciones.

NTOP se comporta como un Sniffer que procesa la información recogida de las

interfaces ETH0y ETH1 en la máquina Linux. Como Sniffer NTOP analiza los paquetes IP que se cursan por las interfaces de red de la máquina, identificando campos en dichos paquetes, tales como direcciones IP, y puertos tanto origen como destino, lo que le permite desplegar información tal como:

- Promedio de tráfico entrante y saliente que se cursa por dirección IP.
- Promedio de tráfico entrante y saliente que se cursa por por protocolo.
- El troughput de la red, etc.

Obsérvese en la figura 39 que al funcionar como un Sniffer, NTOP solo puede mostrar estadísticas del tráfico que pase por la interfaz ETH1 (hablando del segmento de red correspondiente a la subred 192.168.0), y debido a la topología en dicho segmento, en el cual hay un Switch, el tráfico cursado entre el cliente 1 y el cliente 2 no podría ser detectado. Sin embargo acá el interés se centra en el tráfico que se cursa desde el servidor hacia los clientes y el único tráfico entre cliente 1 y 2 corresponde a paquetes SNMP.

Las gráficos extraídos de NTOP que se muestran, corresponden a datos resultado del análisis del tráfico que se cursa por la interfaz ETH1, que es sobre la que se hace el control de tráfico. Básicamente en dichos gráficos se observa el resumen del tráfico que pasa por dicha interfaz, discriminado por dirección IP: NTOP analiza los paquetes de acuerdo a su dirección IP y los contabiliza, pudiendo dar una idea de la cantidad de tráfico que se está cursando en la red.

Otro tipo de gráfico que se muestra, y por el que se hizo uso de esta herramienta, consiste en el resumen de la distribución de tráfico tanto por protocolos a nivel global (sin filtrar por dirección IP) como por host (filtrando tanto dirección IP como puertos)

WHATSUP se basa en SNMP, en donde, de acuerdo a una configuración previa de comunidad SNMP, las máquinas “se reportan” ante el programa, enviando datos a la máquina destinada para tal en la red y denominada trap server. En este caso se escogió como trap server la máquina Cliente 1. En la figura 39 se observa como se establece una comunicación de cada agente SNMP en cada máquina con el trap server de manera que la información esté disponible en este último.

Dado que el interés del montaje es medir el tráfico, se opta por recoger información de cada agente SNMP concerniente a las tarjetas de red de cada máquina y mas exactamente los paquetes entrantes y salientes de dichas interfaces.

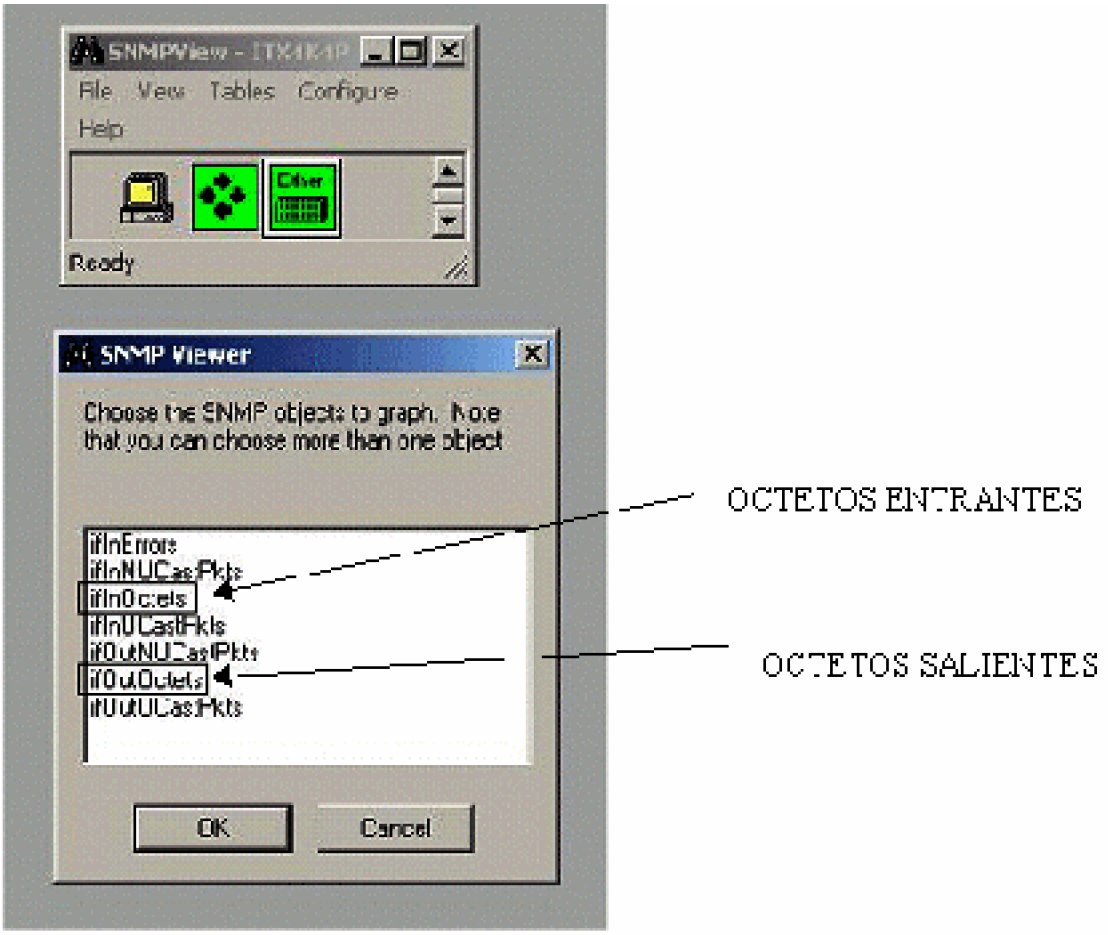
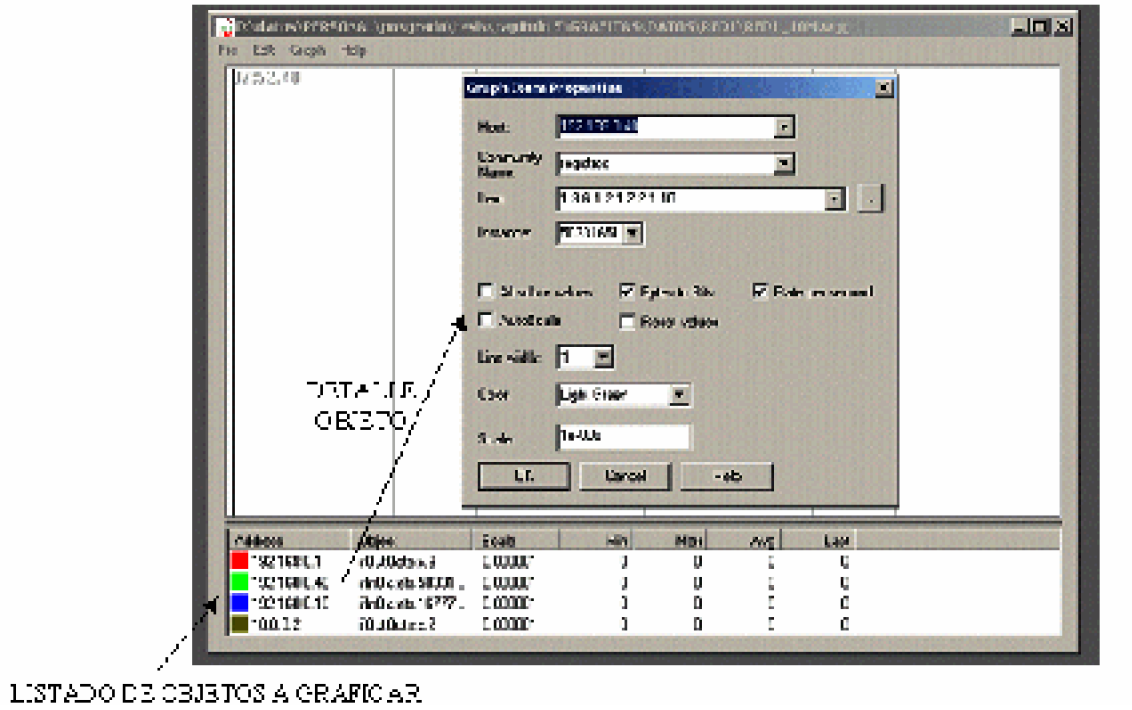


Figura 40. Objetos a monitorear en las máquinas con WHATSUP



En la figura 40 se observa cuáles son los parámetros que se pueden graficar de la interfaz Ethernet. Tal como se explicó antes, dichos parámetros dependen de la implementación SNMP de cada máquina.

Igualmente se muestra una imagen de la parte de la aplicación WAHTSUP usada para hacer el despliegue gráfico de la información: el SNMP GRAPH UTILITY allí se pueden recopilar objetos SNMP para que los valores en el tiempo sean desplegados en una misma gráfica.

Cuando se escoge un objeto SNMP para ser graficado, se debe ser cuidadoso con los siguientes parámetros:

HOST: Dirección IP de la máquina o dispositivo donde reside el objeto.

Community Name: Nombre de la comunidad dado en la configuración SNMP de la red.

Item: Se refiere al identificador del objeto a monitorear.

Instance: Se refiere a un subíndice de dicho objeto, dependiendo de la configuración del objeto monitoreado. Por ejemplo, si se cambia la velocidad de conexión de la tarjeta de red, cambiará la instancia de dicho objeto y el ítem seguirá siendo el mismo.

Si se activan las casillas de Bytes to Bits y Rate Per Second se asegura que la gráfica mostrará valores de tráfico correspondientes a las unidades bps.

La utilidad SNMP GRAPH también permite guardar los datos en archivos de Excel para su posterior análisis.

Como se puede apreciar en la figura 40, los objetos SNMP que se escogieron monitorear son:

- Tráfico Saliente del Servidor Linux.
- Tráfico saliente del equipo Linux con que se hace control de tráfico.
- Tráfico entrante en los dos clientes.

Esta selección permite ver el comportamiento de la red en cuanto al punto de interés que es controlar el tráfico que viajará desde el servidor, pasando por la maquina que hace control de tráfico.

5.3 Caracterización del tráfico a controlar.

Con el fin de establecer un punto de referencia, se realizó un análisis de la red propuesta, en donde se monitoreó el comportamiento de la misma al generar tráfico sobre la red sin tener ninguna tarea de control de tráfico ejecutándose. Dicho análisis permite establecer:

- Capacidad de la red para las aplicaciones y servicios implementados.
- Comparación de la utilización de la red de los diversos servicios.

- Comparación de la utilización de la red de las máquinas cliente.

Para esto se utilizaron archivos residentes en el servidor Linux, cada uno con un tamaño de 638 Mb. Por medio de tres protocolos diferentes se hizo la transferencia de dichos archivos hacia cada uno de los clientes. Los protocolos usados fueron: HTTP, FTP y SSH.

El procedimiento usado consistió en observar el comportamiento de la red en los siguientes casos:

- Un cliente a la vez haciendo uso de cada protocolo.
- Los dos clientes haciendo uso a la vez de cada protocolo
- Un cliente a la vez haciendo uso de los tres protocolos.
- Los dos clientes haciendo uso de los tres protocolos

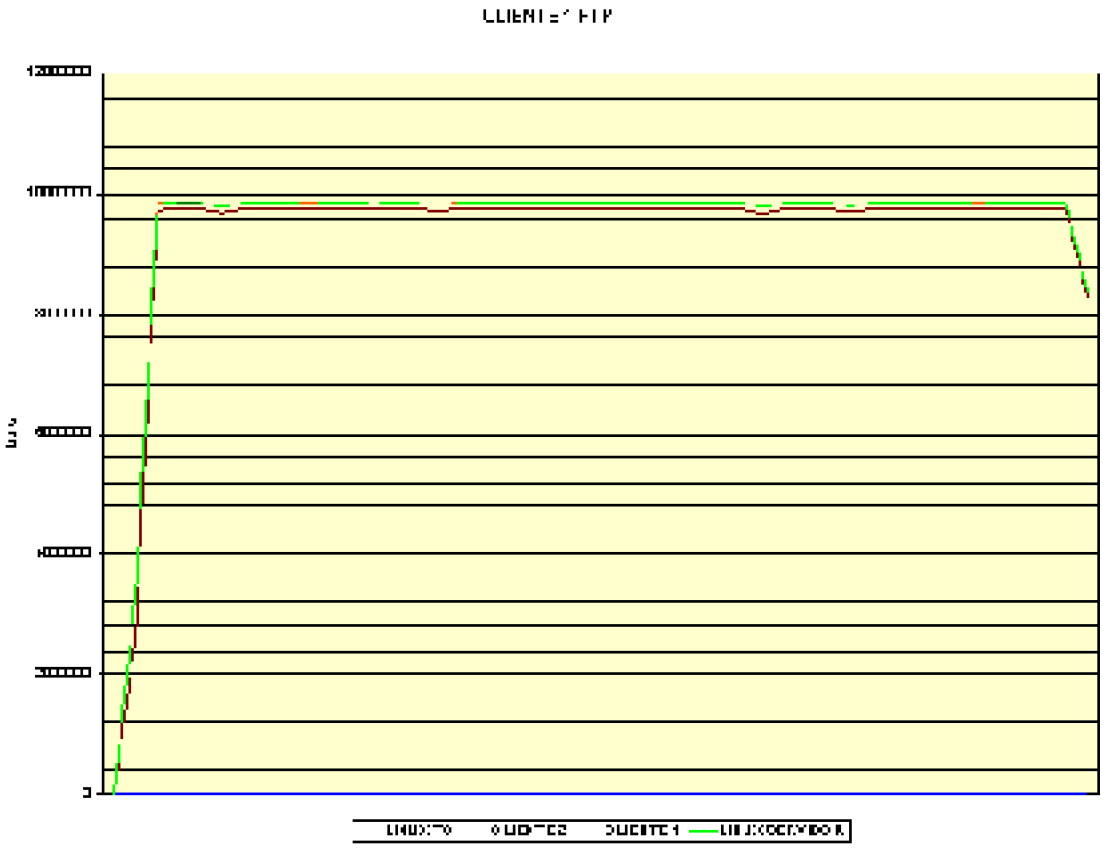
Datos obtenidos para cada cliente individualmente.

5.3.1 Cliente 1.

En las figuras 41, 42 y 43 se puede apreciar el comportamiento del tráfico cuando este cliente hace uso de un solo protocolo en la red. Se observa que optimiza el uso de ancho de banda disponible, independiente del protocolo usado, el tráfico alcanzó valores alrededor de 9.6 Mbps y las gráficas muestran un comportamiento “plano” en el tiempo con respecto al uso del ancho de banda.

En la figura 44 se observa dicho comportamiento cuando el Cliente 1 hace uso de los tres protocolos al tiempo. Nuevamente se nota un aprovechamiento máximo del ancho de banda disponible, el tráfico promedio es de 9.6 Mbps y la tendencia en este cliente es distribuir equitativamente el tráfico entre los tres protocolos.

Datos tomados con WHATSUP



Datos tomados con NTOP

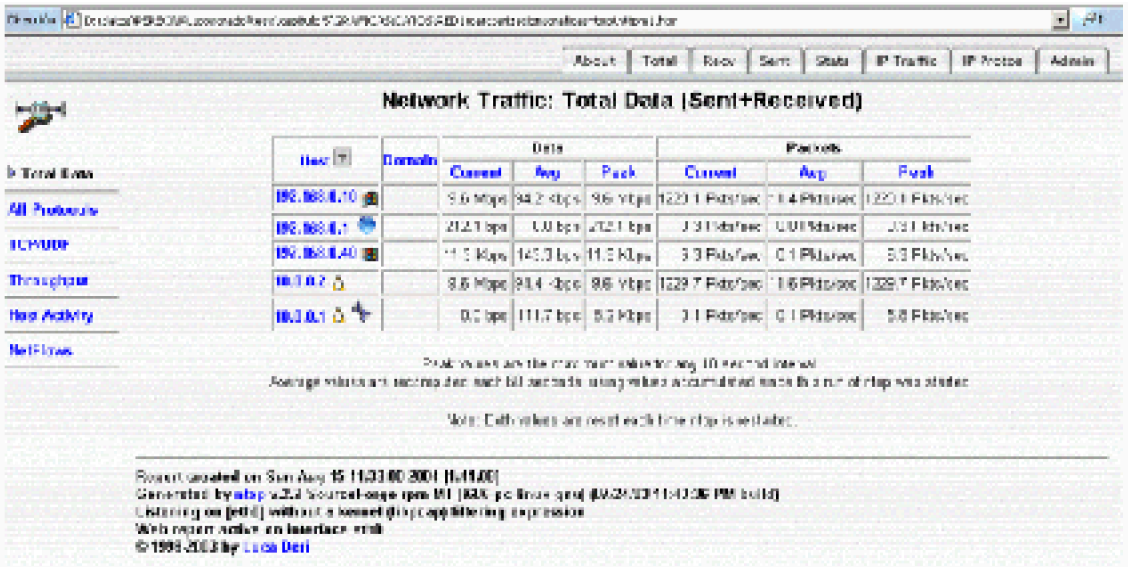
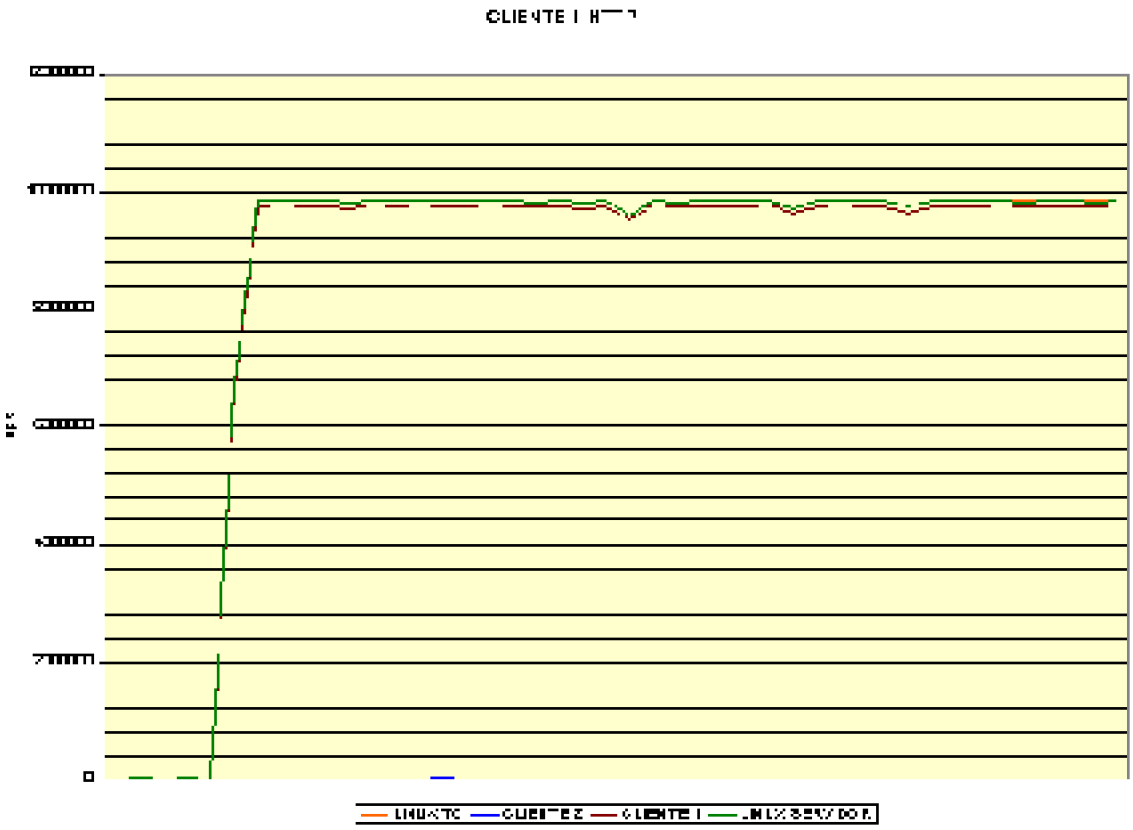


Figura 41. Caracterización tráfico debido al Cliente 1. Protocolo FTP

Datos tomados con WHATSUP



Datos tomados con NTOP

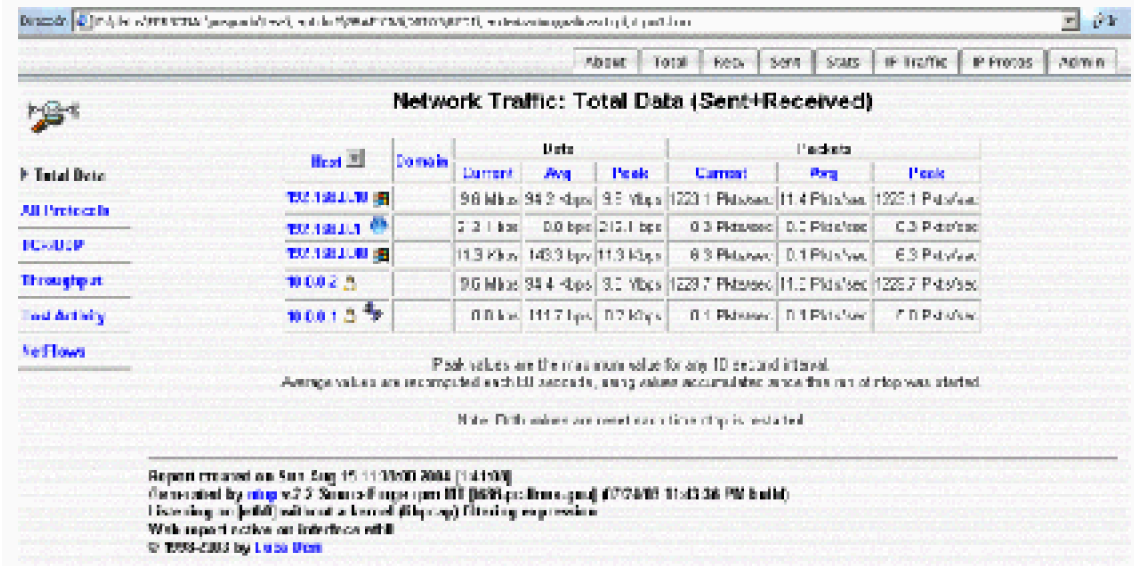


Figura 42. Caracterización tráfico debido al Cliente 1. Protocolo http

Datos tomados con WHATSUP

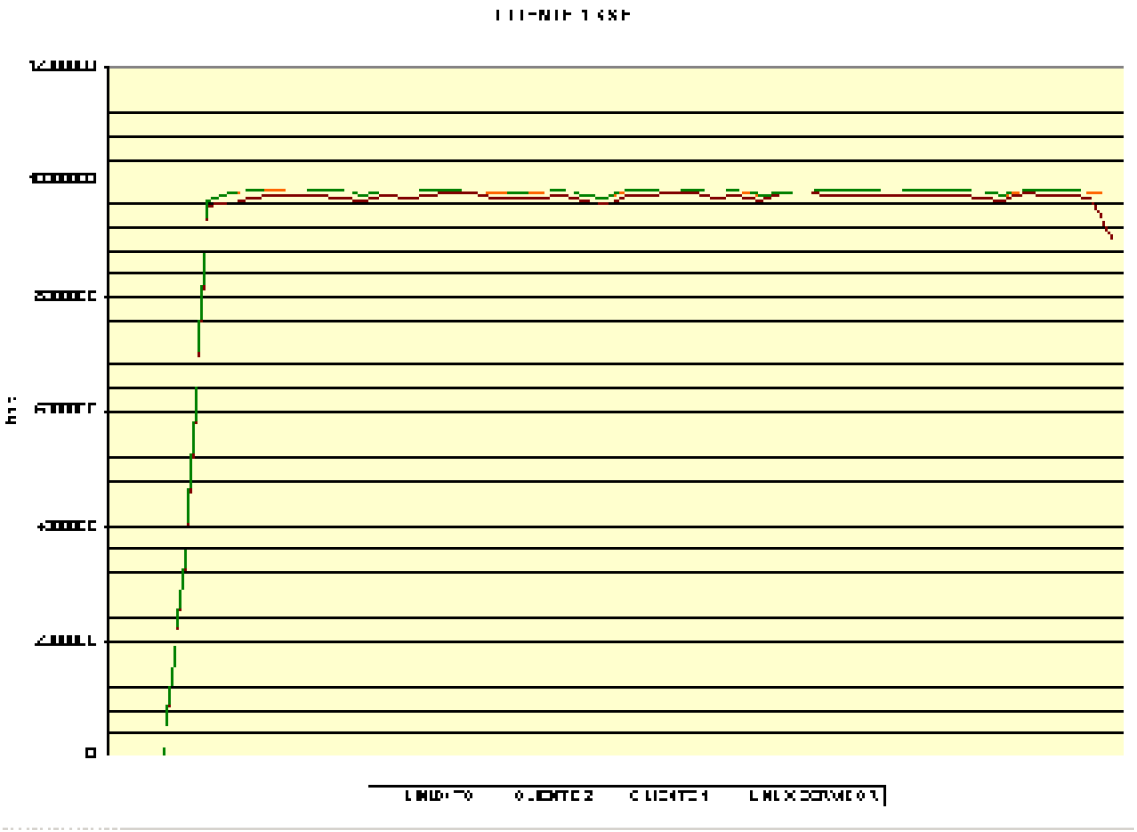
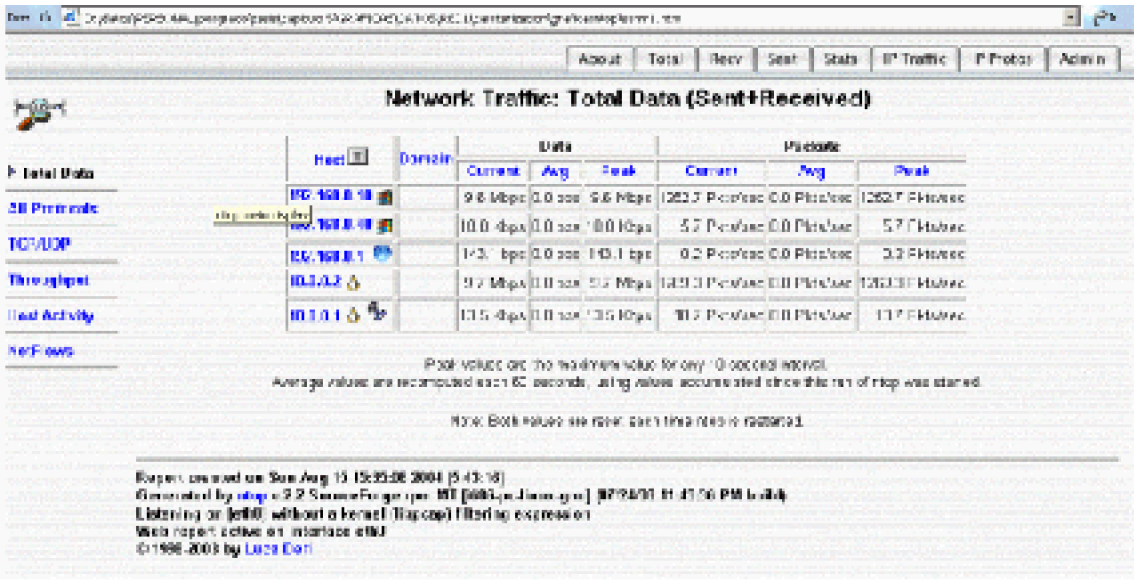


Figura 43. Caracterización tráfico debido al Cliente 1. Protocolo SSH

Datos tomados con NTOP



Datos tomados con WHATSUP

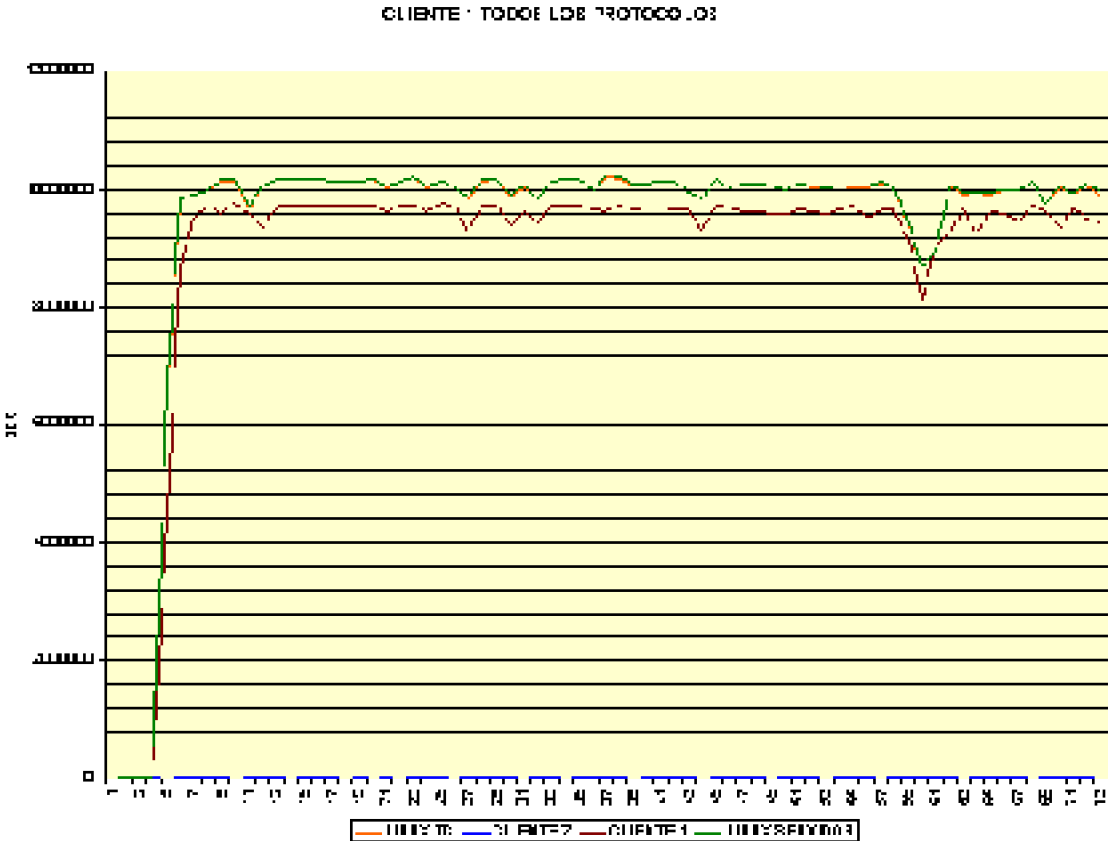
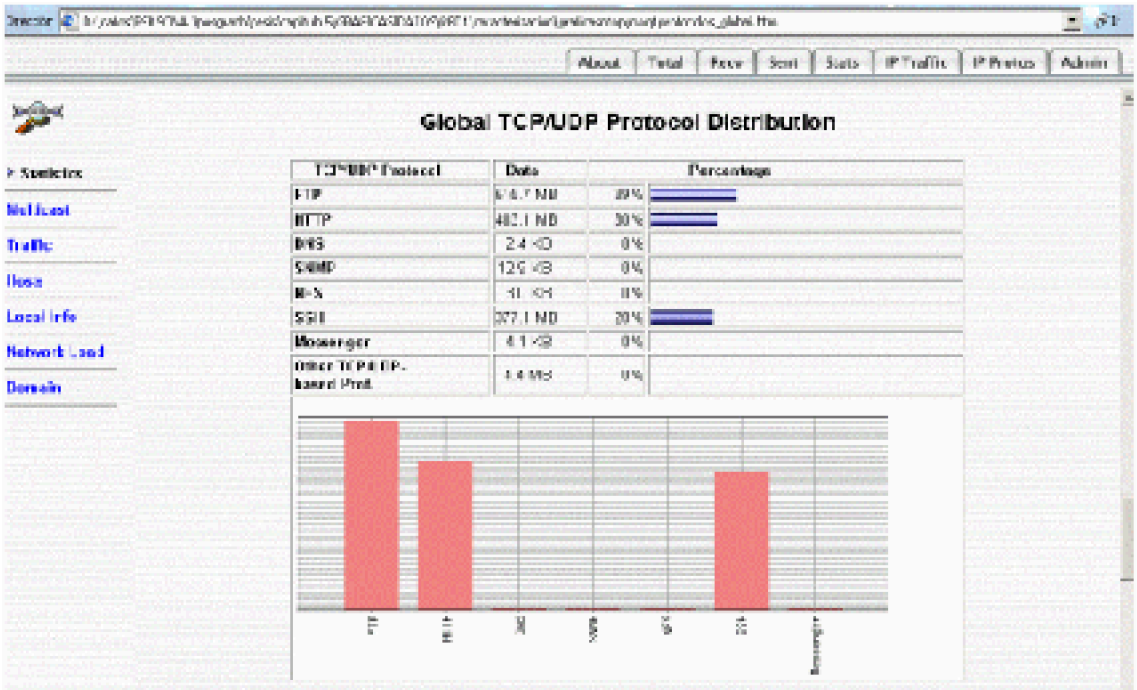


Figura 44. Caracterización tráfico debido al Cliente 1. Protocolos FTP, HTTP y SSH

Datos tomados con NTOP



5.3.2 Cliente 2

En las figuras 45, 46 y 47 se puede apreciar el comportamiento del tráfico cuando este cliente hace uso de un solo protocolo en la red. Para esta máquina se observó un comportamiento diferente al cliente 1, ya que acá se observa dependencia del protocolo y la respuesta en el tiempo no es “plana” en lo que refiere al aprovechamiento del ancho de banda. El tráfico promedio para cada protocolo fue:

- FTP: 5.3 Mbps
- HTTP: 8.2 Mbps
- SSH: 2.3 Mbps

En la figura 48, se observa el comportamiento del tráfico cuando el cliente 2 hace uso de los tres protocolos al tiempo. Se nota que se ve afectado el aprovechamiento del ancho de banda disponible, el tráfico promedio es de 2.85 Mbps y no hay distribución equitativa del tráfico entre los tres protocolos en el Cliente.

Datos tomados con WHATSUP

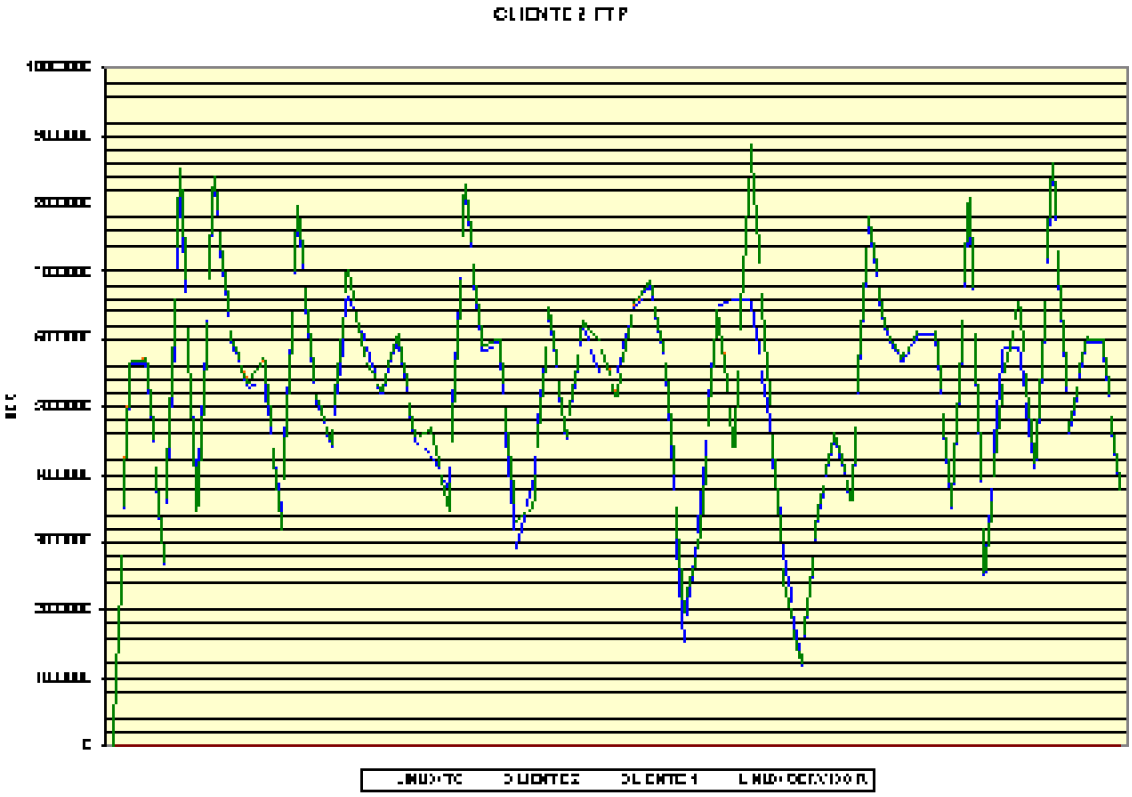
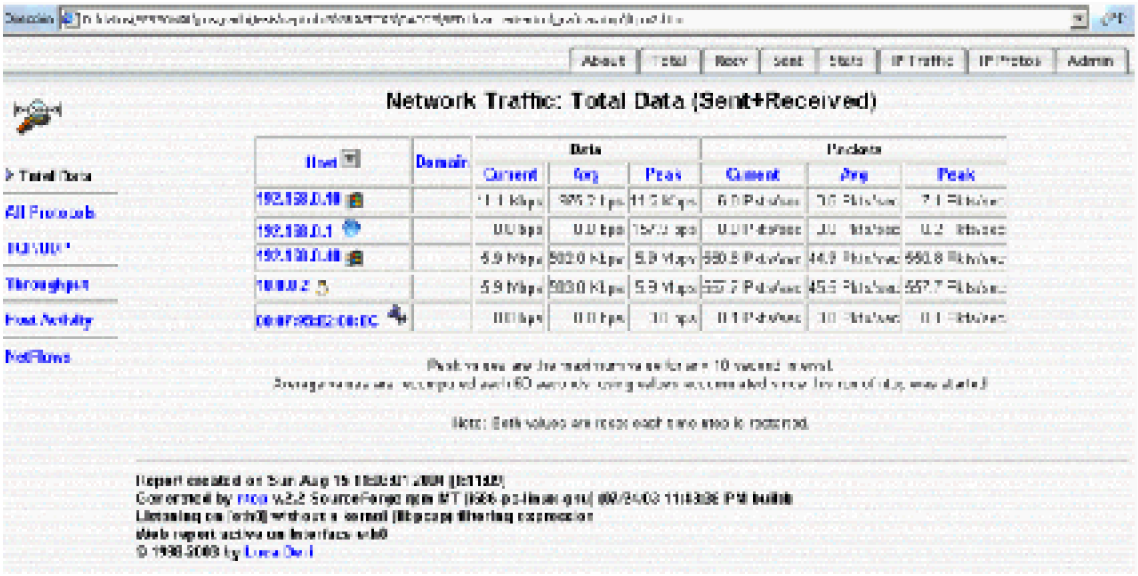


Figura 45. Caracterización tráfico debido al Cliente 2. Protocolo FTP

Datos tomados con NTOP



Datos tomados con WHATSUP

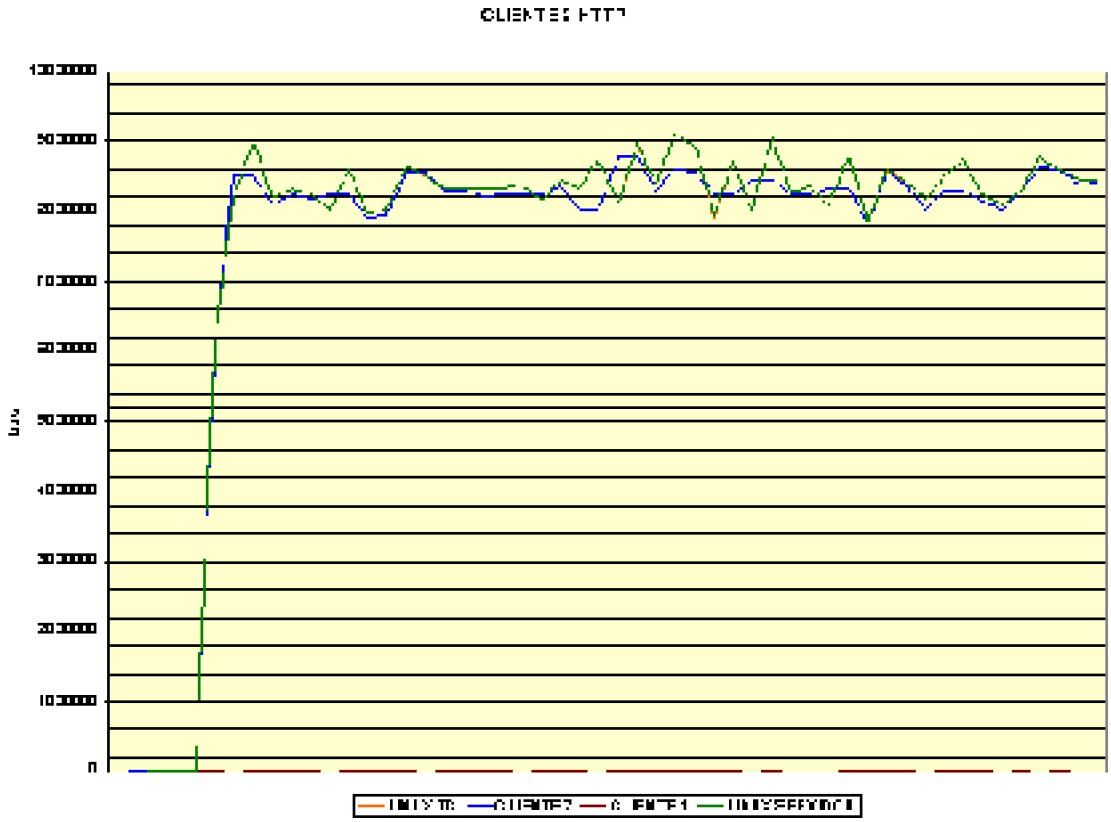
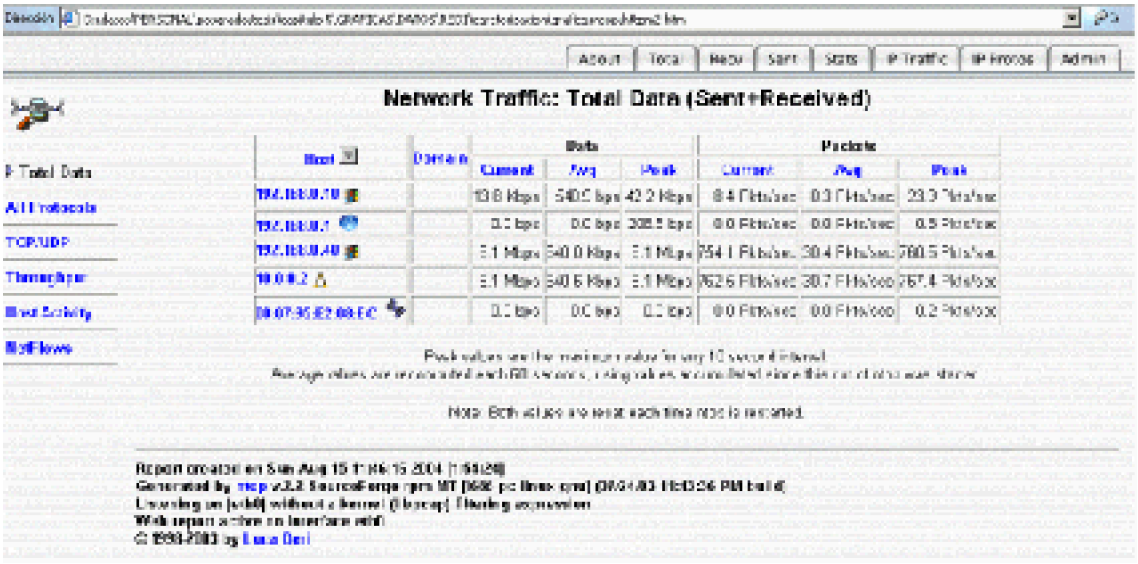


Figura 46. Caracterización tráfico debido al Cliente 2. Protocolo http

Datos tomados con NTOP



Datos tomados con WHATSUP

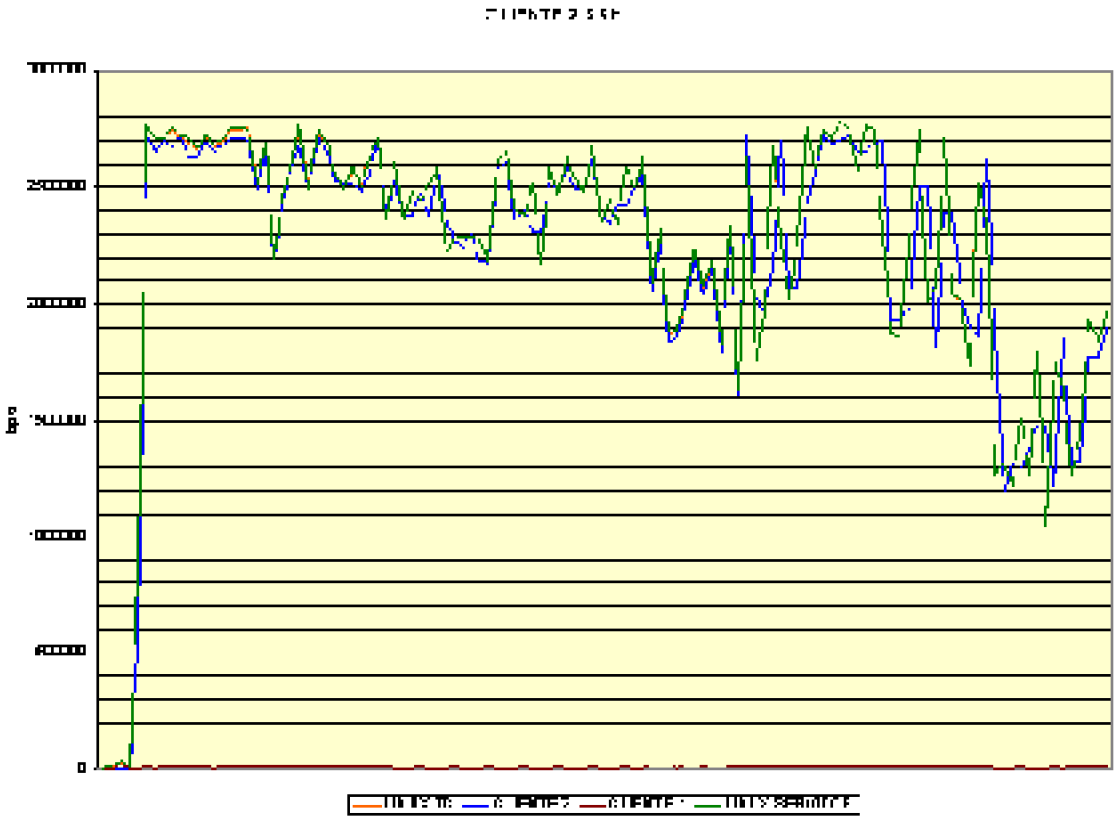
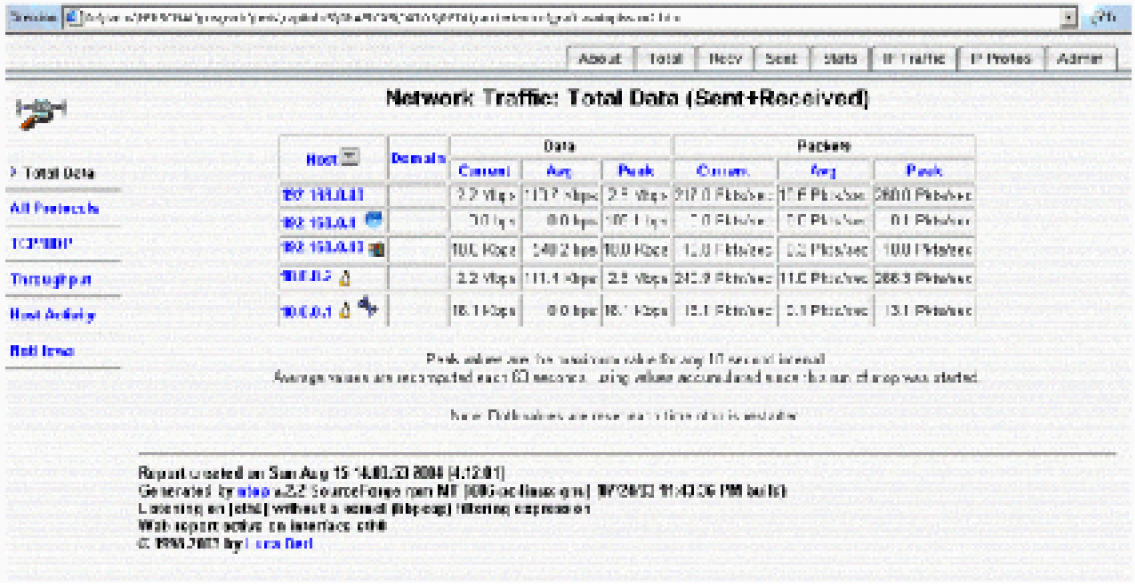


Figura 47. Caracterización tráfico debido al Cliente 2. Protocolo SSH

Datos tomados con NTOP



Datos tomados con WHATSUP

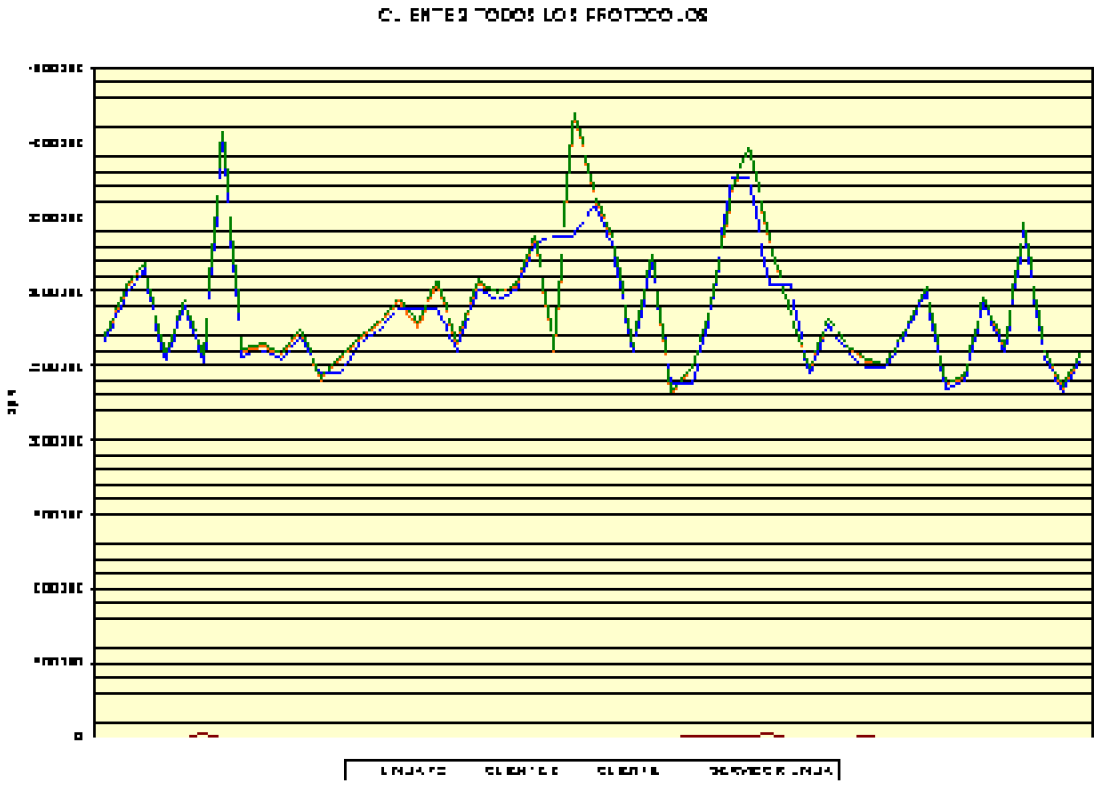
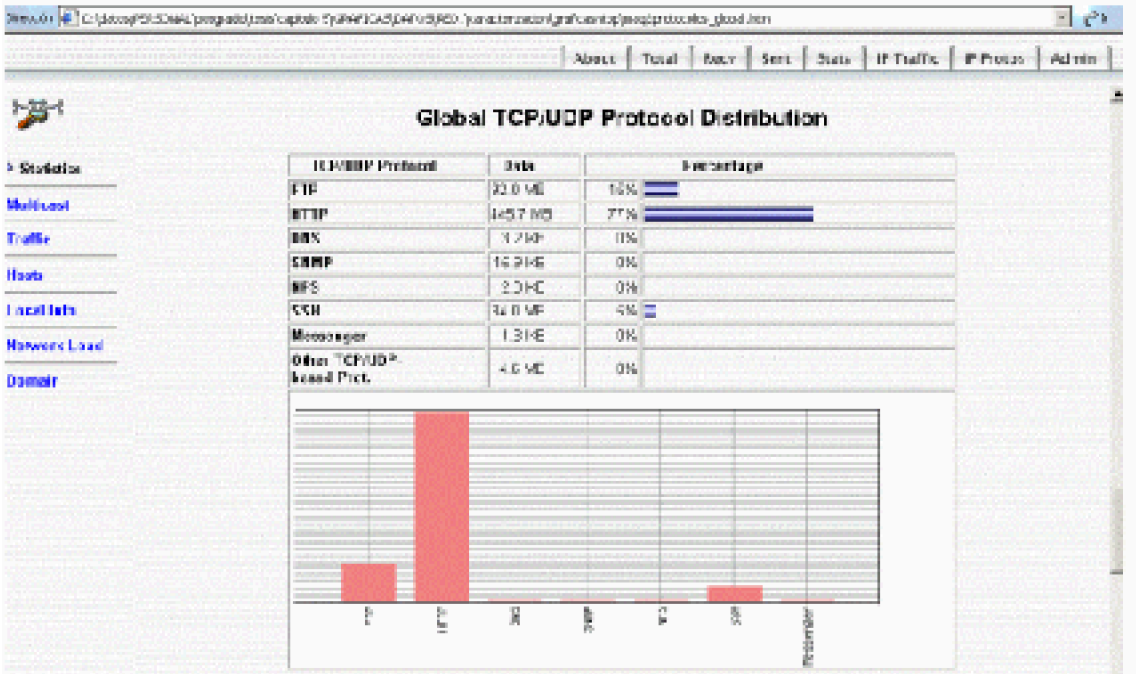


Figura 48. Caracterización tráfico debido al Cliente 1. Protocolos FTP, HTTP y SSH

Datos tomados con NTOP



5.3.3. Datos obtenidos para los dos clientes cursando tráfico

En las figuras 49, 50 y 51 se observa el comportamiento de la red cuando los dos clientes acceden a los servicios de la red, haciendo uso a la vez de cada protocolo

Los resultados obtenidos son:

Protocolo FTP: Ver figura 49. Tráfico promedio por cada cliente:

Cliente 1: 7.45 Mbps

Cliente 2: 2.63 Mbps

Protocolo HTTP: Ver figura 50. Tráfico promedio por cada cliente:

Cliente 1: 9.34 Mbps

Cliente 2: 1.37 Mbps

Protocolo SSH: Ver figura 51. Tráfico promedio por cada cliente:

Cliente 1: 8.66 Mbps

Cliente 2: 1.19 Mbps

Claramente se puede observar que no hay una distribución equitativa del ancho de banda disponible, igualmente ya no se presenta un comportamiento “plano” en el tráfico (incluyendo el cliente 1), esto debido a la disputa por el ancho de banda entre varios clientes

En la figuras 52 y 53, se observa el comportamiento de la red cuando los dos clientes hacen uso de los tres protocolos al tiempo. Este es el estado “normal” de la red, donde se tienen todos los clientes accediendo a los diferentes recursos disponibles en los servidores. De acuerdo a la figura 52, se nota nuevamente distribución no equitativa del

ancho de banda para los dos clientes, el tráfico promedio para cada uno es de 8.19 Mbps para el cliente 1 y de 1.40 Mbps para el Cliente 2, esta distribución hace que para la distribución global del tráfico por protocolo, prime la condición del cliente 1, es decir, se presente una distribución por protocolos con tendencia al equilibrio ver figura 52.

En la figura 53 se observa la distribución por protocolos individual de cada Cliente, nótese que para el Cliente 2 se mantiene la distribución no uniforme entre los tres protocolos.

Datos tomados con WHATSUP

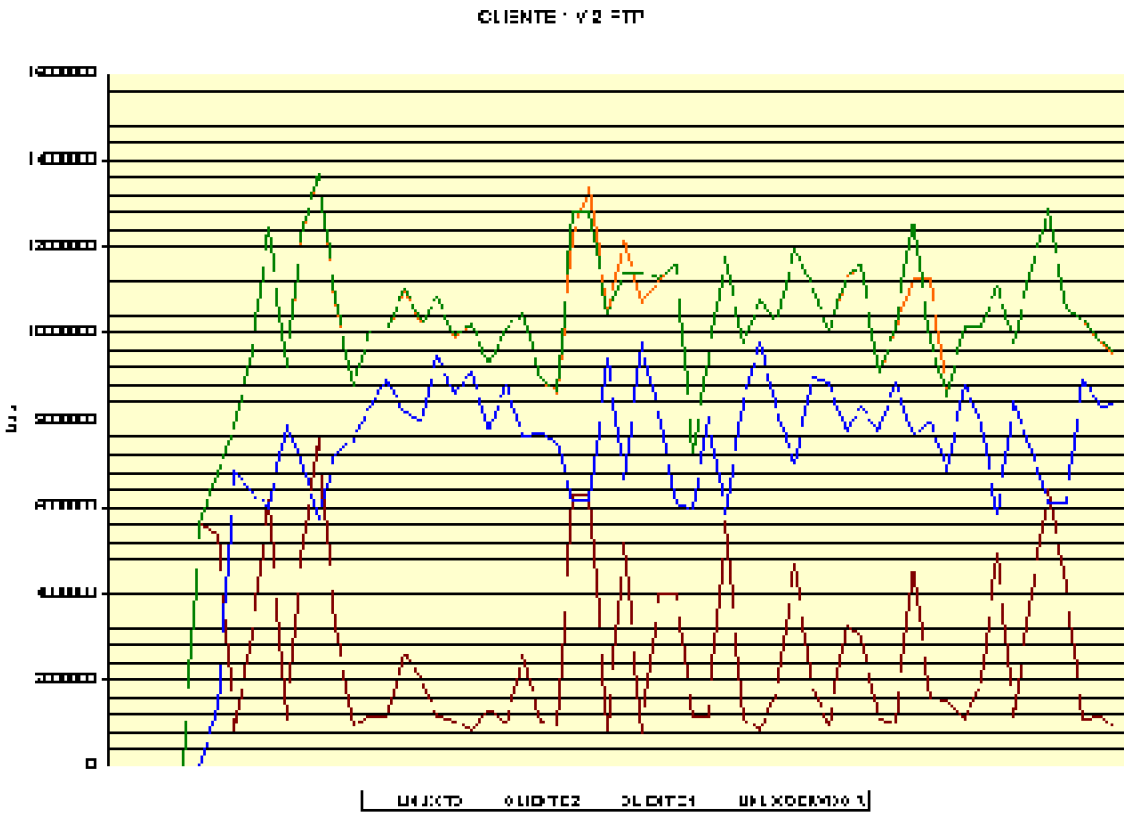
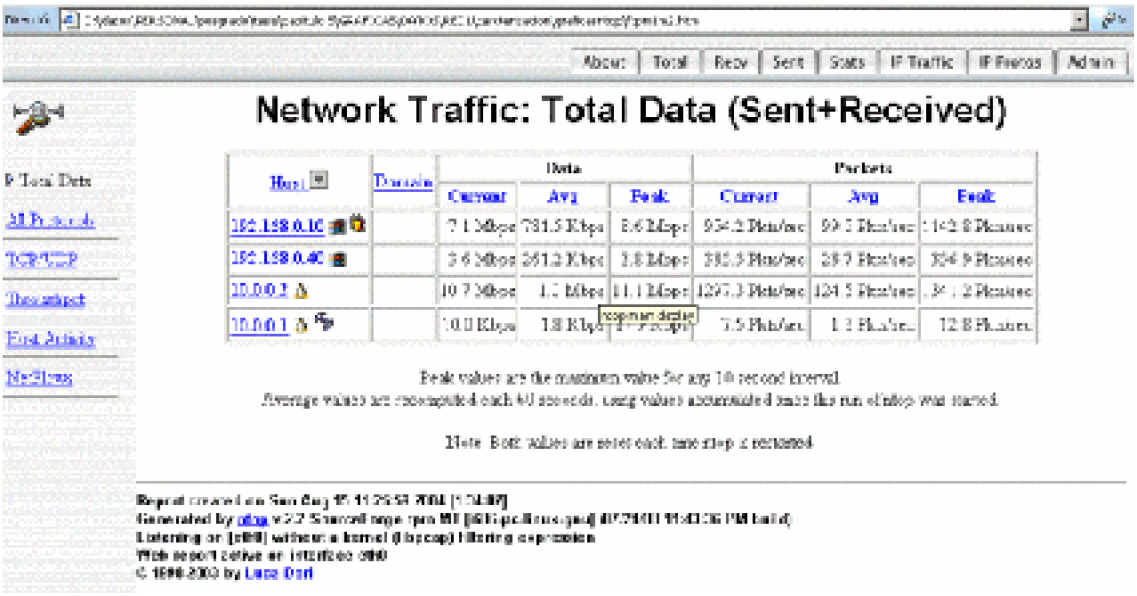


Figura 49. Caracterización tráfico debido a los Clientes 1 y 2. Protocolo FTP

Datos tomados con NTOP



Datos tomados con WHATSUP

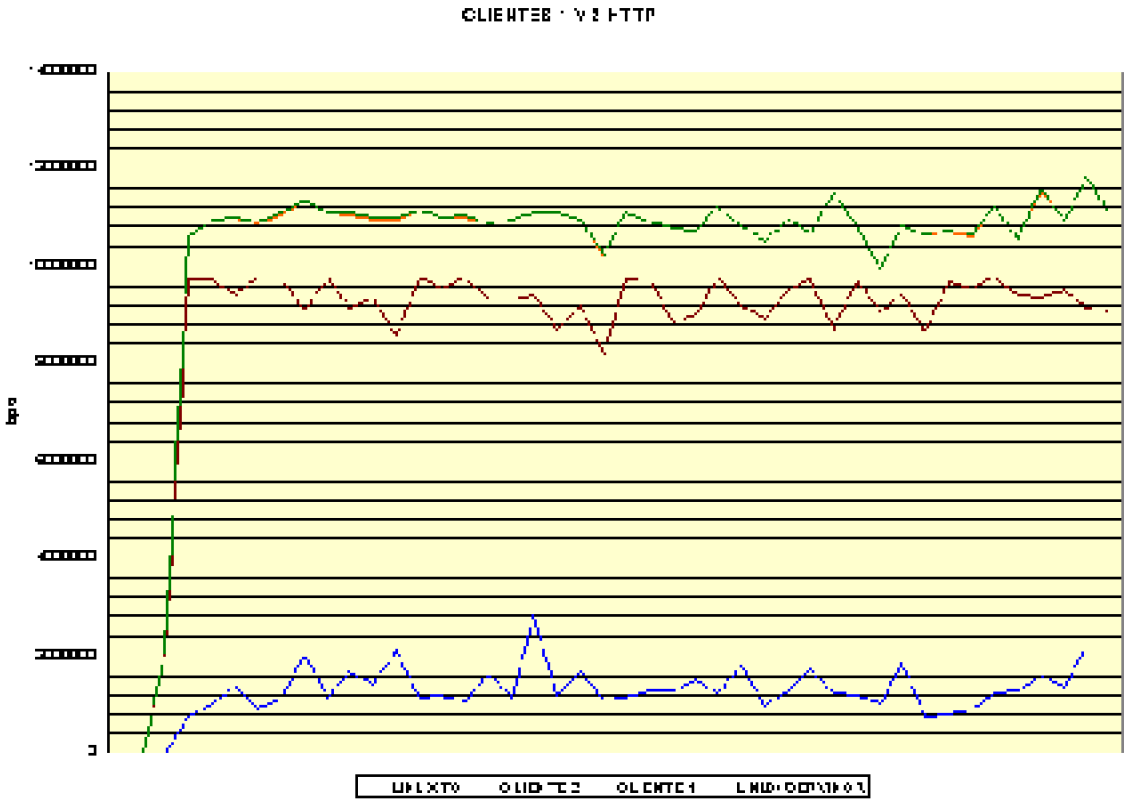
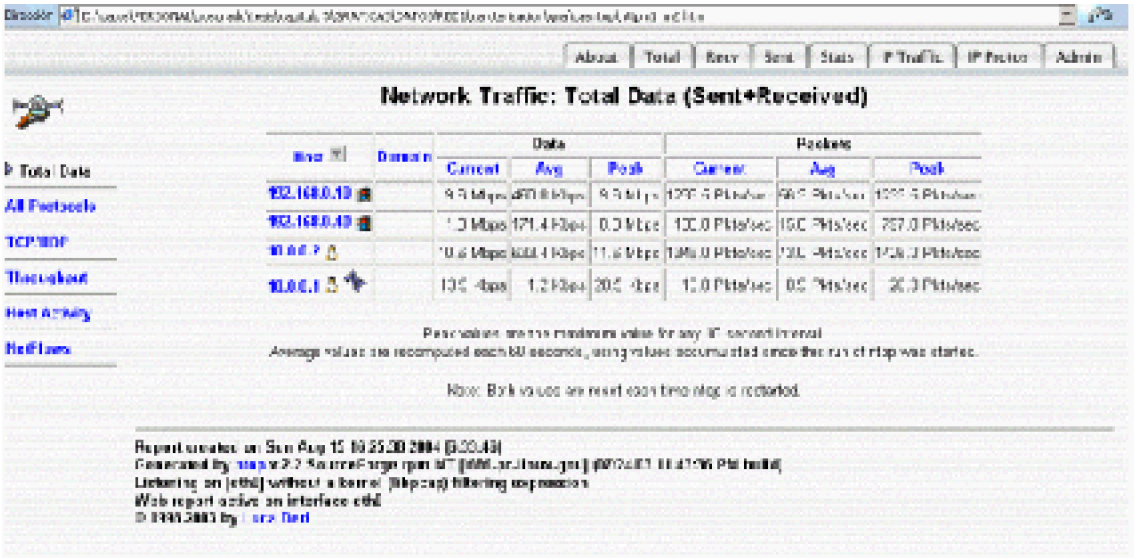


Figura 50. Caracterización tráfico debido a los Clientes 1 y 2. Protocolo http

Datos tomados con NTOP



Datos tomados con WHATSUP

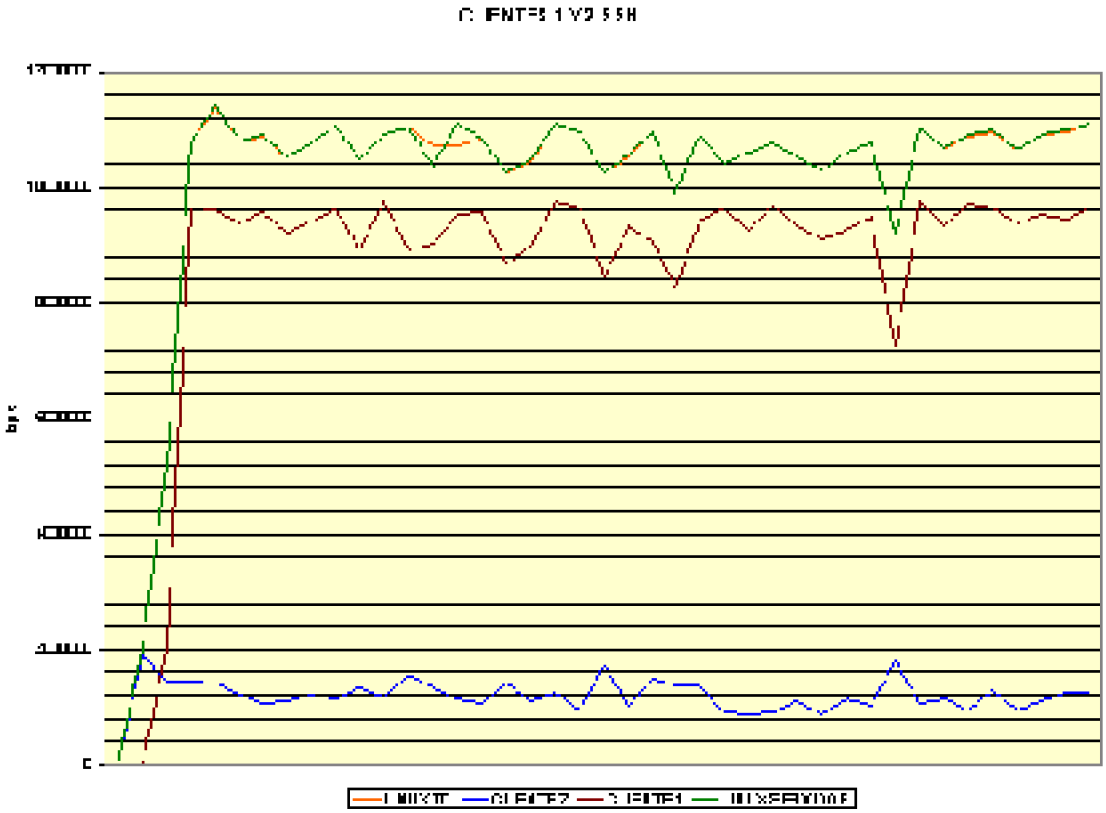
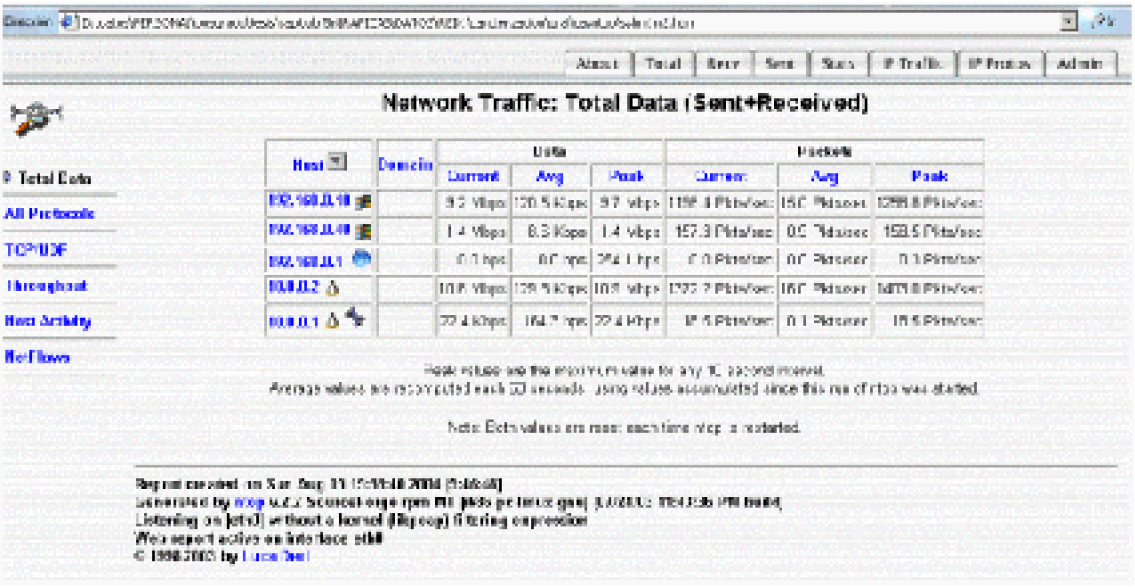


Figura 51. Caracterización tráfico debido a los Clientes 1 y 2. Protocolo SSH

Datos tomados con NTOP



Datos tomados con WHATSUP

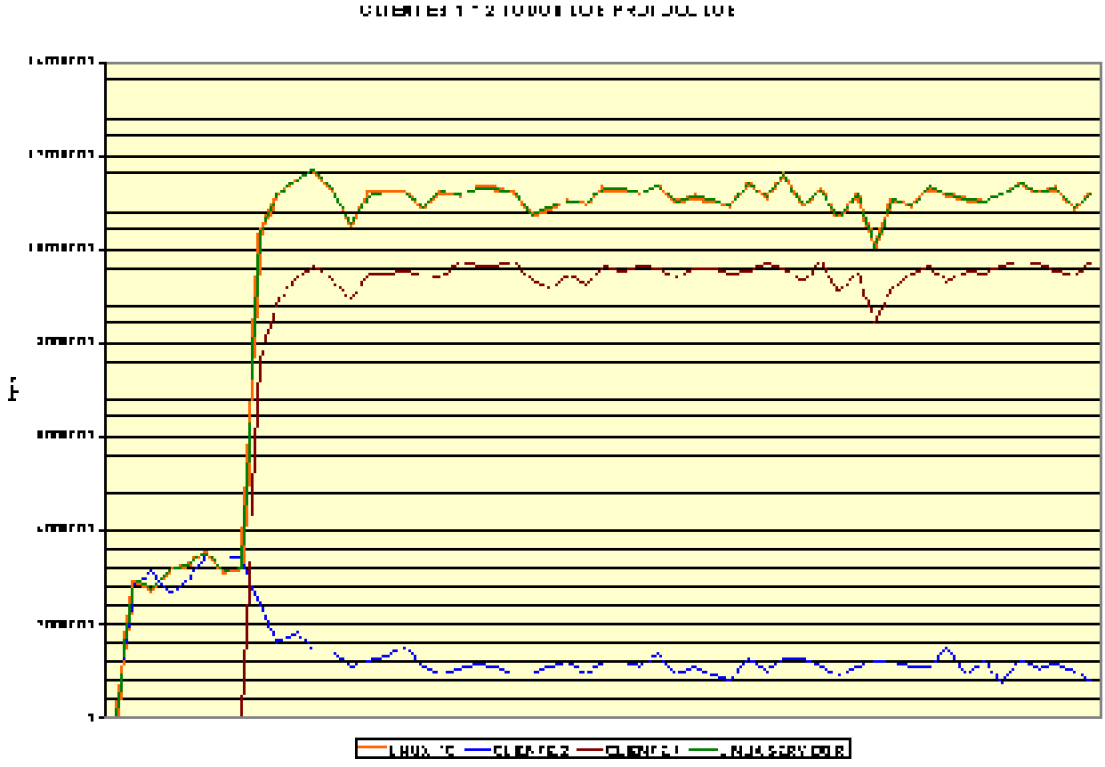


Figura 52. Caracterización tráfico debido a los Clientes 1 y 2. Protocolos FTP, HTTP y SSH

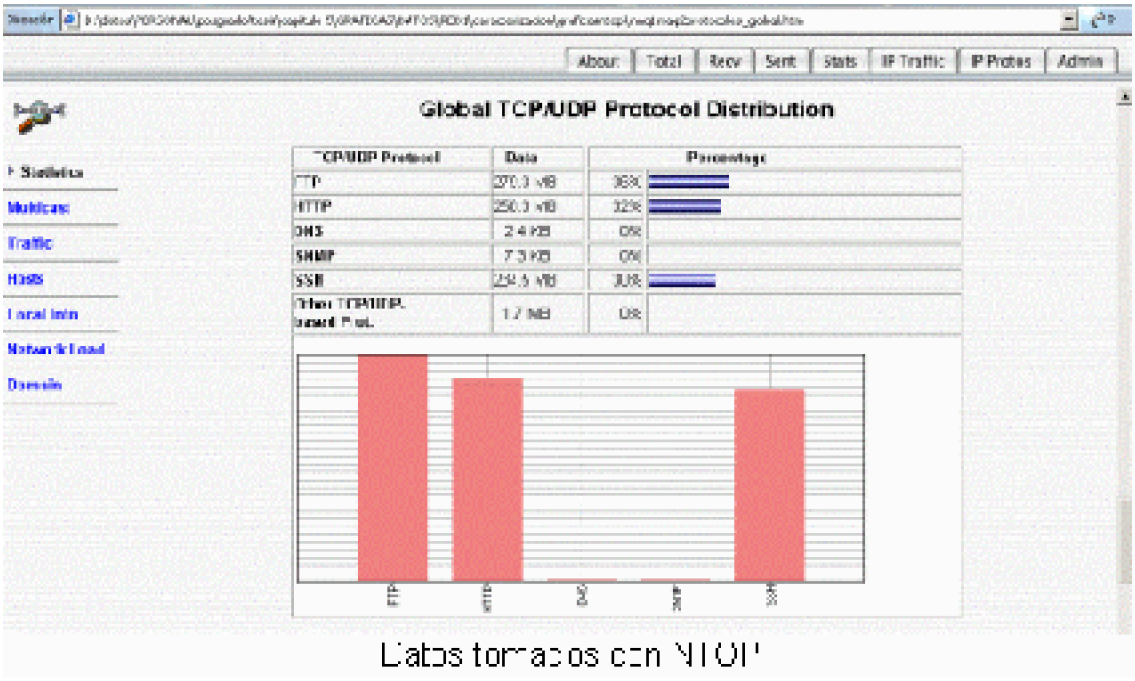
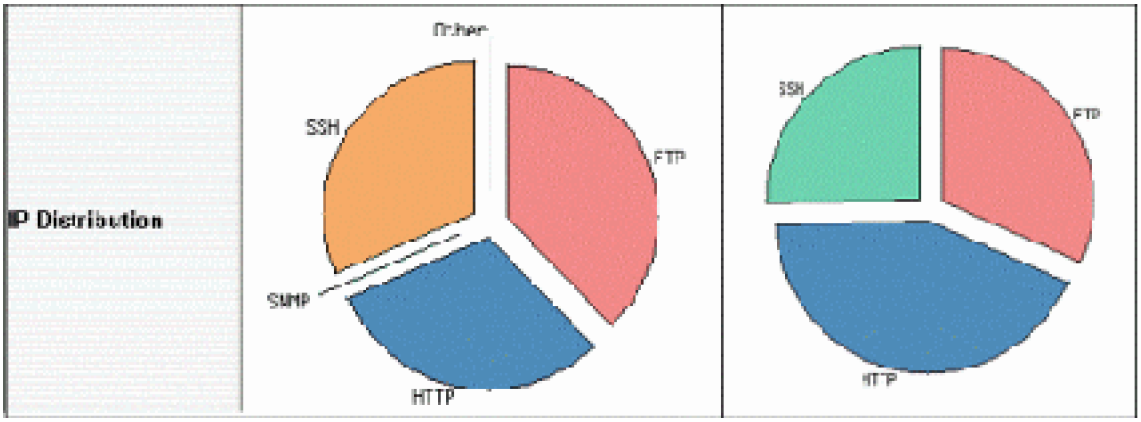


Figura 53. Distribución de protocolos FTP, HTTP y SSH en los clientes 1 y 2

Datos tomados con NTOP



Después del análisis de la red sin ningún tipo de control de tráfico, se puede concluir:

- La capacidad máxima observada para los tres protocolos fue de 9.6 Mbps, esto en la máquina Cliente 1.
- Dependiendo de la máquina que esté haciendo uso de la red se presenta una diferencia entre el ancho de banda disponible y el ancho de banda utilizado. Es así como el Cliente 2 no está en capacidad de aprovechar todo el BW disponible, es mas, su capacidad de uso depende del protocolo que esté siendo usado
- Cuando se presenta competencia entre los dos clientes por el ancho de banda disponible, es clara la tendencia del Cliente 1 a llevarse la mayor parte de este, cursando en promedio el 85% del tráfico.
- Se presentan diferencias en la distribución por protocolos en cada cliente, ya que en el Cliente 1 se hace equitativamente, mientras que en el Cliente 2, el protocolo HTTP

cursa el 77% del tráfico, mientras que el SSH solo cursa el 5%.

Se puede observar en todas las figuras (41 a 53) que el tráfico cursado por el servidor es igual al tráfico cursado por el Enrutador Linux, esto debido a que se hacen requerimientos por parte de los clientes para hacer uso de todo el ancho de banda disponible.

5.4 Control de Tráfico en la red de prueba

En las secciones anteriores se realizó en la red propuesta la caracterización del tráfico para establecer referencias que permitan posteriormente verificar que efectivamente se realiza un control del tráfico en la red de prueba, cuando se implementa una disciplina de cola mediante IPROUTE.

En capítulos anteriores se explicó esta utilidad, se definió de manera general su sintaxis y se introdujeron los conceptos presentes en el tema de encolamiento, a continuación se presentan los diferentes scripts relacionados con las configuraciones definidas para controlar el tráfico en la red de prueba, el control se basa en la disciplina de cola HTB, es decir, los scripts se realizan teniendo en cuenta los parámetros definidos para realizar el control del ancho de banda al emplear este tipo de disciplina de cola, cada script consta de tres partes: la primera define la disciplina de cola raíz en este caso HTB, en la segunda se establecen las diferentes clases y finalmente se implementa el filtro.

5.4.1 Control de Tráfico por Subred

Para este caso se procedió a regular el tráfico basado en la dirección IP de una subred, se emplea el script HTBBASICO, el cual se presenta a continuación.

```
#!/bin/bash
#
# Se realiza borrado de las disciplinas de colas que se
# encuentren configuradas, etapa egress (tráfico saliente)
# e ingress (tráfico entrante) en eth1
#
tc qdisc del dev eth1 root 2> /dev/null > /dev/null
tc qdisc del dev eth1 ingress 2> /dev/null > /dev/null
#
# Se define la disciplina de cola tipo HTB para tráfico saliente
# (egress) en la raíz handle 1:0 para el dispositivo de red eth1
#
```

```
tc qdisc add dev eth1 root handle 1:0 htb
#
# Se agrega la clase hija parent 1:0 tipo HTB identificada como
# classid 1:1 para una rata máxima de 9000kbps
#
tc class add dev eth1 parent 1:0 classid 1:1 htb rate 9000kbit
#
# Se agrega una nueva clase hija identificada como classid 1:2
# esta es pariente de la clase 1:1 definida como parent 1:1 se le
# asigna un rate de 7000kbps con un valor máximo de 7500kbps definido
# por el parámetro ceil.
#
tc class add dev eth1 parent 1:1 classid 1:2 htb rate 7000kbit ceil 7500kbit
#
#
#
# Se le asigna la disciplina de cola tipo pfifo a la clase 1:2
# con un máximo de 10 paquetes en cola, y se define un manejador
# handle 10:
#
tc qdisc add dev eth1 parent 1:2 handle 10: pfifo limit 10
#
# Se asigna un filtro para clase 1:2 definido por el flowid 1:2
# aquí se está controlando el tráfico por subred con el filtro u32
# basado en el protocolo ip con prioridad 1 y toma como referencia
# la dirección destino en este caso la subred 192.168.0/24
#
tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip dst 192.168.0/24
flowid 1:2
#
# Fin del script HTBBASICO
#
```

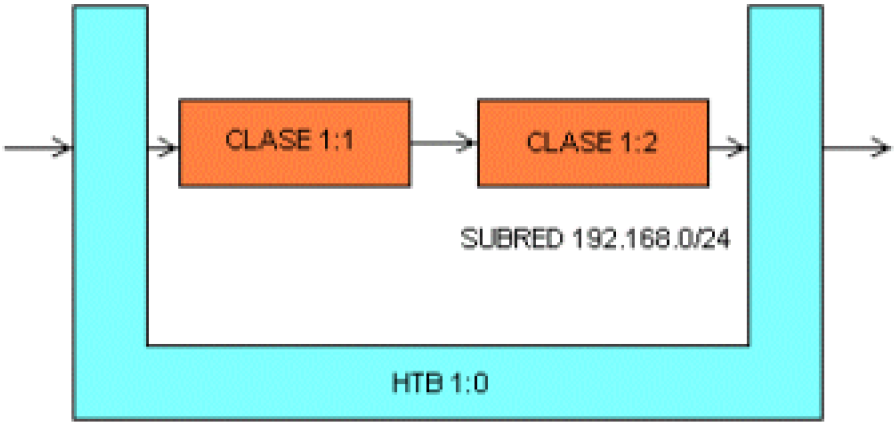


FIGURA 54 Representación gráfica del script HTBBASICO

Mediante la figura 54 se puede apreciar como el tráfico es tratado con la disciplina cola raíz HTB 1:0, y se representan las distintas clases en este caso solo aparecen dos clases, la clase 1:1 a la cual se le define la clase hija 1:2 esta tiene un filtro para controlar el tráfico por dirección de subred, lo importante aquí es la posibilidad de definir clases hijas adicionales para distintas subredes, es decir, pueden existir clases hijas 1:3, 1:4, etc a las cuales de asignan filtros por subred

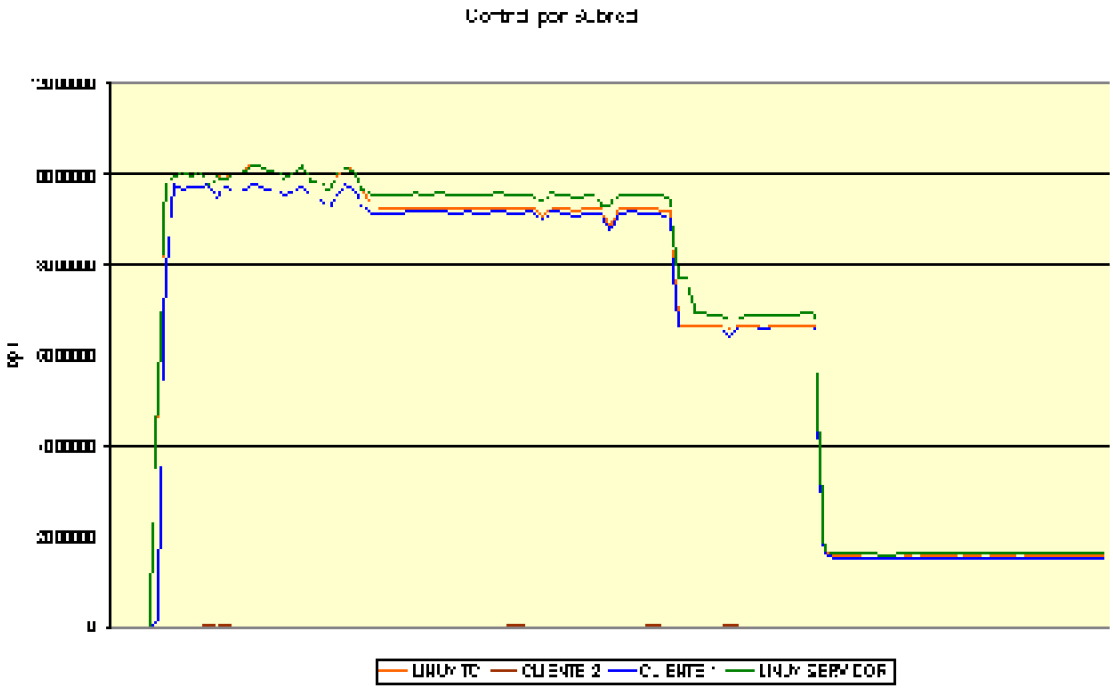


Figura 55 Control de tráfico por sub red desde 9 a 1.5 Mbps

En la figura 55 se presenta el tráfico hacia la subred 192.168.0/24, para valores de ancho de banda desde 9 Mbps a 1.5 Mbps, la cual ilustra como se regula el tráfico empleando el script HTBBASICO. La variación del ancho de banda asignado a la subred, está dado por el parámetro rate del script HTBBASICO

5.4.2 Control de Tráfico por Dirección IP

Para este tipo de control se tiene como punto de partida el script HTBBASICO, al cual se le hacen modificaciones para que, en lugar de controlar el tráfico por subred, lo haga ahora por dirección IP, se requiere entonces definir clases adicionales para lograr el objetivo, es decir, se define la disciplina de cola raíz que para toda esta implementación es la HTB, después se define una clase que permita luego construir clases hijas basadas en direcciones ip, para este caso se requiere de dos clases hijas una por cada dirección ip, pues se cuenta solo con dos clientes, se explica solamente en el script aquello que es nuevo. A continuación se presenta el script correspondiente

```
#!/bin/bash
#
# Este Script controla el ancho de banda asignado a dos máquinas
# diferentes accediendo al servidor
#
# Se realiza borrado de las disciplinas de colas que se encuentren
# configuradas, etapa egress e ingress
#
tc qdisc del dev eth1 root 2> /dev/null > /dev/null
tc qdisc del dev eth1 ingress 2> /dev/null > /dev/null
#
# Se define la disciplina HTB tipo egress para esta disciplina se
# establece una clase por defecto llamada clase 12, la cual maneja
# el otro tráfico que no pasa por filtros
#
tc qdisc add dev eth1 root handle 1: htb default 12
#
# Se define la clase principal para la disciplina HTB, se limita el ancho
# de banda a 9Mbps
# Note que todo el comando está definido en una sola línea
#
tc class add dev eth1 parent 1: classid 1:1 htb rate 9050kbit ceil 9050kbit
#
# Se definen las clases para cada maquina y se asignan los anchos de
# banda garantizado 8Mbps y máximo posible de 9Mbps Cliente 1
```

```
#
tc class add dev eth1 parent 1:1 classid 1:10 htb rate 8000kbit ceil 9000kbit
#
# Se definen las clases para cada maquina y se asignan los anchos de
# banda garantizado 1 Mbps y máximo posible 9 Mbps Cliente 2
#
tc class add dev eth1 parent 1:1 classid 1:11 htb rate 1000kbit ceil 9000kbit
#
# Se establece una clase por defecto para el resto de tráfico que no vaya
# para los clientes 1 y 2
#
tc class add dev eth1 parent 1:1 classid 1:12 htb rate 50kbit ceil 9000kbit
#
# Se asigna a cada una de las clases 10, 11 y 12, colas tipo sfq que son # del tipo de colas Justas con tiempo de reconfiguración cada 10
# segundos
#
tc qdisc add dev eth1 parent 1:10 handle 10: sfq perturb 10
tc qdisc add dev eth1 parent 1:11 handle 11: sfq perturb 10
tc qdisc add dev eth1 parent 1:12 handle 12: sfq perturb 10
#
# Se definen los filtros que permiten la escogencia de las maquinas a las # que se les controlara el ancho de banda, se trabaja nuevamente el
# filtro u32 pero esta vez sobre la dirección ip de las maquinas, para el
# Cliente 1 se le asigna la clase 10, Cliente 2 se le asigna la clase 11
# y la clase 12 para el del resto del tráfico hacia la subred
# 192.168.0/24 en caso de que hubiesen otros clientes este seria su
# filtro y el ancho de banda será de 50 Kbps, garantizado
#
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.10 flowid 1:10
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.40 flowid 1:11
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0/24 flowid 1:12
```

#

Fin del script

#

Con la figura 56 se puede resumir lo que se hace el script para el control de ancho de banda por dirección IP, se aprecia la definición de dos clases especificas una por cada cliente y una tercera asignada al tráfico restante en la red, se debe tener en cuenta que cada clase final tiene asociado un filtro que para el caso es el filtro u32, aquí las clases finales son 1:10, 1:11 y 1:12

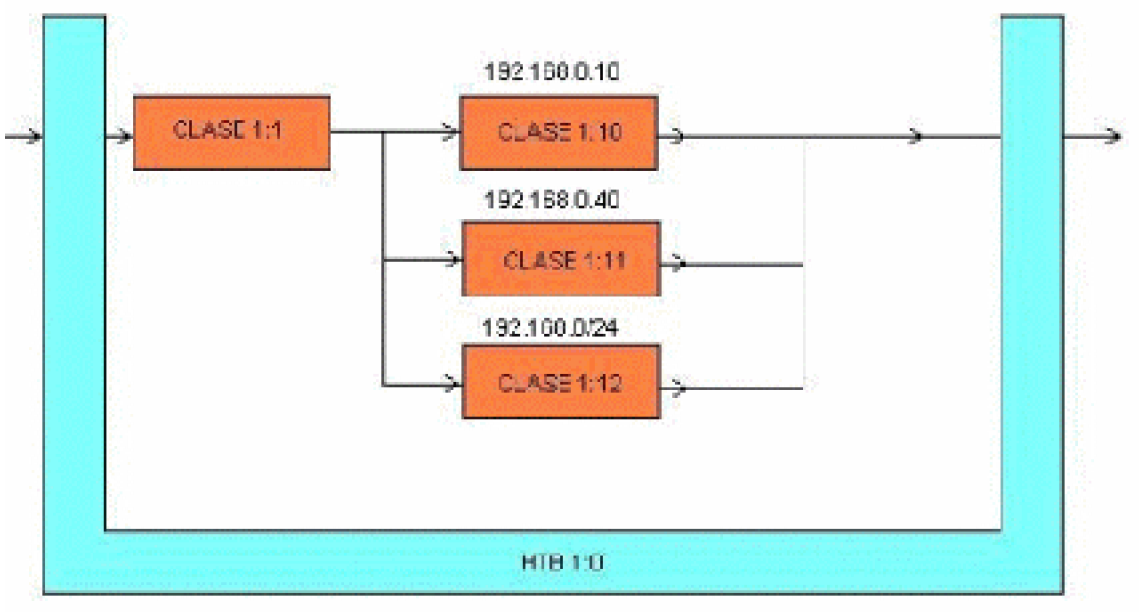


Figura 56 Representación gráfica del script para control de tráfico por dirección IP

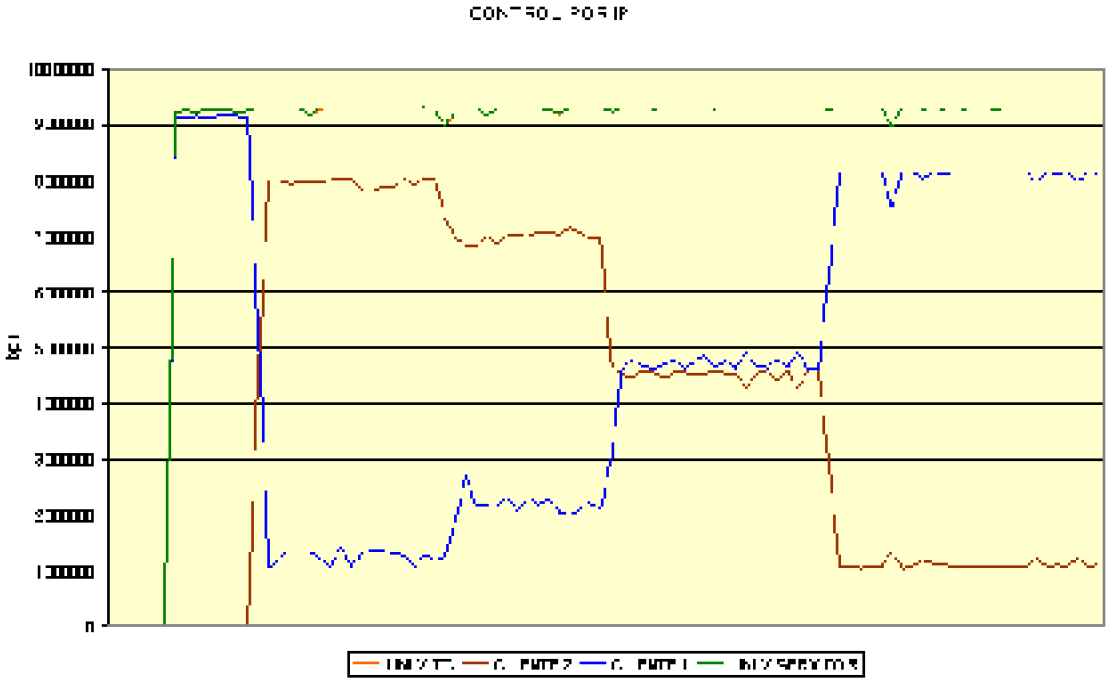


Figura 57 Control de tráfico por dirección IP Clientes 1 y 2

La figura 57 permite ver como se controla el ancho de banda para cada cliente, esto significa cambiar en el script el valor de rate. Para el script presentado, al Cliente 1 se le asignó un ancho de banda de 8Mbps, mientras el Cliente 2 cuenta con un ancho de banda de 1 Mbps, en el caso de solo existir un Cliente en la red este puede hacer uso de todo el ancho de banda disponible hasta un máximo de 9 Mbps, definido por el parámetro ceil.

5.4.3 Control de Tráfico por protocolos

En esta sección se presenta y describe el script empleado para realizar el control del ancho de banda basado en el protocolo empleado por cada cliente, en este caso se utilizan los protocolos HTTP, SSH y FTP, definiéndose un porcentaje para cada uno según el ancho de banda asignado a cada cliente, lo cual ocurre cuando el cliente está haciendo uso de todos los protocolos a la vez.

En la tabla 4 se aprecia la distribución porcentual de tráfico:

	CLIENTE 1	CLIENTE 2	TOTAL
	80%	20%	
FTP	15%	50%	22.00%
HTTP	60%	20%	52.00%
SSH	25%	30%	26.00%

Tabla 4 Distribución porcentual por protocolos del script para control de tráfico

Del total de ancho de banda disponible, se asigna al Cliente1 el 80% y el restante 20% al Cliente 2. A su vez se hace subdivisión en cada cliente, esto de acuerdo a la tabla 4. En la columna TOTAL de la tabla 4 se observa cual es el porcentaje esperado de distribución global de protocolos en la red, cuando se ejecuta el script de control de tráfico por protocolos, el cual se detalla a continuación

```
#!/bin/bash
# Este Script controla el ancho de banda asignado a dos máquinas
# diferentes accediendo al servidor
# Se realiza borrado de las disciplinas de colas que se encuentren
# configuradas, etapa egress e ingress
tc qdisc del dev eth1 root 2> /dev/null > /dev/null
tc qdisc del dev eth1 ingress 2> /dev/null > /dev/null
# Se define la disciplina HTB tipo egress
tc qdisc add dev eth1 root handle 1: htb default 41
# Se define la clase principal para la disciplina HTB, se limita el ancho
# de banda a 5 Mbps
```

```
tc class add dev eth1 parent 1: classid 1:1 htb rate 5100kbit ceil 5500kbit
# Se define una clase por cada Cliente
# Cliente 1
tc class add dev eth1 parent 1:1 classid 1:2 htb rate 4050kbit ceil 5000kbit
# Cliente 2
tc class add dev eth1 parent 1:1 classid 1:3 htb rate 1050kbit ceil 5000kbit
# Otros Clientes
tc class add dev eth1 parent 1:1 classid 1:4 htb rate 100kbit ceil 5000kbit
# HTTP Cliente 1
tc class add dev eth1 parent 1:2 classid 1:21 htb rate 2400kbit ceil
5000kbit
#SSH Cliente 1
tc class add dev eth1 parent 1:2 classid 1:22 htb rate 1000kbit ceil 5000kbit
#FTP Cliente 1
tc class add dev eth1 parent 1:2 classid 1:23 htb rate 600kbit ceil 5000kbit
# Otro tráfico Cliente 1
tc class add dev eth1 parent 1:2 classid 1:24 htb rate 50kbit ceil 5000kbit
# HTTP Cliente 2
tc class add dev eth1 parent 1:3 classid 1:31 htb rate 200kbit ceil 5000kbit
# SSH Cliente 2
tc class add dev eth1 parent 1:3 classid 1:32 htb rate 300kbit ceil 5000kbit
# FTP Cliente 2
tc class add dev eth1 parent 1:3 classid 1:33 htb rate 500kbit ceil 5000kbit
# Otro tráfico Cliente 2
tc class add dev eth1 parent 1:3 classid 1:34 htb rate 50kbit ceil 5000kbit
# Disciplinas de colas para cada clase final
tc qdisc add dev eth1 parent 1:21 handle 21: sfq perturb 10
tc qdisc add dev eth1 parent 1:22 handle 22: sfq perturb 10
tc qdisc add dev eth1 parent 1:23 handle 23: sfq perturb 10
tc qdisc add dev eth1 parent 1:24 handle 24: sfq perturb 10
tc qdisc add dev eth1 parent 1:31 handle 31: sfq perturb 10
tc qdisc add dev eth1 parent 1:32 handle 32: sfq perturb 10
tc qdisc add dev eth1 parent 1:33 handle 33: sfq perturb 10
tc qdisc add dev eth1 parent 1:34 handle 34: sfq perturb 10
```

```
tc qdisc add dev eth1 parent 1:4 handle 41: sfq perturb 10
# Se definen los filtros que permiten la escogencia de los Clientes a los
# que se les controlará el ancho de banda
#
# Filtros para las Clases asociadas al Cliente 1
#
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.10 match ip sport 80 0Xfff flowid 1:21
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.10 match ip sport 22 0Xfff flowid 1:22
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.10 match ip sport 20 0Xfff flowid 1:23
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.10 match ip sport 21 0Xfff flowid 1:23
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.10 flowid 1:24
#
# Filtros para las Clases asociadas al Cliente 1
#
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.40 match ip sport 80 0Xfff flowid 1:31
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.40 match ip sport 22 0Xfff flowid 1:32
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.40 match ip sport 20 0Xfff flowid 1:33
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.40 match ip sport 21 0Xfff flowid 1:33
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.10 flowid 1:34
#
# Filtros para las Clases asociadas al Resto de los Clientes en la Red
#
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.10 flowid 1:41
#
# Fin del Script
```

#

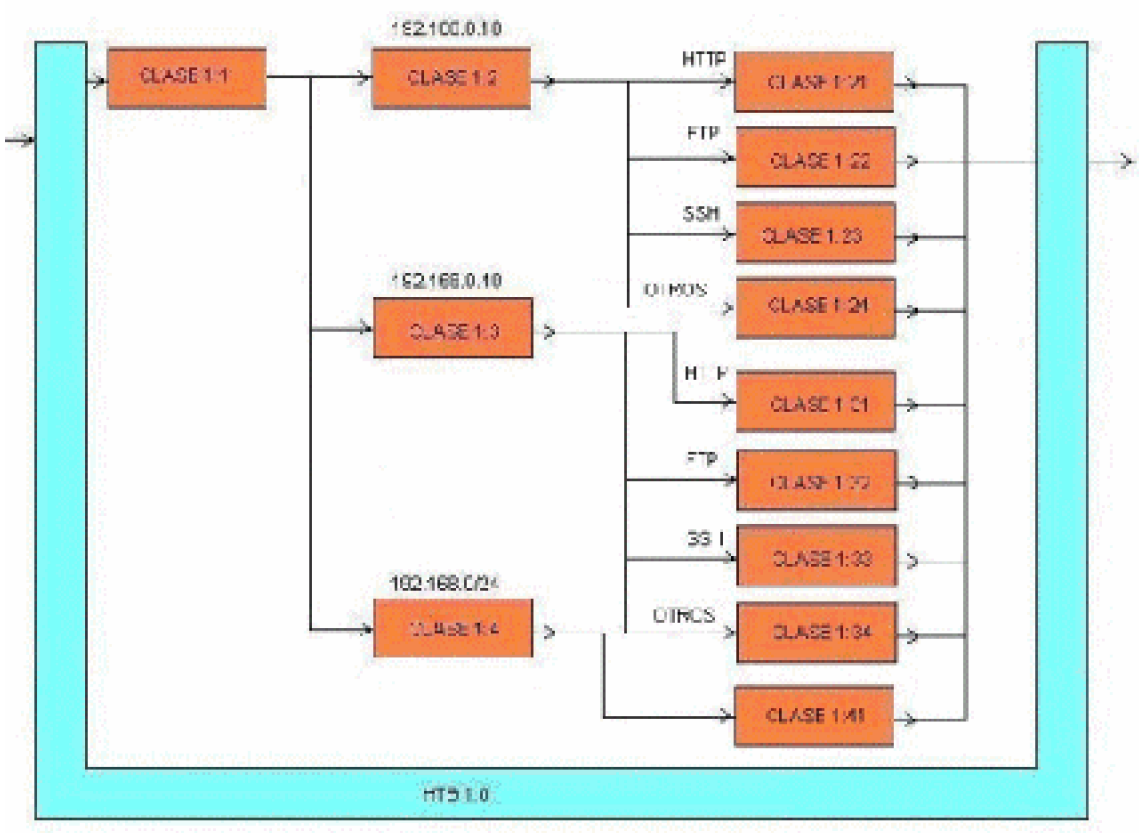


Figura 58 Representación Gráfica del Script Para Control de Ancho de Banda a Nivel de Protocolos.

De la figura 58 se aprecia gráficamente como se relacionan las diferentes clases en el proceso de control de ancho de banda por protocolos y permite entender de manera grafica como funciona el script planteado para dicho control, se debe tener en cuenta que a cada clase final existe un filtro asociado, que para el caso es el filtro u32 definido con base a los protocolos empleados en un cliente especifico.

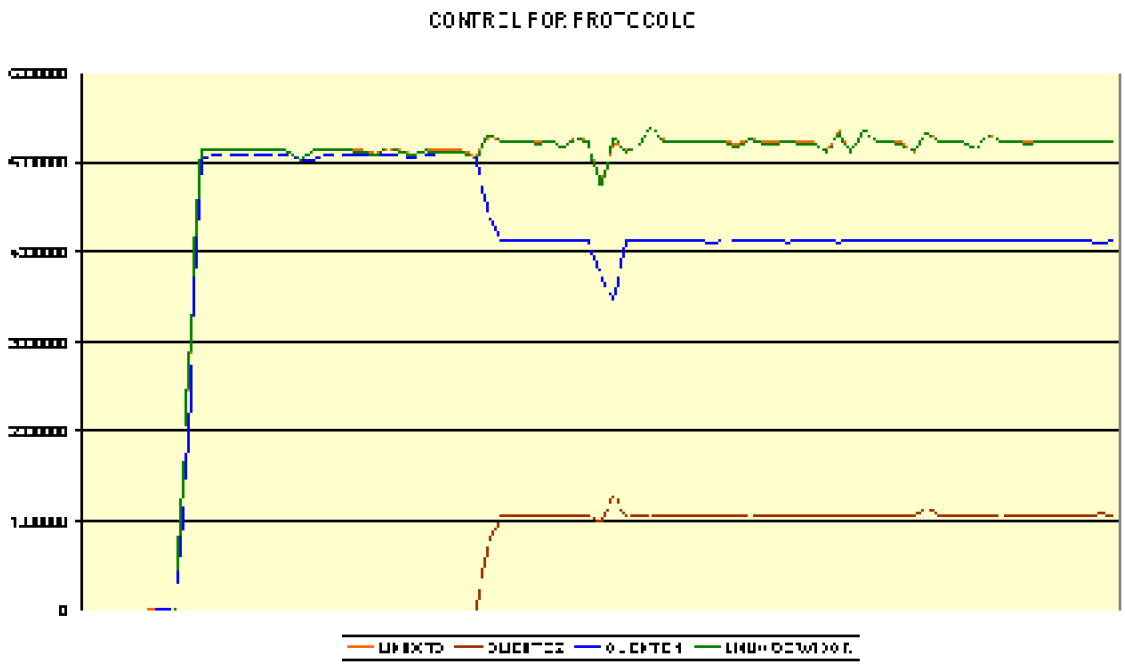


Figura 59 Control de tráfico por protocolo clientes 1 y 2.

La figura 59 permite apreciar el comportamiento del tráfico hacia los Clientes 1 y 2, en donde al revisar el script para el control del ancho de banda, al cliente 1 se le asigna un ancho de banda de 4 Mbps, con un máximo posible de 5 Mbps, de igual manera al Cliente 2 tiene un ancho de banda de 1 Mbps, con la posibilidad de llegar a 5 Mbps, para poder llegar a este máximo se requiere que no existan otros clientes en la red con los cuales se deba compartir el ancho de banda.

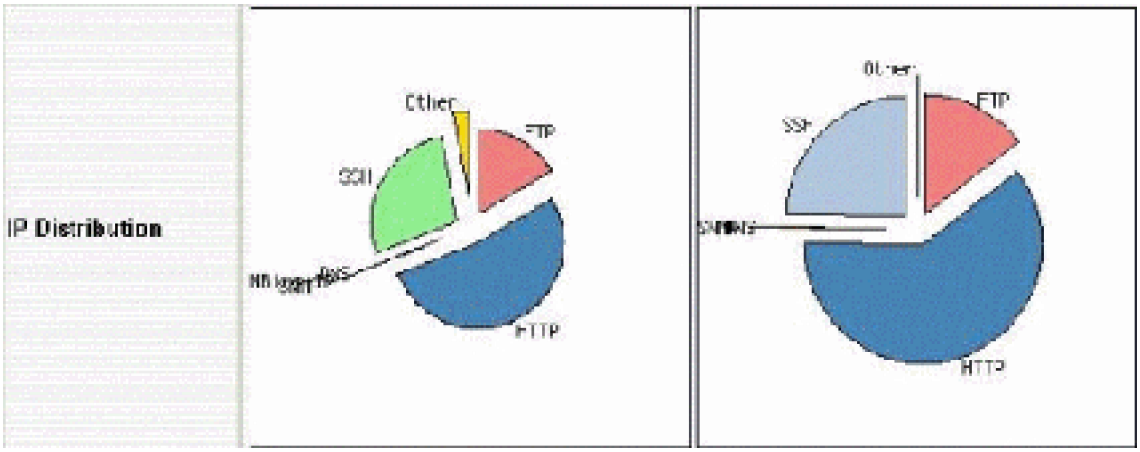


Figura 60. Distribución del tráfico saliente y entrante en el Cliente 1.

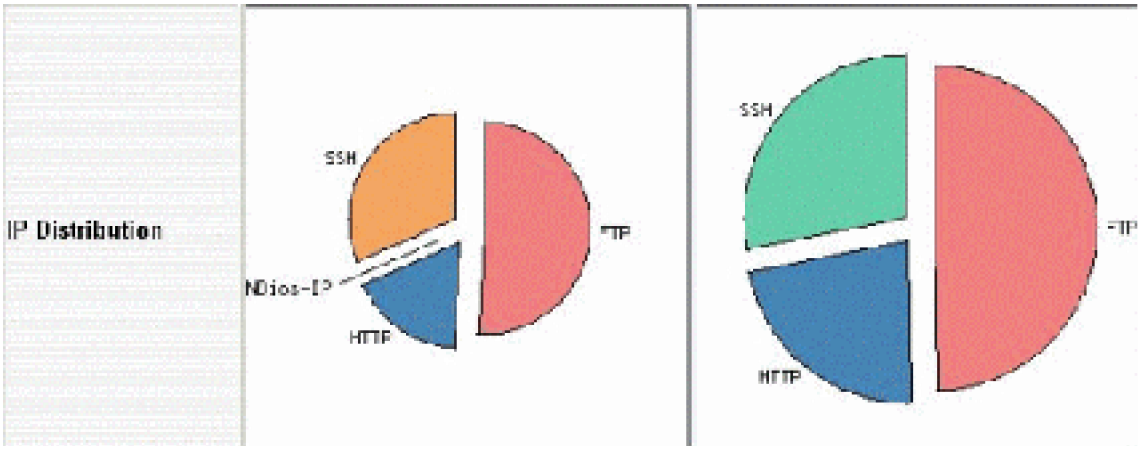


Figura 61 Distribución del Tráfico Saliente y Entrante Cliente 2

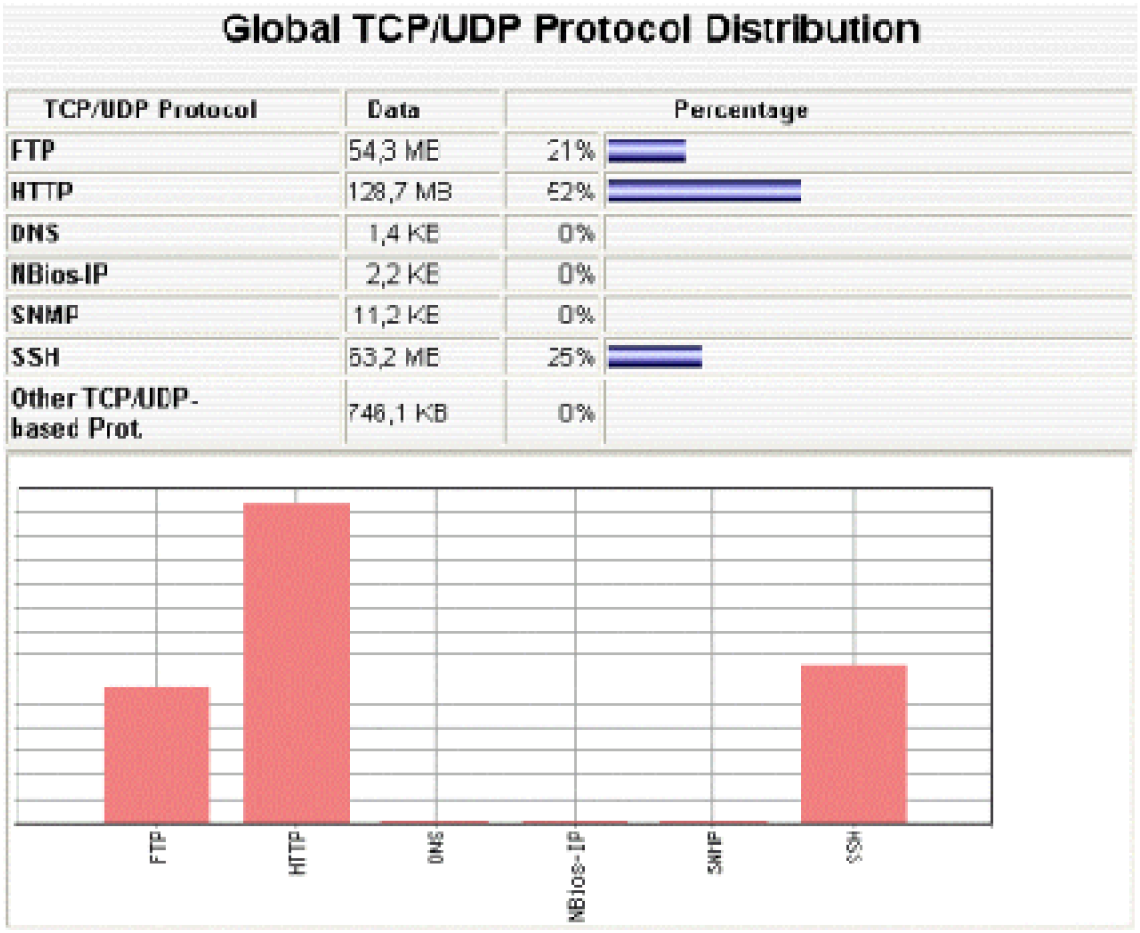


Figura 62 Distribución del tráfico global por protocolos de los Clientes 1 y 2

Las figuras 60 y 61 obtenidas por medio de la herramienta NTOP muestran la distribución por protocolos en cada cliente, al revisar dicha distribución del ancho de banda, en cada cliente este es dividido en porcentajes, se tiene que para el Cliente 1 el ancho de banda asignado es de 4 Mbps, con una distribución aproximada de HTTP 60 %, SSH 25% y FTP 15%, para el caso del Cliente 2 se le asigna un ancho de banda de 1 Mbps, con una distribución aproximada de HTTP 20%, SSH 30% y FTP 50%, es claro

que se ha definido una clase en cada cliente, con ancho de banda mínimo para el manejo de tráfico que corresponda a otro tipo de protocolos, la figura 62 resume el tráfico global de los clientes 1 y 2 en donde se aprecia que la distribución aproximada de los protocolos es de HTTP 52%, SSH 25% y FTP 21%. Obsérvese que los resultados obtenidos se ajustan a los esperados (ver columna TOTAL en la tabla 4)

CONCLUSIONES

Los conceptos básicos de teoría de colas vistos, permiten tener una base para la comprensión de los mecanismos involucrados en los procesos de envío y recepción de datos en los dispositivos de una red de comunicaciones. La posibilidad que se tenga de administrar las colas en los diferentes dispositivos de una red, puede afectar el rendimiento de la misma, de manera que dicho rendimiento, no solo dependerá de factores como Hardware, Ancho de Banda disponible y cantidad de tráfico cursado, sino también de la correcta gestión de colas en la red.

Las versiones del kernel de Linux de la 2.4 en adelante cuentan con herramientas que permiten implementar tareas basadas en la gestión de colas. Dichas herramientas permiten definir clases, políticas y filtros que pueden usarse para convertir una máquina de gama media en un enrutador que podrá realizar tareas de control de tráfico, dignas de cualquier enrutador comercial, tales como:

- Posibilidad de manejar variedad de algoritmos de encolamiento
- Priorizar tráfico.
- Distribución equitativa de ancho de banda
- Reserva de ancho de banda (QoS)

Mediante el montaje práctico realizado, se verificó la eficacia de Linux como dispositivo de control de tráfico, al poderse realizar tareas como:

- Restricción del Ancho de Banda
- Reserva de Ancho de Banda
- Distribución de Ancho de Banda

Cada una de estas tareas se implementó utilizando la disciplina de colas HTB, en conjunto con el filtro U32, lo cual permitió que dichas tareas se aplicaran a nivel de redes, subredes, direcciones IP y protocolos.

Los scripts presentados para control de tráfico pueden ser implementados tan flexiblemente como se quiera: pueden usarse variables para facilitar el cambio de parámetros y automatizar su ejecución.

Es de anotar que Linux posee mas disciplinas de colas y otros tipos de filtros. Un estudio más profundo podrá determinar cuales son los más adecuados de implementar de acuerdo a la necesidad específica.

GLOSARIO

ANCHO DE BANDA: término aplicado usado para describir la capacidad promedio de un medio, protocolo, o conexión dados. Efectivamente describe el "tamaño del tubo" requerido para la aplicación a comunicar sobre la red.

CLASIFICACIÓN: la clasificación es el mecanismo mediante el cual los paquetes son separados para diferente tratamiento, posiblemente en diferentes colas de salida. Durante el proceso de aceptar, enrutar y transmitir un paquete, un dispositivo de red puede clasificar el paquete de un número de maneras diferentes. La clasificación puede incluir el marcado del paquete, lo cual usualmente pasa en los límites de una red que esté bajo un solo control administrativo o la clasificación puede ocurrir en cadasaltoindividualmente.

El modelo Linux permite que un paquete pase por una serie de clasificadores en una estructura de control de tráfico y que sea clasificado en conjunto con los policers.

COLA: es una locación (o Buffer) que contiene un número finito de ítems esperando por una acción o servicio. En redes de datos, una cola es el lugar donde los paquetes (los ítems) esperan para ser transmitidos por el hardware (el servicio).

Sin ningún otro mecanismo, una cola solo ofrece dos acciones interesantes: encolar y desencolar. Una cola cobra importancia cuando se acopla con otros mecanismos los cuales pueden retrasar, reordenar, descartar y priorizar paquetes en múltiples colas.

CONMUTACIÓN DE CIRCUITOS: técnica que permite establecer físicamente una ruta de información (circuito) entre una estación que llama y una estación que recibe,

hasta que la conexión sea liberada.

CONMUTACIÓN DE PAQUETES: método para transmisión de datos en la red, consiste en enviar por bloques los datos a la estación remota. La información a ser enviada se subdivide en paquetes individuales de datos, asignándosele a cada uno una identificación propia y la dirección destino. Cada paquete ocupa el canal de transmisión solo por un instante, quedando este disponible para transmitir el siguiente paquete. Por estar identificados, cada paquete puede viajar por una ruta diferente y pueden llegar a su destino en orden distinto al del punto de transmisión.

CoS: denominación genérica que permite clasificar los tipos de servicios que se entregan en una red de datos.

FILTRO: es el componente más complejo en el sistema de control de tráfico de Linux. Provee un mecanismo conveniente para unir varios de los elementos claves del control de tráfico. El rol mas simple y obvio de un filtro es el de clasificar los paquetes. Los filtros de Linux le permiten al usuario clasificar paquetes en una cola de salida. El filtro puede contener un clasificador y un policer y puede ser aplicado ya sea a colas sin clases o con clases, sin embargo el paquete encolado siempre entra primero a la cola root, después de esto el paquete puede ser direccionado a cualquier subclase (la cual tiene sus propios filtros) donde el paquete puede sufrir clasificaciones subsiguientes.

FLOW (FLUJO) un flujo es una conversación o conexión entre dos hosts. Cualquier grupo único de paquetes ente dos hosts puede ser reconocido como un flujo. Bajo TCP, el concepto de una conexión con una direcciones IP y puertos fuente y destino, representa un flujo. Un flujo UDP puede ser definido similarmente.

Los mecanismos de control de tráfico frecuentemente separan el tráfico en clases o flujos, los cuales pueden ser agregados y transmitidos como un flujo agregado (DiffServ). Mecanismos alternos pueden tratar de dividir el ancho de banda equitativamente, basados en los flujos individuales.

FTP: protocolo definido en el ámbito de Internet para permitir la transferencia de archivos entre terminales.

HANDLE (MANEJADOR): es el identificador único dentro de la estructura de control de tráfico que cada clase y disciplina de colas sin clases requiere. El Handle está constituido por dos miembros: un número mayor y un número menor. Estos números pueden ser asignados arbitrariamente por el usuario de acuerdo a las siguientesreglas:

Major: Este parámetro es completamente libre de significado para el Kernel de Linux. El usuario puede usar un esquema de numeración arbitrario, sin embargo todos los objetos en la estructura de control de tráfico con el mismo padre, deben compartir el mismo número mayor. Esquemas convencionales de numeración empiezan en 1 para objetos relacionados directamente a la qdisc root.

Menor: Este parámetro identifica el objeto como una qdisc si es igual a 0. Cualquier otro valor identifica el objeto como una clase. Todas las clases que comparten un padre, deben tener números menores únicos.

El manejador es usado como el objetivo en frases de identificador de clases o flujos en las sentencias de control de tráfico. Estos manejadores son identificadores externos

para los objetos y se usan en las aplicaciones de usuario. El kernel mantiene identificadores internos para cada objeto

HTTP: protocolo de transferencia utilizado para el servicio Web (www) en Internet. El lenguaje de escritura es el HTML.

HTB: es una disciplina de colas escrita por Martin Devera. Conceptualmente HTB es un número arbitrario de tokens y buckets organizados de forma jerárquica. La qdisc primaria de salida de cualquier dispositivo se denomina root qdisc.

La qdisc root contendrá una clase. Esta simple clase HTB tiene dos parámetros: una rata y un ceil, estos valores deben ser los mismos para la clase de más alto nivel y representan el ancho de banda total disponible de un enlace.

HTB implementa un mecanismo de encolamiento sin clases para el sistema de control de tráfico de linux., permitiéndole al usuario controlar el ancho de banda absoluto a una clase o tráfico particular así como también indica la rata de distribución de l ancho de banda extra cuando está disponible.

IPTABLES: es el paquete de NAT/firewall que por defecto instala Red Hat Linux, surgió para cubrir limitaciones del anterior (ipchains) siendo la principal, que corría como un programa separado y no como parte del kernel. Se considera Iptables mas rápido y seguro, ya que todos los paquetes inspeccionados por iptables pasan a través de una secuencia de colas para su procesamiento. Cada una de las colas está dedicada a un tipo particular de paquete y es controlada por una cadena asociada transformación/filtrado de paquetes.

PAQUETE: representa unidades de capa 3 que incluyen principalmente 3 elementos: información de control (destinatario, remitente y longitud del paquete), los datos que se han de transmitir y los bits destinados al código de detección y corrección de errores. Los datos, los elementos de control y la información de control de error se organizan en un formato específico que depende del protocolo usado.

POLICING: permite controlar la máxima rata de tráfico enviada o recibida en una interfaz y particionar la red en múltiples niveles o clases de servicios.

Un policer es una pregunta si/no acerca de la rata a la cual el tráfico entra a una cola. Si el paquete va a entrar una cola por debajo de una rata dada, se toma una acción (permite la entrada). Si el paquete va a entrar a una cola por encima de una rata dada, se toma otra acción. Aunque el policer usa un mecanismo interno de token -bucket, no tiene la capacidad de retrasar un paquete, tal como lo hace un mecanismo de shaping. Es decir, el policing es un mecanismo más abrupto de control de tráfico, y constituye una solución más radical en el tema de control de tráfico

QDISC: se denomina Disciplina de colas el algoritmo que gestiona el proceso de encolar 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).

ROUTER: es un dispositivo que está conectado a varios switchs o hubs, para permitir a cada una de sus redes la comunicación entre ellas.

Los enrutadores pueden ser configurados también para denegar la comunicación

entre servidores específicos en diferentes redes. También pueden filtrar tráfico basándose en el puerto TCP de cada paquete. Por lo tanto, los enrutadores dirigen y regulan el tráfico entre redes separadas

SHAPER: es una porción de software que pretende suavizar el flujo de tráfico en una interfaz, para evitar congestión en el enlace y cumplir con requerimientos de un proveedor de servicios. El shaper suaviza el tráfico tipo ráfaga para cumplir con el CIR configurado, mediante el encolamiento de los paquetes que exceden la rata principal

Por algún tiempo un mecanismo IP simple ha sido provisto por el kernel de Linux. Este es llamado dispositivo Shaper y fue diseñado para limitar la cantidad de ancho de banda que las rutas de datos pueden consumir. Consiste en un dispositivo de red virtual que es configurado con dos importantes parámetros: un límite de ancho de banda y un dispositivo de red físico al cual es ligado. La suma total de tráfico enrutado por el shaper será limitada para cumplir con el límite de ancho de banda, simplemente descartando los datagramas que podrían causar que se supere el límite.

SCHEDULING: Hablando de teorías de colas en redes de computadores, Scheduling es el proceso de toma de decisión por el cual los paquetes son ordenados y reordenados para su transmisión. Otros mecanismos genéricos de encolamaiento, tratan de compensar varias condiciones de la red: Un algoritmo de "colas justas" (SFQ) trata de prevenir que un solo cliente o flujo domine el uso de la red. Un algoritmo round-robin (WRR) le da a cada flujo o cliente un turno para desencolar paquetes. Algoritmos más sofisticados de encolamiento tratan de prevenir sobrecargas en el backbone de una red (GRED) o refinar otros mecanismos más sencillos ESFQ).

TRAMA: término típicamente usado para describir una unidad de datos de capa 2 (enlace de datos) que será reenviada al próximo recipiente. Las interfaces Ethernet, PPP y E1/T1, todas nombran su unidad de datos de capa 2 como trama. La trama es la unidad en la cual el control de tráfico es efectuado.

BIBLIOGRAFIA

[1] HARRIS Nick et al. Linux Handbook, A Guide to IBM Linux Solutions and Resources. Redbooks, 2003. <http://ibm.com/redbooks/sg247000>

p. 1,9,10,12-45, 47-57, 59-69. [Consulta: 15 ene. 2004]

[2] COOPER, Robert. Introduction to Queueing Theory. 2 ed. New York: North Holland, 1981 p. 1-2, 34-64.

[3] GELENBE, E. ; PUJOLLE G. Introduction to Queueing Networks. Great Britain: John Wiley & Sons Ltd. 1987 p. 1-8, 10-11

[4] VASTOLA, Kenneth. Performance Modeling and Analysis of Computer Communications Networks < http://networks.ecse.rpi.edu/~vastola/> [Consulta: 04 mar. 2004].

[5] STALLINGS, William. Sistemas Operativos. 2 ed. España: Prentice Hall, 1997. p. 631-650

[6] DAIGLE, John. Queueing Theory for Telecommunications. United States of America : Addison-Wesley, 1992 p. 23-46, 214

[7] NAIN, Philippe. Basic Elements of Queueing Theory – Application to the Modelling of Computer Systems. 1998 p. 4-50.

<http://www.cs.columbia.edu/~misra/comse6180/nain.pdf> [Consulta: 23 feb.2004]

[8] ADAN, Ivo; RESING, Jacques. Queueing Theory. 2001. 180 p.

<http://www.win.tue.nl/~iadan/sdp/> [Consulta: Consulta 23 feb 2004]

[9] SEMERIA, Chuck. Supporting Differentiated Service Classes:Queue Scheduling Disciplines. Juniper Networks Inc. 2001. p. 4-5, 6-17.

<http://www.juniper.net/solutions/literature/whire-papers/200020.pdf >

[Consulta: 27 ene 2004]

[10] MARSH, Mathew G. Policy Routing With Linux – On Line Edition

< http://www.policyrouting.org/PolicyRoutingBook> [Consulta: 15 feb 2004]

[11] ANDREASSON, Oskar. Iptables Tutorial 1.1.19 <http://iptables-tutorial.frozentux.net/iptables-tutorial.html> [Consulta: 24 ene 2004]

[12] KUZNETSOV, Alexey N. IP Command Reference. 1999. < http://linux-ip.net/gl/ip-cref> [Consulta: 25 ene 2004]

[13] STANIC, Milan P. tc- traffic control – Linux QoS control tool. < http://www.rns-nis.co.yu/~mps/linux-tc.html > [Consulta: 25 ene 2004]

[14] iproute2 + tc notes. <http://snafu.freedom.org/linux2.2/iproute-notes> [Consulta: 25 ene 2004]

BROWN, Martin A. Guide to IP Layer Network Administration with Linux : version 0.4.4 < http://www.securepipe.com> [Consulta: 25 ene 2004]

[16] Kernel Packet Traveling Diagram.

< http://www.docum.or/stef.coene/qos/kptd> [Consulta: 31 ene 2004]

[17] ALMESBERGER, Werner. Linux Network Traffic Control-Implementation Overview . Abril 23 1999 < http://www.almesberger.net/cv/papers/tcio8 > [Consulta: 17 ene 2004]

[18] HUBERT, Bert et al. Linux Advanced Routing & Traffic Control HOWTO

<http://www.linuxdoc.org/HOWTO/Adv-Routing-HOWTO.html> [Consulta: 24 ene 2004]

[19] BROWN, Martin A. Linux Traffic Control HOWTO <http://www.tldp.org/HOWTO/Traffic-Control-HOWTO> [Consulta: 24 ene 2004]

[20] http://www.opalsoft.net/qos [Consulta: 24 ene 2004]

[21] RADHAKRISHAN, Saravanan. Linux – Advanced Networking Overview Version1

< http://qos.ittc.ukans.edu/howto> [Consulta: 24 ene 2004]

[22] DEVERA, Martin; Cohen Don. HTB Linux queuing discipline manual – user guide. < http://luxik.cdi.cz/~devik/qos/htb/manual> [Consulta: 24 ene 2004]

ANEXO

Sistema Operativo Linux y Control de Tráfico en Redes de Computadores

Juan Carlos Rengifo Salazar, Álvaro Urrea Lujan

Abstract— This paper shows the traffic control which is the set tools that allows to have granular control over these queues and queuing mechanisms of a networked device, which has grown and matured under Linux kernels 2.2 and 2.4 .

Index Terms—classes, filters, policing, queuing disciplines.

INTRODUCCION

La amplia presencia redes de computadores, la necesidad de interconexión entre ellas para el intercambio de información y la exigencia de emplear eficientemente los recursos, llevan a considerar el empleo de aplicaciones de software libre para generar soluciones que satisfagan estas necesidades y exigencias, uno de los recursos importantes en la redes de computadores es el ancho de banda que se requiere para la transmisión de la información, es sabido que este es costoso y limitado, para utilizar eficientemente el canal se necesita controlar de alguna forma el trafico en la red o entre las redes, Linux aparece como una opción para dar solución a este problema, pues cuenta en su kernel con un módulo para redes el cual tiene todo el potencial de las soluciones propietarias a nivel comercial.

Linux trabaja bajo un esquema multitarea y multiusuario, cuenta con herramientas de desarrollo y por su estructura permite a los desarrolladores tener acceso al hardware y a las redes que a este se conectan. La siguiente tabla ilustra cual es la presencia de Linux

en el mundo de las telecomunicaciones y la informática:

Tabla 1. Presencia de Linux en el mundo de las telecomunicaciones y la informática [1]

Áreas de presencia del sistema operativo Linux

Client Interface

Line of Business

Middleware

Data Store

Infraestructura

Operating Systems

Si bien uno de los puntos fuertes de LINUX es su carácter de código abierto, en comparación a sistemas operativos como UNIX y WINDOWS, presenta otras fortalezas:

- Costo total de compra y envío inferior.
- Tecnología creciente y apoyo de la industria.
- Gran numero de ambientes y aplicaciones.
- Crecimiento constante del conocimiento del mismo por parte de los desarrolladores.
- Seguridad.
- Flexibilidad.

Desde la versión 2.0.x el kernel de Linux trae un conjunto de utilidades relacionadas con el enrutamiento avanzado y la conectividad de redes de computadores, que recogen temas como control de trafico, servicios diferenciados, calidad de servicio, firewalls, túneles, etc. Se puede destacar ipfwadm e ipchains en el kernel 2.2.x y netfilter a partir del kernel 2.3.15 y superiores con el comando iptables. Adicionalmente se cuenta con soporte para el modulo de control de trafico la utilidad TC (Traffic Control) de la herramienta IPROUTE2.

Linux nos brinda la posibilidad de efectuar tareas de control de tráfico, para gestión del ancho de banda de una red de computadores, algunas de ellas se listan a continuación:

- Limitar el ancho de banda conocido a un valor predeterminado.
- Limitar el ancho de banda de un cliente en particular.
- Reservar ancho de banda para una aplicación en particular de un usuario específico.

Se hace necesario disponer de soluciones que hagan un uso eficiente del ancho de banda disponible y que además se puedan brindar servicios diferenciados, Linux se convierte entonces en una de ellas, por su creciente popularidad, utilidades, evolución, costos y flexibilidad.

En el presente trabajo se revisan la las herramientas con que cuenta el sistema

operativo Linux, para realizar tareas de control de trafico, se toma como referencia la versión REDHAT 9, cuyo kernel es 2.4.20-8, este inicia tratando los conceptos básicos necesarios alrededor de la teoría de colas para comprender su importancia en las redes de computadores, enseguida se abordan los conceptos presentes en el control de trafico, luego se revisan las características de Linux relacionadas con el enrutamiento avanzado se presentan las herramientas fundamentales para efectuar tareas tales como túneles, firewall, múltiples tablas de enrutamiento, gestión del ancho de banda, multicasting, etc., se dedica una sección a la temática de control de trafico manejada desde la óptica del sistema operativo Linux, centrándose en presentar los conceptos de disciplinas de colas, clases y filtros, finalmente se presenta un ejemplo practico de aplicación de los conceptos de gestión de ancho de banda sobre una red de computadores simple, con el fin de ilustrar el potencial con que cuenta Linux.

Teoría de Colas y las Redes de Computadores

Las líneas de espera o colas y los sistemas de colas han sido sujeto de extensos estudios, desde el mismo surgimiento de la telefonía, hasta hoy con la presencia de amplias redes de computadores.

Sobre todo a nivel de redes de computadores el tráfico de datos que circula en estas requiere de evaluación de modelos que incluyan análisis de colas. Se habla entonces de la Teoría de Colas la cual pertenece a las matemáticas y específicamente a una rama aplicada de las probabilidades y los procesos estocásticos, se busca entonces dar respuesta de como es el comportamiento de ingreso a una cola, los tiempos de espera, el servicio, el tiempo de servicio y la salida del sistema buscando la optimización del mismo.

En redes de comunicación, por ejemplo en una red de computadores los paquetes que fluyen por la red solicitan ser servidos por los diferentes nodos (Routers, Switches, Bridges, Hubs), se requiere considerar entonces el ancho del canal de transmisión, la velocidad de procesamiento, el tamaño del buffer y la priorización del flujo de estos paquetes en la red. Para el diseño y evaluación de equipos de procesamiento y redes de computadores se necesita hacer uso de la Teoría de Colas, mediante el empleo de modelos de colas y análisis simplificado de estos, soportados en suposiciones que permiten reducir la complejidad del análisis, los cuales conducen a resultados satisfactorios.

En general para un modelo de colas se pueden definir tres características básicas: proceso de entrada, mecanismo de servicio y disciplina de cola.

Proceso de entrada: describe la secuencia de solicitudes de servicio, generalmente el proceso de entrada es descrito en términos de distribución a lo largo del tiempo entre instantes consecutivos de ingreso de clientes.

Mecanismo de servicio: tiene que ver con el número de servidores y la cantidad de tiempo que el cliente usa los servidores.

Disciplina de colas: especifica la disposición de que se tiene para bloquear los clientes (clientes que encuentran todos los servidores ocupados), puede darse que los clientes bloqueados decidan abandonar el sistema inmediatamente o que esperen en la cola por el servicio, los cuales pueden ser atendidos en orden especifico. [2,3]

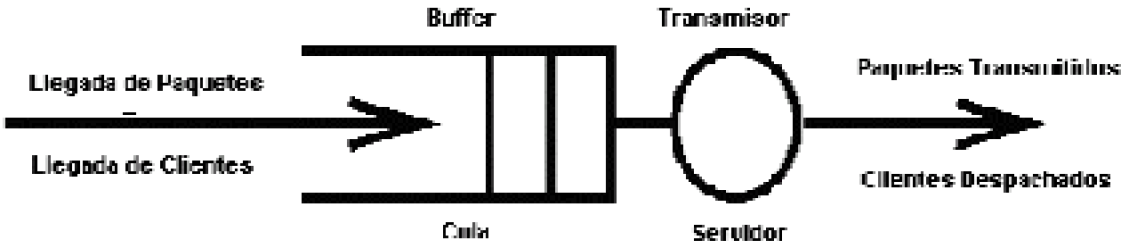


Figura 1. Modelo de Cola de un solo servidor. [4]

La figura 1 presenta un esquema de cola de un solo servidor donde se visualizan en la parte superior los términos de las redes de conmutación de paquetes en un enlace de transmisión y en la parte inferior los términos asociados a la teoría de colas.

La teoría de colas es matemáticamente compleja, sin embargo es posible a nivel aplicativo realizar un análisis de colas de manera directa, basta con tener conocimientos básicos de estadística y comprender la aplicabilidad de la teoría de colas.

En general un sistema de colas se clasifica de acuerdo a sus propiedades, algunas de ellas son:

- La forma de la distribución de llegada de los clientes.
- La forma de la distribución del tiempo de servicio.
- El numero de llegadas en un grupo.
- El numero de servidores.
- La disciplina de servicio, es la manera en el cual el servicio es condicionado, si el sistema tiene prioridades.
- Numero de clientes que se les permite esperar.
- Numero de clientes en la población. [6]

Algunas de las cantidades de interés en el estudio de sistemas de colas son la distribución del tiempo de espera, la distribución del tiempo del sistema, la distribución del numero de clientes en el sistema, la probabilidad de que el servidor este ocupado o no, la distribución de la longitud de un periodo de ocupación, la distribución del numero de clientes servidos durante un periodo de ocupación, promedios para tiempos de espera.

Muchos sistemas de colas en el proceso de llegada son caracterizados como tipo Poisson, pues al asumir que el comportamiento de llegada de los clientes en una cola se asemeja a un proceso de Poisson, nos permite reducir la complejidad analítica del problema permitiendo obtener resultados útiles.

Se tiene un proceso A(t) que corresponde al numero de llegadas en un tiempo t donde A(0)=0, y A(t) - A(s) son el numero de llegadas en un intervalo (s, t].

El proceso A(t), con t >= 0sigue una distribución de Poisson con parámetro

$$\lambda\tau$$

de acuerdo a:

Donde

$$\lambda\tau$$

numero medio de llegadas y

$$\lambda$$

es la rata de llegadas.

Cabe destacar dos propiedades importantes de un proceso de Poisson:

1. Mezclado: Se asume que se tienen N1(t) y N2(t) ambos procesos de Poisson independientes con ratas

$$\lambda_1 \text{ y } \lambda_2$$

respectivamente de estos dos procesos se da origen a un nuevo proceso de Poisson con rata

2. División: Se asume que se tiene un proceso de Poisson N(t) con una rata

$$\lambda$$

y cada llegada esta marcada por una probabilidad P independiente de las otras llegadas. Dejando que N1(t) y N2(t) sean respectivamente los procesos de salida de este, entonces se tiene una distribución de Poisson con ratas

$$p\lambda \text{ y } (1-p)\lambda$$

para cada camino de salida respectivamente.

Se tiene entonces que un proceso de Poisson se mantiene bajo el mezclado o la división, figuras 2 y 3.

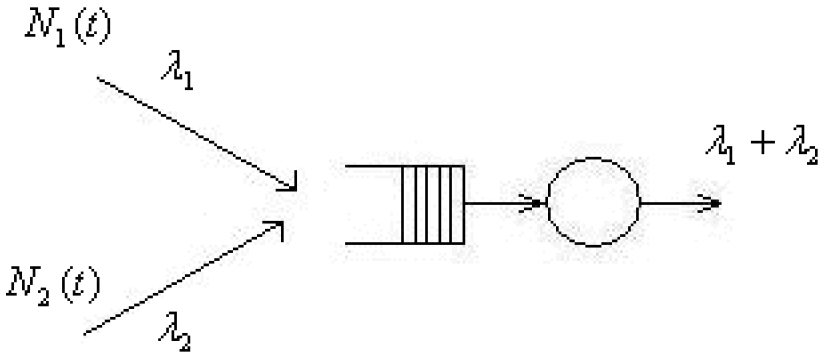


Figura 2 Propiedad de Mezclado de un proceso de Poisson

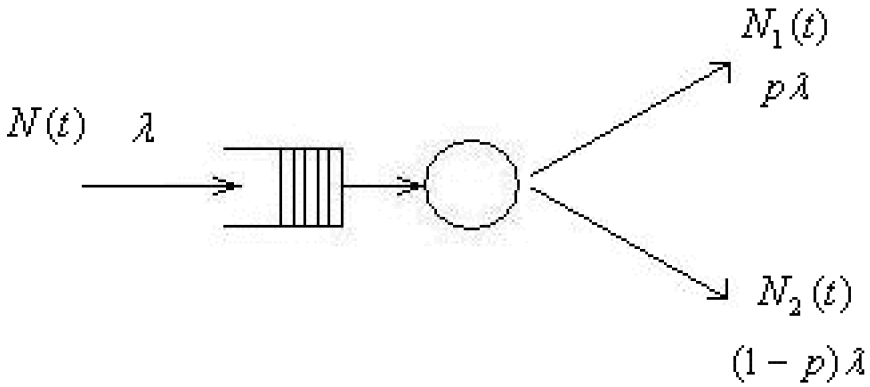


Figura 3. Propiedad de División de un Proceso de Poisson

En general se puede presentar el modelo de colas gráficamente figura 4, en la cual aparecen los parámetros de interés en un modelo de colas, donde

$$\lambda$$

rata de llegadas o numero medio de llegadas por segundo, Lq numero de clientes esperando ser servidos, W tiempo de espera (incluye los clientes que han de esperar y los clientes con tiempo de espera = 0), L numero de clientes en el sistema tanto los que están en espera como los que están siendo servidos, S tiempo que gasta un cliente en el sistema, u tiempo de servicio por cada llegada, que es el tiempo empleado por el servidor para atender un cliente y no incluye el tiempo de espera en la cola , p fracción de tiempo que el servidor o servidores están ocupados. Con estos parámetros podemos no solo caracterizar la cola sino que además permite realizar un análisis de la misma.

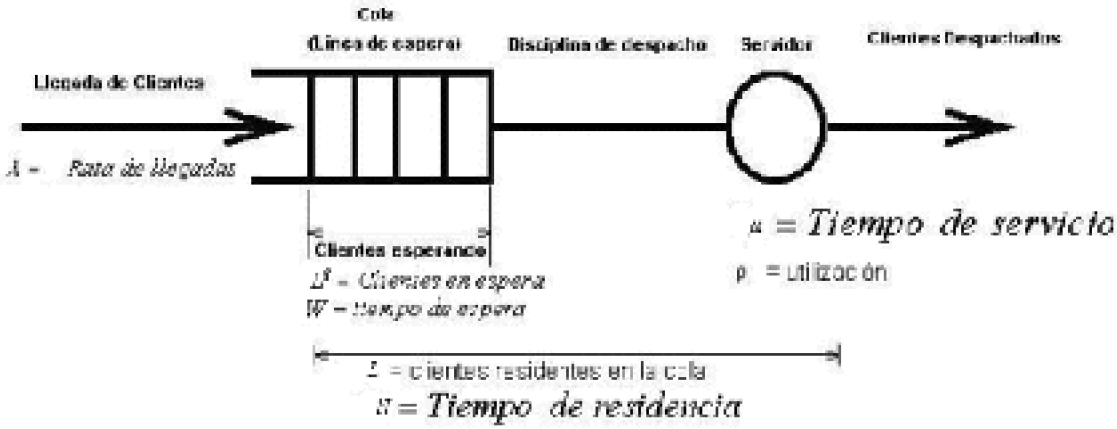


Figura 4. Cola de un solo servidor con sus parámetros. [5]

En un esquema donde existe un solo servidor el cual provee servicio a los clientes, estos últimos llegan solicitando ser servidos, si el servidor se encuentra desocupado un cliente es atendido, de lo contrario los clientes que llegan se unen a una línea de espera o cola, adicional a esto el servidor se toma un tiempo en atender cada cliente.

Para el caso que se tenga una cola que comparte múltiples servidores, si llega un

cliente y existe un servidor libre este es atendido, pero si todos los servidores se encuentran ocupados se inicia la formación de una línea de espera o cola. Se puede asumir que los servidores en paralelo son idénticos, los clientes son servidos en orden de llegada. Según la notación de Kendall se tiene un modelo M/M/c, con comportamiento exponencial entre llegadas, tiempo de servicio exponencial y numero finito de servidores c, ver figura 7.

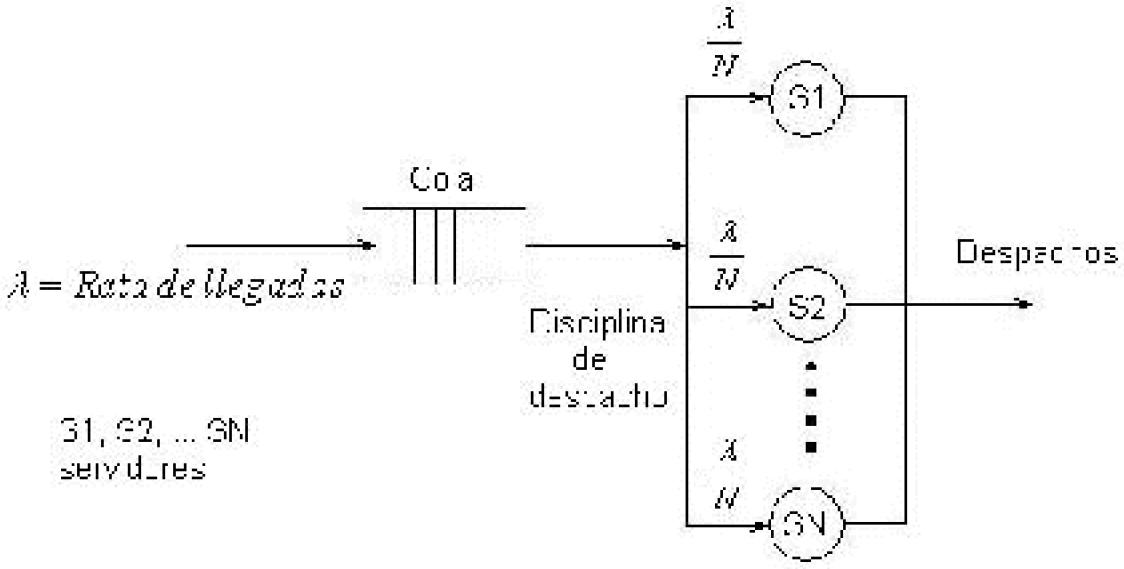


Figura 5. Cola con múltiple servidores [5]

La presencia de clientes con diferentes exigencias crea la necesidad de pensar en priorizar el servicio de atención de estos, se revisara este tópico usando el modelo de cola con un solo servidor. Se requiere entonces servir a los clientes en orden de prioridad debiendo ser atendidos primero aquellos con una prioridad mayor. En un sistema de colas con prioridad los clientes son divididos en K clases, con i =1,2....K, la menor prioridad, el mayor numero de clase, es decir, clientes con prioridad i se les da preferencia sobre los clientes con prioridad j si i<j Los clientes llegan de acuerdo a un proceso de Poisson con rata

$$\lambda_i$$

el tiempo de servicio es independiente con una función de distribución Gk(x), media

$$1/\mu .$$

según notación de Kendall el modelo es M/G/1. Se asume que los tiempos de servicio y los tiempos de llegada son independientes e idénticamente distribuidos. Adicionalmente se tiene que

$$\rho = \lambda / \mu < 1$$

es la intensidad de tráfico de la clase i.

De la figura 8 se asume que en cada clase de cola los clientes son servidos en el orden primero en ingresar-primero en salir y existe una cola por cada clase.

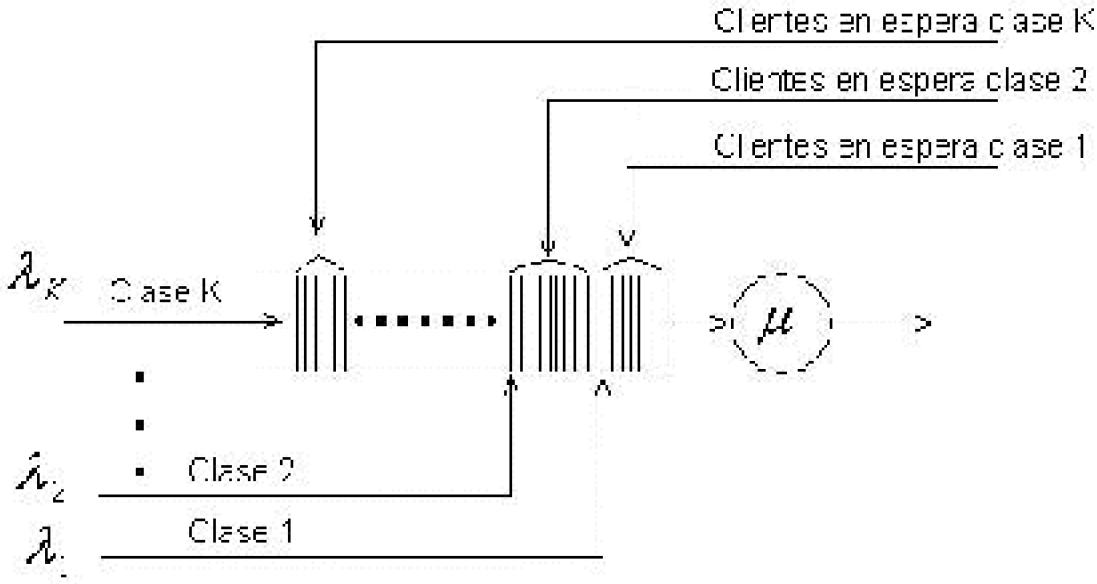


Figura 6. Cola de un solo servidor con prioridad [6]

Se consideran dos tipos básicos de prioridad, la primera es prioridad de no interrupción donde los clientes de mayor prioridad no pueden interrumpir el tiempo de servicio de un cliente de menor prioridad debiendo esperar a que termine de ser servido para poder ser atendidos, la segunda es prioridad de interrupción –reinicio aquí las interrupciones son permitidas para dar servicio a los clientes de mayor prioridad debiendo reiniciar el tiempo servicio después de la interrupción en el punto donde fue interrumpido, existe también la opción de una prioridad de interrupción-repetición en este caso particular el cliente que fue interrumpido cuando de nuevo es servido, su servicio vuelve a empezar como si hubiese acabado de ingresar al servidor. Estos dos tipos de prioridad también son conocidos como no absoluta y absoluta respectivamente.

En ambientes distribuidos no solo existen colas aisladas, si no que estas se encuentran interconectadas formando una red de colas, esta situación lleva a que el análisis sea de mayor complejidad y dos elementos que contribuyen a esto son:

- La división y mezclado del tráfico.
- La existencia de colas en tandem o serie.

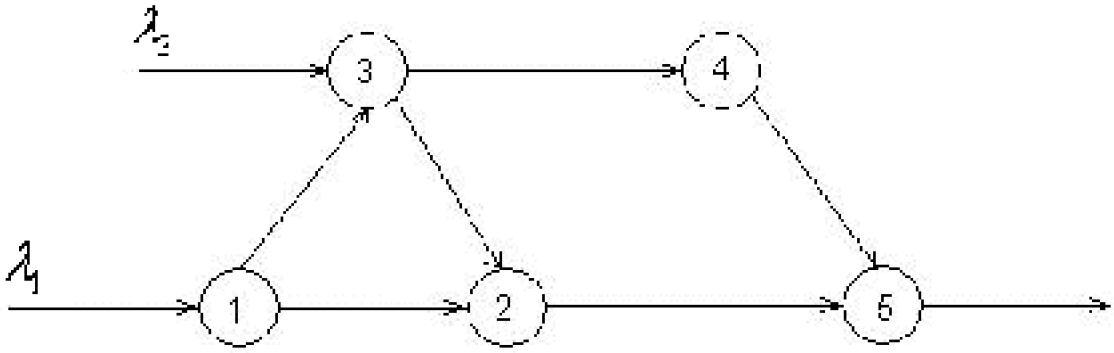


Figura 7. Grafico de una red de colas [5]

De la figura se tiene que los nodos representan colas y las líneas son el flujo de tráfico, aquí en el nodo 5 muestra un mezclado de tráfico, el nodo 1 división de tráfico y los nodos 3-4 y 1-2 ilustran colas en serie.

Básicamente cuando se habla de redes de colas se presentan dos modelos las redes de colas abiertas y las redes de colas cerradas y asociado a este tema de redes esta presente el teorema de Jackson y las redes de Jackson.

El teorema de Jackson es utilizado para realizar análisis de colas en redes de colas, alrededor de este tema se habla de redes de Jackson, para este caso una red de Jackson es una red de K colas que satisface tres propiedades:

- Los servidores en cada una de las K colas son independientes unos de otros y tiene un tiempo de servicio

$$\mu_i$$

- con i=1,2.....K.
- Las llegadas externas en cada cola siguen un proceso independiente de Poisson con ratas

$$\lambda_i \geq 0$$

para i=1,2.....K..

Las redes utilizan un enrutamiento aleatorio.

El teorema de Jackson establece:

- Las colas en una red de Jackson son colas independientes tipo M/M/1.
- El numero de paquetes y los retrasos en cada cola son independientes de estos en todas las otras colas.

Si la probabilidad $P(n_1, n_2, \ldots n_K)$ es igual a la probabilidad n_1 paquetes en la cola Q_1, n_2 paquetes en la cola Q_2 ... n_K paquetes en la cola Q_K entonces se tiene:

$$P(n_1, n_2, \ldots n_K) = \rho_1^{n_1}(1 - \rho_1)\rho_2^{n_2}(1 - \rho_2)\ldots\rho_K^{n_K}(1 - \rho_K) \quad (16)$$

Donde $\rho_k = \frac{\lambda_k}{\mu_k}$ con λ_k como la rata de llegada de todos los paquetes a la cola Q_k para $k = 1,2,\ldots,K$

La figura 12 presenta una red LAN que se conecta a través de un router a Internet, se puede apreciar como las estaciones y los servidores generan trafico tanto al interior de la red como hacia Internet, este trafico debe ser manejado ya sea por las estaciones, servidores y router, en cada una de estos equipos se generan colas de tramas de datos, es decir, se tiene una red de colas, las cuales deben ser manejadas de alguna forma, la mas típica es tipo FIFO, pero como se vera mas adelante existen muchas otras formas de atender este trafico, lo importante aquí es la presencia de colas a lo largo de la red, ya sea al interior de una estación debido a que las diferentes aplicaciones que corren solicitan servicio de procesamiento, en los servidores pues deben servir las solicitudes de las estaciones y en el caso del router este debe atender todo el trafico saliente y entrante de la LAN, en general se dispone de una red de colas, la comprensión de los elementos básicos relacionados con las colas permite entender como se puede controlar el trafico de datos en una red, pensando en definir algún tipo de prioridad para los diferentes clases de flujo de datos según sus características, es decir, definidos por origen, destino, tipo de aplicación, transferencia de datos, correo electrónico, navegación, consultas a bases de datos, gestión remota de equipos, etc., todos estos elementos pueden ser tenidos en cuenta para realizar el control de dicho trafico, pues al considerar temas como tamaño de cola, tipo de servicio, tiempo de servicio, mecanismo de atención y prioridad elementos característicos de una cola es posible desarrollar algoritmos que consideren todos estos criterios.

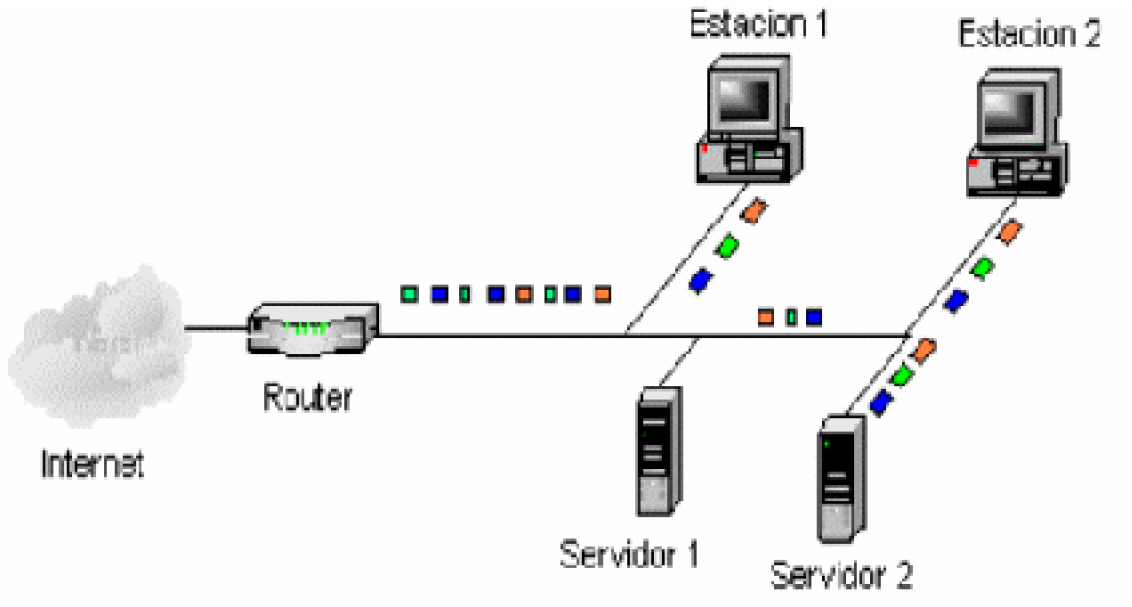


Figura 8. Esquema simple de red

CONTROL DE TRAFICO

Control de tráfico es el nombre dado al conjunto de sistemas de encolado y mecanismos por los cuales los paquetes son recibidos y transmitidos en un enrutador. Esto incluye la toma de decisión de a que rata aceptar los paquetes en la entrada de una interface y determinar cuales paquetes transmitir, en que orden y a que rata a la salida de una interface. Dado que los enlaces de red transmiten los datos en una forma serial, una cola es requerida para manejar los paquetes de datos salientes.

El control de tráfico es el grupo de herramientas que permiten a un administrador tener un control granular sobre estas colas y los mecanismos de encolamiento de un dispositivo en red. El término calidad de servicio es usualmente usado como sinónimo de control de tráfico.

Dentro de las razones de uso del control de tráfico es que este permite al administrador encolar paquetes de manera diferente, basado en los atributos del paquete. Incluso se puede usar para simular el comportamiento de una red basada en circuitos.

Algunos ejemplos de problemas comunes los cuales se resuelven o al menos se amainan mediante el uso de control de tráfico, que pueden dar una idea del tipo de problemas que pueden ser solucionados usando estas técnicas además de maximizar el uso de una conexión de red, son:

- Limitar el ancho de banda a un valor conocido.
- Limitar el ancho de banda a un usuario, servicio o cliente determinado.
- Maximizar el troughput de un enlace asimétrico, priorizar la transmisión de paquetes y ACK.
- Reservar ancho de banda para un usuario o aplicación particular.
- Dar prioridad al tráfico sensitivo a la latencia.

- Permitir distribución equitativa de ancho de banda no reservado.
- Asegurar que un tipo particular de tráfico sea rechazado.

Dentro de las ventajas que ofrece el control de tráfico cuando se emplea apropiadamente, es de poder lograr tener un uso mas predecible de los recursos de la red y a una contención menos volátil de los mismos, la red cumple entonces con las metas de la configuración del control de tráfico, para el caso de tráfico basura se puede reservar una cantidad razonable de ancho de banda, aun cuando el tráfico interactivo este simultáneamente servido, incluso al tráfico de baja prioridad como el correo electrónico se le puede reservar ancho de banda sin afectar otros tipos de tráfico, además si la configuración de control de tráfico representa una política que ha sido comunicada a los usuarios, entonces ellos sabrán que esperar de la red.

En lo que respecta a las desventajas, la complejidad es fácilmente la mas significante al usar control de tráfico, sin embargo hay formas de familiarizarse con las herramientas de control de tráfico, las cuales facilitan la curva de aprendizaje acerca de control de tráfico y sus mecanismos, pero identificar una mala configuración de control de tráfico puede ser todo un reto.

Existen una serie de elementos a considerar cuando se habla de control de tráfico, estos son:

Colas: una cola es una locación (o buffer) conteniendo un número finito de ítems esperando por una acción o servicio.

Flujos: un flujo es una conexión o conversación entre dos hosts, un único set de paquetes entre dos hosts puede ser considerado como un flujo. Bajo el concepto TCP una conexión con una dirección y puerto fuente y una dirección y puerto destino, representa un flujo.

Token y Buckets: Un flujo token bucket se define por (r,b), r denota la rata a la cual los tokens (créditos) se acumulan y b es la profundidad de la “piscina” de tokens (en bytes).

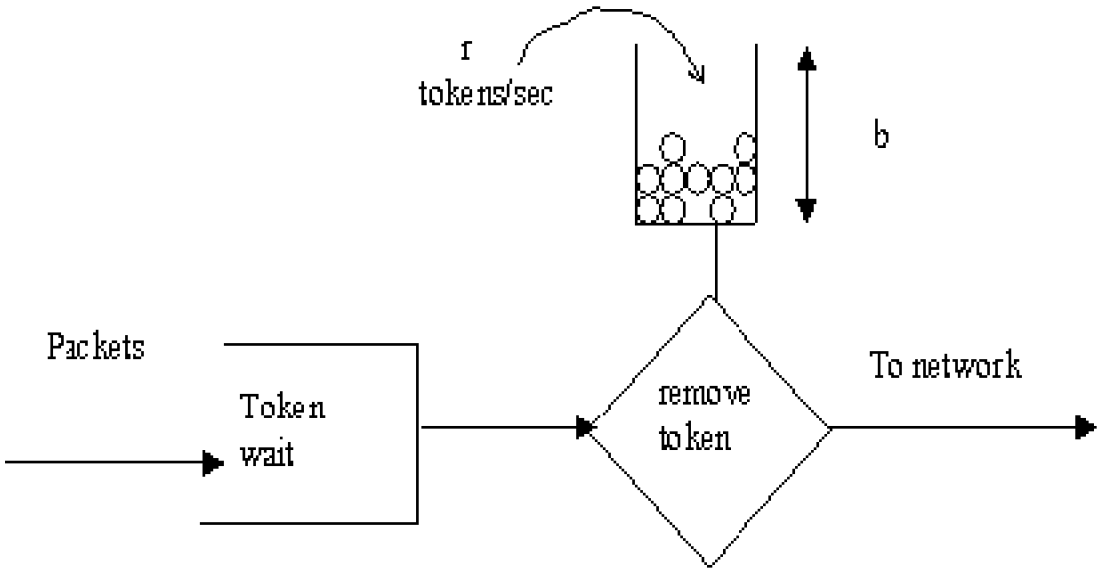


Figura 9. El concepto token y bucket [7]

Como se muestra en la figura anterior, los nuevos token se adicionan al bucket a una rata de r tokens/sec, el máximo de tokens que se puede acumular es de b bytes. Si el bucket está lleno, los tokens que entren serán arrojados afuera. El perfil Token Bucket contiene tres parámetros: una rata promedio, una rata máxima y un tamaño de ráfaga.

Traffic Shaping (Ajuste de Trafico): El ajuste de tráfico trabaja midiendo y encolando paquetes IP en tránsito, es decir, tratan de medir la cantidad de tráfico priorizado y limitar el tráfico no priorizado dinámicamente, de manera que no interfiera con el troughput del tráfico priorizado.

Scheduling: es el mecanismo mediante el cual los paquetes son organizados entre la entrada y la salida de una cola en particular.

Classifying: La clasificación es el proceso mediante el cual los paquetes son separados para diferente tratamiento, posiblemente en diferentes colas de salida.

Policing: Los controladores miden y limitan el tráfico en una cola particular, es el mecanismo mediante el cual el tráfico puede ser limitado.

Disciplinas de Colas: Una disciplina de colas, permite manejar el acceso a una cantidad fija de BW de un puerto de salida, mediante la selección del siguiente paquete que será transmitido a dicho puerto. Hay diferentes disciplinas de colas, y cada una apunta a encontrar el balance entre complejidad, control y equilibrio, buscando darle manejo a la congestión. Es así como cada dispositivo de red puede contar con una disciplina de colas asociada, lo cual no es mas que el algoritmo que controla las colas, es decir, como los paquetes encolados en el dispositivo son tratados.

La disciplina de colas mas simple puede consistir solo de una cola, donde todos los paquetes se almacenan en el orden en el que llegaron y la cual se vacía tan rápido como el dispositivo al que sirve lo puede hacer. Disciplinas de colas mas elaboradas pueden usar filtros para distinguir entre diferentes clases de paquetes y procesar cada clase de una forma diferente, por ejemplo, dándole prioridad de alguna clase sobre otras.

Las disciplinas de colas tradicionales son:

- Colas FIFO (First In First Out).
- Colas de prioridad (PQ, de Priority Queuing).
- Colas justas (FQ de Fair Queuing).
- Colas justas por pesos (WFQ de Weighted fair Queuing).
- Colas por pesos round-robin o colas basadas en clases (WRR o CBQ).
- DWRR. (Deficit Weighted Round Robin).

IV. ENRUTAMIENTO AVANZADO CON LINUX

El concepto de enrutamiento avanzado esta relacionado con la posibilidad de explotar al máximo las herramientas con que cuenta Linux para efectuar tareas de gestionar trafico entre diferentes redes, es decir, que este sistema operativo cuenta con todo lo necesario para realizar tareas similares e incluso superiores a un enrutador comercial, tales como tener varias tablas de enrutamiento, reservar ancho de banda, brindar calidad de servicio (QoS), balancear carga, túneles, cortafuegos, multicasting, etc.

Con Linux a nivel de enrutamiento es posible realizar tareas tales como:

- Controlar el ancho de banda de algunos computadores.
- Controlar el ancho de banda hacia algunos computadores.
- Compartir el ancho de banda de manera justa.
- Proteger la red de ataques de negación de servicios (DoS).
- Proteger a Internet de sus clientes.
- Balancear la carga de múltiples servidores.
- Restringir el acceso a sus computadores.
- Restringir el acceso de sus usuarios a otros computadores.
- Enrutamiento basado en la identificación del usuario, dirección MAC, dirección fuente, puerto, tipo de servicio, hora del día y contenido.

Todas estas posibilidades que brinda el kernel de Linux lo hacen viable para considerarlo como una opción cuando se trata de ofrecer soluciones a nivel de redes de computadores, pues dentro del conjunto de sistemas operativos se convierte en un fuerte competidor, gracias a las características de enrutamiento, clasificación y filtrado.

Cuando se habla de enrutamiento se piensa en tablas para enrutar el trafico entre diferentes redes, tradicionalmente enrutar trafico IPv4 es un proceso de dirigir los paquetes basados en su destino, para ello Linux cuenta con dos herramientas básicas para realizar dicha tarea que son ifconfig y route, estas basan su funcionamiento en la dirección destino del paquete, es decir, se verifica cual es su destino para basados en la tabla de enrutamiento reenviar el paquete a su destino, con estos comandos se busca configurar las interfaces e inicializar las tablas de enrutamiento.

Lo importante aquí es poder enlutar paquetes diferentemente para ello se requiere

pensar en políticas de enrutamiento, las cuales están relacionadas con enrutar los paquetes no solo basados en su dirección destino sino también se pueden considerar la dirección fuente, el protocolo IP, los puertos del protocolo de transporte o el contenido del paquete, es decir, pensar en políticas de enrutamiento es proveer capacidad de enrutamiento basados en una o todas las facetas de un paquete.

El núcleo de la política de enrutamiento esta cimentada en tres elementos, estos son dirección, ruta y regla.

Dirección: define la localización del servicio

Ruta: establece la ubicación de la dirección

Regla: describe la localización de la ruta

Lo interesante de definir una política de enrutamiento es emplear estos tres elementos de manera combinada, ofreciendo así mayor flexibilidad y posibilidad de abordar tareas más complejas.

Hablar Linux en términos de enrutamiento avanzado, se refiere a definir estructuras de políticas de enrutamiento a través de la base de datos de políticas de enrutamiento (RPDB – Routing Policy DataBase)[8], que es un conjunto de rutas, tablas de rutas y reglas, RPDB es parte integral de Linux desde el kernel 2.2, y es una lista lineal de reglas ordenadas por un valor numérico que define la prioridad, básicamente provee una estructura interna y un mecanismo para implementar las reglas en la política de enrutamiento.

RPDB permite verificar características de los paquetes tales como dirección fuente, dirección destino, interface de entrada, TOS y emplear valores fwmark para verificar protocolos IP y puertos de transporte.

En RPDB cada regla de enrutado de la política esta formada por un selector y una acción, lo que se hace al interior de la base de datos es revisar en orden de incremento de la prioridad con el selector de cada regla aplicada a la dirección fuente, dirección destino, interface de entrada, TOS y fwmark, si el selector halla la coincidencia entonces la acción es realizada, de lo contrario se evalúa la siguiente regla.

RPDB puede contener reglas de los siguientes tipos:

- Unicast: Retorna la ruta encontrada en la tabla de enrutamiento referenciada por la regla.
- Blackole: Rechaza un paquete silenciosamente.
- Unreachable: Genera un error red inalcanzable.
- Prohibit: Genera un error comunicación administrativamente prohibida.
- Nat: traduce la dirección fuente del paquete IP a otro valor.

Otra característica de RPDB es su compatibilidad con las utilidades estándares de red tales como ifconfig y route, pero no se requiere hacer uso de estas pues basta la utilidad ip que es superior a estas. En Linux se puede hacer uso de características extendidas de la política de enrutamiento basta con compilar el kernel para que de soporte a NETLINK y

RT_NETLINK.

La RPDB constituye el núcleo que provee la funcionalidad alrededor de la cual la política de enrutamiento y las características de enrutamiento avanzado pueden ser construidas.

En Linux se dispone de dos herramientas que permiten tratar temas de enrutamiento avanzado, netfilter con su utilidad iptables e iproute2 con las utilidades ip y tc.

Iptables: Utilidad de Netfilter usada para establecer reglas para filtrado de paquetes, definición de cortafuegos (firewalls), esta basada en un filtrado puro de paquetes

Haciendo uso de la utilidad iptables de Netfilter, es posible filtrar los paquetes o marcarlos, los paquetes pueden ser etiquetados con un número, en concreto con iptables es posible filtrar el tráfico por:

- Dirección fuente
- Dirección destino
- Puerto fuente
- Puerto destino
- Identificación de protocolo

La sintaxis típica para generar una regla con iptables es:

iptables [-t table] command [matches] [target/jump] [9]

Tablas en iptables son:

mangle: Usada para el marcaje de los paquetes, para cambiar su contenido, cambiando sus encabezados.

filter: Empleada para filtrado de paquetes.

nat: Para realizar traducción de direcciones.

Lo importante es que al usar la tabla mangle de iptables es posible hacer uso de características avanzadas para el enrutamiento de trafico, las etiquetas validas únicamente con esta tabla son: TOS, TTL y MARK, la etiqueta TOS brinda la posibilidad de actuar sobre el campo TOS (Type of Service) del paquete, permitiendo definir políticas para el enrutamiento de los paquetes, tales como minimizar el retraso, maximizar la velocidad, minimizar el costo, servicio normal, maximizar la transferencia, maximizar la fiabilidad, la etiqueta TTL se emplea para modificar el campo TTL (Time To Live) del encabezado IP.

Iptables es una utilidad que permite definir un conjunto de reglas para filtrar el trafico que circula por lo menos entre dos redes, permitiendo con este filtrado proteger la maquina que hace esta labor y las redes detrás de esta, adicional a esto combinado con otras utilidades como ip y tc de iproute2, es posible desarrollar tareas de enrutamiento avanzado.

Iproute2: La posibilidad de disponer de la herramienta iproute2 en el sistema operativo Linux, en especial en versiones de kernel 2.4.x.+ , brinda opciones como

túneles, multicasting, control del tráfico, múltiples tablas de enrutamiento, balanceo de carga, etc.

A partir de la versión de kernel 2.2, se cuenta con el socket NETLINK, que en principio permite desarrollar en espacio de usuario código para gestión de paquetes y trafico IP, llevo a que Alexey Kuznetsov desarrollará el código iproute2, el cual permite hacer entre muchas cosas las siguientes:

Unificar los comandos relacionados con la gestión del trafico IP.

- Monitoreo de los periféricos, direcciones y rutas
- Gestión de tablas ARP.
- Uso de tablas de enrutamiento múltiples
- Creación de túneles IP.
- Reserva de ancho de banda.

Utilidad ip: a continuacion presenta la sintaxis general de esta utilidad, la cual es utilizada para configurar y gestionar interfaces de red, definir rutas, reglas y tablas de enrutamiento.

ip [OPTIONS] OBJECT [COMMAND [ARGUMENTS]]

OPTIONS se refiriere a un conjunto de modificadores que afectan el comportamiento y la salida de la utilidad ip, todas las opciones empiezan con el carácter “-“ y pueden ser usados en forma larga o corta, las opciones disponibles son [10]:

-V, -Version: imprime la versión de la utilidad ip.

-s, -stats, -statistics: entrega como salida mayor información.

-f, -family {inet, inet6, link}: forza el uso de una familia de protocolos.

OBJECT tiene que ver con el tipo de objeto con el que se desea operar u obtener información, estos son:

- link: dispositivo de red físico o lógico.
- address: direccion IPv4 o IPv6 sobre el dispositivo.
- neighbour: entrada cache ARP o NDISC.
- route: entrada a tabla de enrutamiento.
- rule: regla en la RPDB.
- maddres: direccion multicast.
- mroute: entrada al caché del enrutamiento multicast.
- tunnel: tunel sobre IP.

COMMAND especifica la acción a realizar sobre el objeto, el conjunto de acciones depende del objeto sobre el que se actúa, los más típicos son add, delete y show, no todos los objetos permiten todas las acciones anteriores y tienen comandos adicionales. El comando help esta disponible para todos los objetos el cual muestra los comandos

disponibles y la sintaxis.

ARGUMENTS es una lista de opciones de comando específica del comando, estos argumentos dependen del comando y del objeto, existen dos tipos de argumentos: flags (consisten de una sola palabra clave), parameters (consisten de una palabra clave seguida de un valor).

Utilidad tc: esta utilidad de iproute2 interactúa con el kernel para la creación, borrado o modificación directa de estructuras de control de tráfico, su sintaxis es como sigue:

tc [OPTIONS] OBJECT { COMMAND | help } [11]

Donde OBJECT se refiere a qdisc, class o filter, OPTIONS define { -s[tatistics] | -d[etails] | -r[aw] }

Para el manejo de la utilidad tc se requiere conocer los parámetros particulares de cada qdisc, class o filter, que pueden diferir de una a otro en cada objeto.

Por qdisc se entiende el conjunto de disciplinas de colas que Linux puede manejar, class define las clases para las disciplinas que soportan clases y filter brinda la posibilidad de clasificar paquetes.

Con tc es posible definir estructuras para gestionar el ancho de banda de un enlace y cuenta con unas reglas para especificación del ancho de banda:

mbps = 1024 kbps = 1024 x 1024 bps (bytes/s)

kbps = 1024 bps = 1024 (bytes/s)

kbit = 1024 (bits/s)

mbit = 1024 kbits (kilobits/s)

mb = 1024 kb = 1024 x 1024 b (byte)

Para la definición de tiempo se tiene ms, msec o msecs se refiere a milisegundos y us, usec o usecs define microsegundos.

Se puede apreciar en la siguiente figura que le ocurre a un paquete dentro del kernel, se establecen diferencias según sea procesado por la utilidad iptables (ipchains) o iproute2, para el caso de control de trafico se deben observar los bloques delineados en rojo, lo cuales son controlados por las utilidades ip y tc, donde la referencia a QOS se asocia a control de trafico, se ve la existencia de un tratamiento a la entrada y a la salida y esta presente la RPDB o PDBB (Routing Policy DataBase).

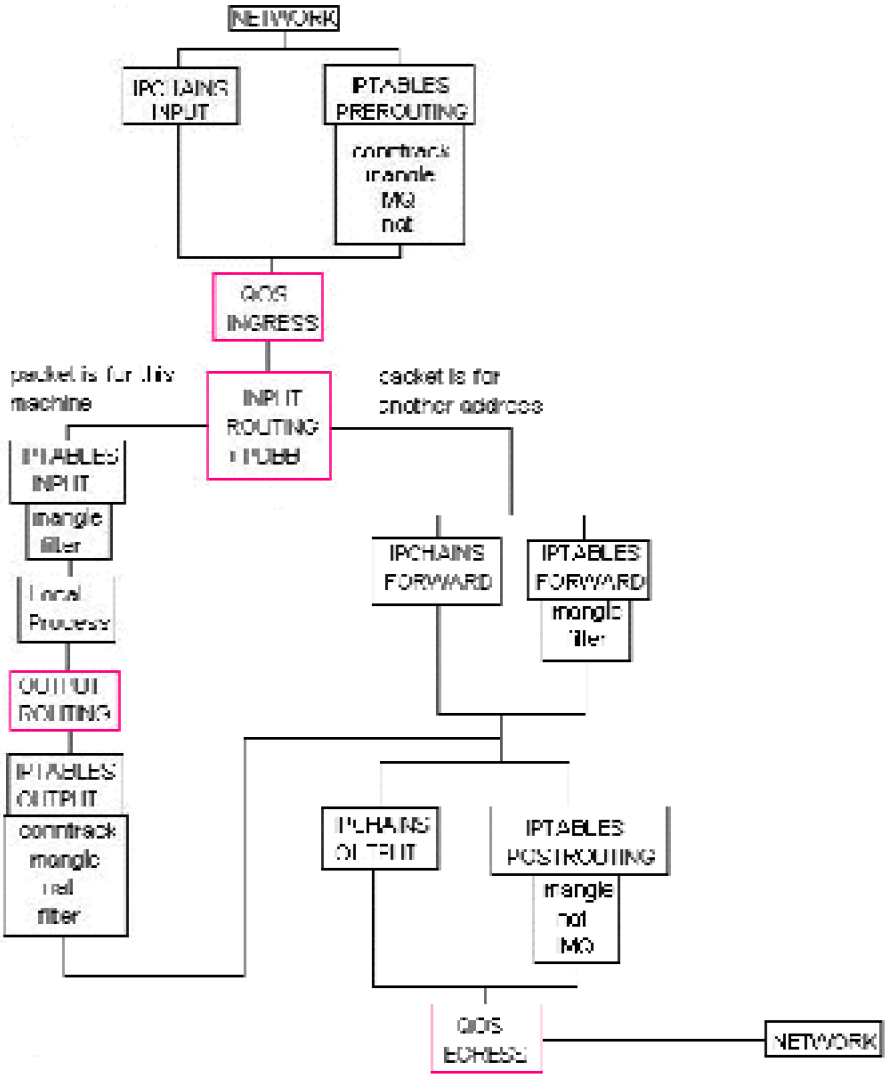


Figura 10. Diagrama sobre como viaja un paquete en el kernel [12]

LINUX Y EL CONTROL DE TRÁFICO

El sistema operativo Linux a partir de la versión de kernel 2.2.x cuenta con un conjunto de herramientas o utilidades que permiten realizar tareas de enrutamiento avanzado, esto consiste en poder efectuar la gestión y manipulación del los paquetes que se transmiten.

El control de tráfico hace referencia a un conjunto de sistemas de colas y los mecanismos bajos los cuales los paquetes son recibidos y transmitidos por un router, buscando decidir cuales paquetes y a que rata se les permite la entrada a través de una interface y también se debe determinar que paquetes serán transmitidos y en que orden por la interface de salida.

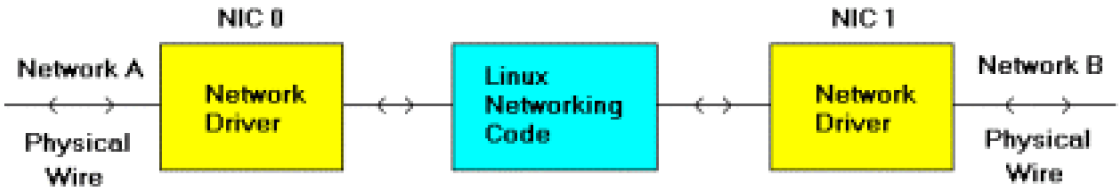


Figura. 11 Operación de Linux como un Router [13]

Se tiene entonces que control de tráfico es un conjunto de herramientas que permiten tener un control fino sobre las colas y los mecanismos de encolamiento de un dispositivo de red.

Teniendo un maquina con sistema operativo Linux operando como un router puro figura 11, es decir, realizando el reenvío de paquetes de una interface a otra, sin generar o consumir paquete alguno, entonces lo que ocurre es unos paquetes pueden estar entrando desde la red A por la interface de red NIC 0 allí el controlador (Network driver) se encarga de entregar los paquetes al código de red de Linux (Linux Networking Code) este realiza el reenvío de estos paquetes, solicitando a la interface de red NIC 1 en enviar estos paquetes a la red B.

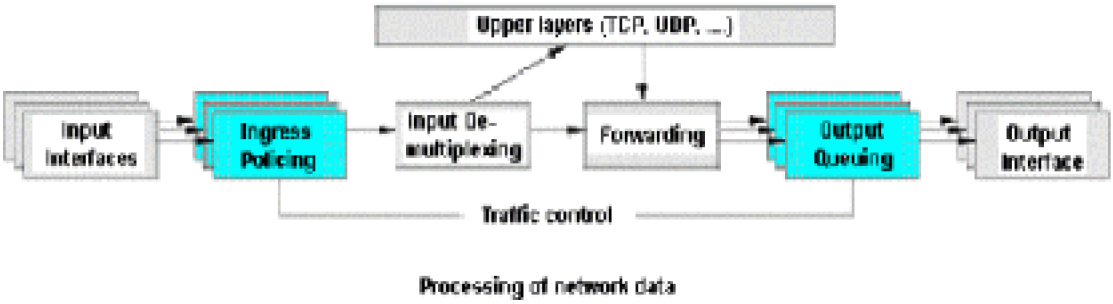


Figura 12. Control de tráfico con Linux [13]

En la figura 12 se puede a preciar que los paquetes que ingresan por las interfaces de entrada están sometido a lo que el kernel puede decidir mediante una regulación de ingreso (Ingress Policing), es decir, permitir su entrada o descártala debido a que están llegando demasiado rápido, después de ello interesa los que son reenviados a la cola de salida (Output Queuing) donde nuevamente el kernel decide a que cola deben ir los paquetes, por cual interface van a salir, si los paquetes serán encolados o no, como serán enviados los paquetes según un orden, si es posible retrasar su envío, después de que el kernel realiza su control los paquetes son enviados a las interfaces de salida. En resumen lo que se tiene son dos etapas donde se realiza el control del trafico por parte del kernel de Linux una en la entrada Ingress Policing y la otra a la salida con las colas Output Queueing.

Entonces a nivel de su estructura en lenguaje C, el kernel de Linux para realizar el control de tráfico se basa en los siguientes conceptos:

- Queueing disciplines
- Classes

- Filters y Policers

Mediante el empleo de las características del kernel de Linux para realizar control de tráfico, es posible llevar a cabo tareas tales como:

- Limitar el ancho de banda a un valor conocido.
- Limitar el ancho de banda de un usuario o servicio.
- Reservar ancho de banda para una aplicación en particular.
- Priorizar el traffic.
- Permitir equilibrar el uso del ancho de banda entre diferentes usuarios.
- Realizar balanceo de carga entre varios servidores.
- Rechazar cierto tráfico en particular.

Disciplinas de Colas: son algoritmos que se encargan de gestionar las colas en todo el proceso de ingreso (ingress) y salida (egress) de los paquetes sobre un dispositivo de red, estas son la base sobre la que se fundamenta la gestión de trafico en Linux, cada dispositivo de red tiene asociado un proceso de ingreso y de egreso.

Existen dos grupos de disciplinas de colas, unas con clases y las sin clases, aquellas con clases permiten al usuario crear subdivisiones para realizar diferenciación en el tratamiento del tráfico, mientras en que las disciplinas de colas sin clases esto no es posible.

Linux soporta varias disciplinas de colas y son las siguientes:

- CBQ: Class Based Queue
- TBF: Token Bucket Filter
- CSZ: Clark-Shenker-Zhang
- FIFO: First In First Out
- TEQL: Priority Traffic Equalizer
- SFQ: Stochastic Fair Queuing
- ATM: Asínchronous Transfer Mode
- RED: Random Early Detection
- GRED: Generalized Random Early Detection
- DSMARK: Diff-Serv Mark
- HTB: Hierarchical Token Bucket

Clases: Las disciplinas de colas que permiten definir clases (Classes) en su interior brindan al usuario la posibilidad de establecer configuraciones para diferentes tipos de tráfico a los cuales se les da tratamiento según la clase. Cada clase posee una cola que defecto es una tipo FIFO, cuando la función de encolamiento de la disciplina de cola es llamada, la disciplina de cola aplica los filtros para determinar a que clase pertenece el paquete.

Existen dos maneras mediante las cuales se puede identificar una clase, una es por medio de la identificación asignada por el usuario y la otra es interna y es realizada por kernel, esta última se conoce como identificador interno (internal ID) y es único, el otro se conoce como identificador de clase (class ID). El class ID es un dato tipo u32, mientras que el interno es un entero largo sin signo, la mayoría de las funciones sobre las clases usan el internal ID para identificar la clase, aunque existen funciones como get change que utilizan el class ID también. La estructura del class ID es major number : minor number, donde el major number corresponde a la instancia de la disciplina de clase y el minor number identifica la clase dentro de esa instancia.

Filtros: Para que Linux pueda realizar tareas de control de tráfico, cuenta en su kernel con utilidades que permiten tratar de manera diferente los paquetes, en este caso se explicara lo relacionado con los filtros (Filters) y los reguladores (Policers).

Los filtros están asociados a las disciplinas de colas pues es a través de estos que se define a que clase serán asignados los paquetes que ingresan, un filtro básicamente se usa para clasificar los paquetes basados en propiedades del campo TOS, la dirección IP, el número de puerto, etc.

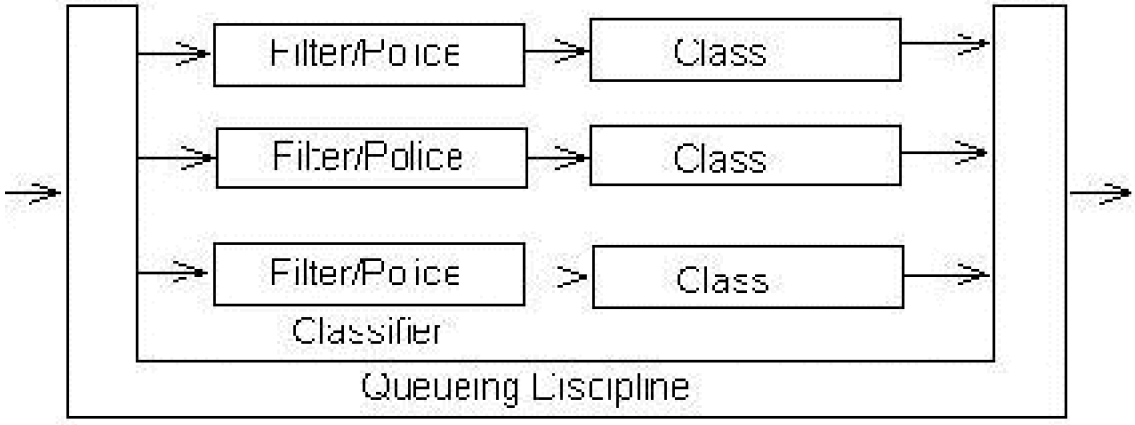


Figura 13. Modelo para control de tráfico del kernel de Linux [14]

De la figura 13 se puede apreciar como el kernel de Linux maneja el tráfico donde la disciplina de cola tiene asociada un filtro/regulador y unido a esto puede venir una clase.

Con los filtros se forman listas de estos que son mantenidos por la disciplina de cola o la clase, según sea el diseño de la disciplina de cola, estas listas son ordenadas por prioridades de forma ascendente, adicional a esto un filtro puede tener una estructura interna de elementos que son referenciados por un manejador (handle). Estos manejadores son identificados por un número de 32 bits, un filtro con manejador 0, se refiere al filtro en si.

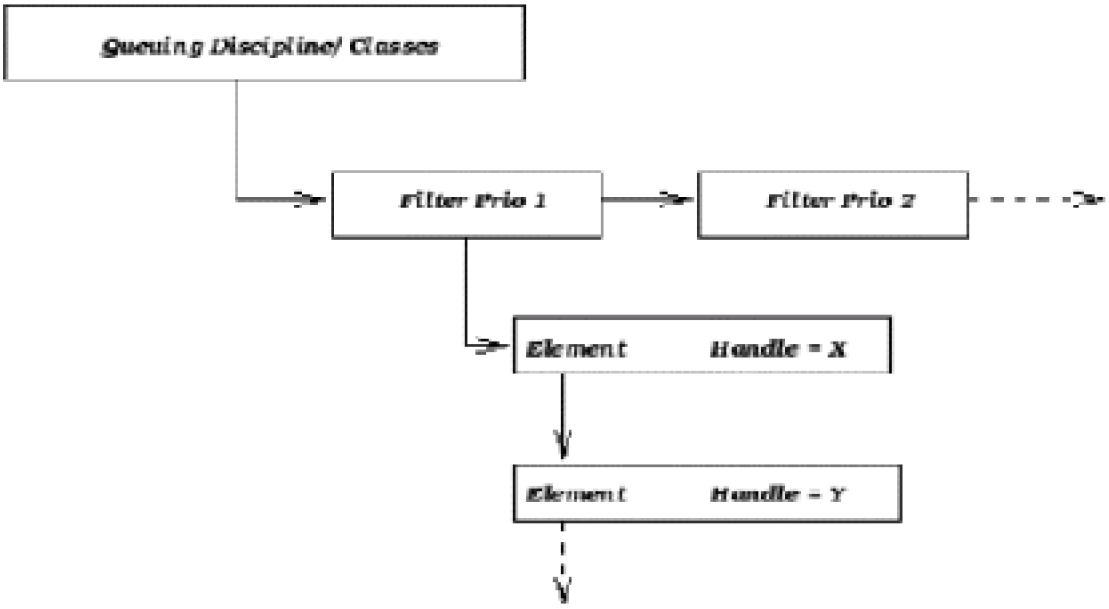


Figura 14. Estructura de los Filtros [14]

De la figura 14 se establece la estructura de filtros y como un filtro contiene elementos asociados con su respectivo manejador. El proceso se realiza como sigue, dentro de cada filtro los elementos son revisados con el objeto de clasificar el paquete, una vez clasificado el paquete este es encolado de acuerdo a la disciplina de cola y su clase asociada si hay alguna.

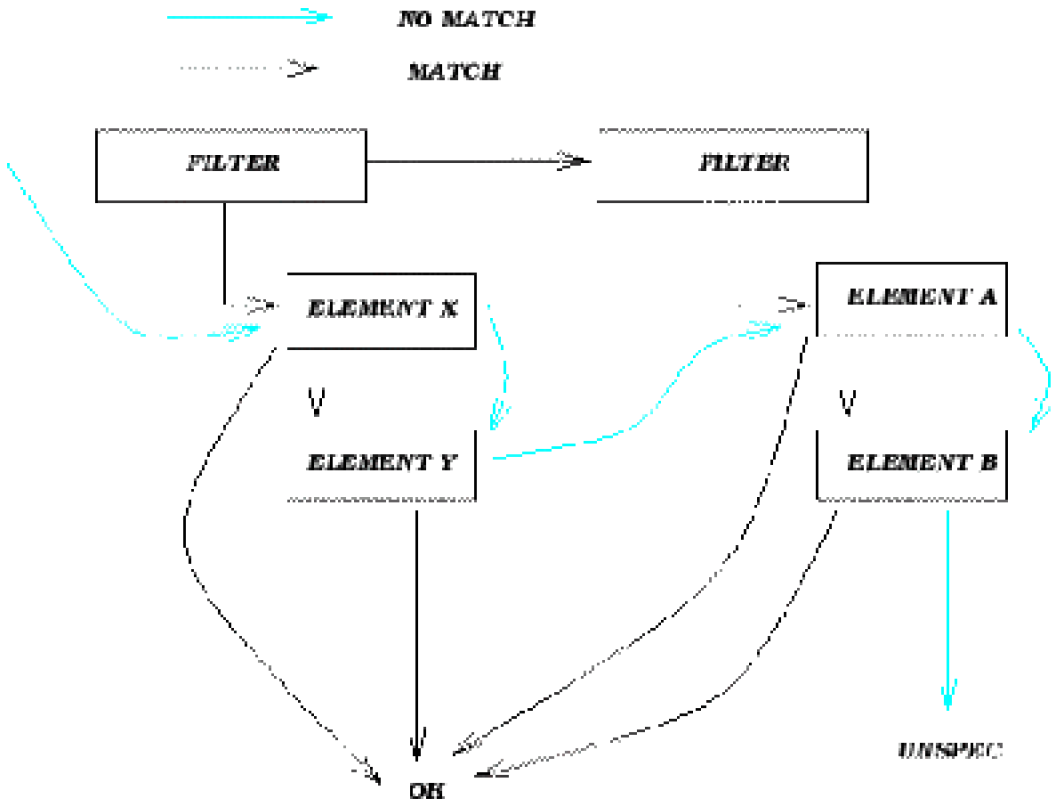


Figura 15. Proceso de selección en un Filtro [14]

Se tiene entonces que para identificar una disciplina de colas y una clase dentro de estructura de control de tráfico se requiere un único identificador el cual es conocido como manejador (handle) el cual esta constituido por dos partes un numero mayor (major number) y un numero menor (minor number), estos números son asignados por el usuario de manera arbitraria de acuerdo a la siguientes reglas:

major: este parámetro es completamente libre de significado para el kernel. Se puede usar un esquema arbitrario de numeración, sin embargo todos los objetos en la estructura de control de tráfico con el mismo parentesco deben compartir un major number.

minor: este parámetro identifica sin ambigüedad el objeto como una qdisc si tiene un valor de cero, cualquier otro valor identifica el objeto como una clase, todas las clases que comparten un pariente deben tener minor numbers únicos.

Un manejador (handle) especial es ffff:0 que esta reservado para el ingress, el manejador es usado como una etiqueta en classid y flowid de la utilidad tc cuando con esta se define un filtro, básicamente los manejadores (handles) son identificadores externos usados por el usuario, el kernel cuenta con identificadores internos para cada objeto.

Básicamente en Linux se definen dos tipos de filtros los genéricos y los específicos y su diferencia radica en el alcance que se le da a los paquetes según la instancia de clasificación.

Para los filtros genéricos se requiere una instancia del filtro por disciplina de cola

para clasificar los paquetes para todas las clases.

En el caso de los filtros específicos estos necesitan una o más instancias de un filtro o de sus elementos internos por clase para identificar la pertenencia de los paquetes a una clase

Regulación o Control : Cuando se habla de regulación o control (Policing), para efectuar control de tráfico, se refiere a regular el tráfico de los paquetes para que estos no excedan un límite.

En principio se consideran cuatro tipos de de mecanismos de regulación ocontrol:

- Decisiones de regulación por filtros
- No aceptar encolar un paquete
- Rechazo de un paquete desde una cola interior
- Rechazo de un paquete cuando se encola uno nuevo.

Cuando se hace referencia a decisiones de regulación por filtros, son estos últimos los que deciden que hacer con el paquete, ya sea, no realizar nada, reclasificarlo si este excede ciertos limites o simplemente descartarlo por que el paquete violo los limites.

En el caso de no aceptar encolar un paquete, básicamente se presenta cuando la disciplina de cola falla en encolar el paquete y este es descartado, algunas disciplinas de colas cuentan con mecanismos de realimentación que permiten volver a llamar la disciplina de cola, dando así una segunda oportunidad al paquete para que sea encolado.

En general la regulación busca medir y limitar el tráfico en una cola en particular, esta como elemento del control de tráfico se convierte en un mecanismo para limitarlo, la regulación es usada al extremo de la red para asegurar que no se exceda un ancho de banda asignado, el tema de regulación (Policing) de tráfico en Linux esta directamente relacionado con los filtros, pues son parte de estos.

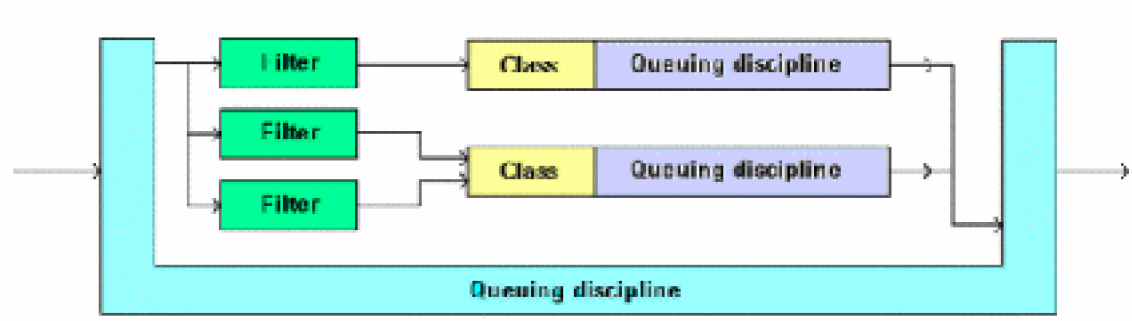


Figura 16. Manejo de disciplinas de colas con clases y filtros. [15]

La figura 16 resume los temas tratados hasta el momento, disciplina de colas, clases y filtros.

Linux dispone entonces de un conjunto de componentes para realizar el control de tráfico, que tienen su análogo en los elementos tradicionales de control de tráfico, a continuación se listan:

Tabla 2. Comparación entre control de tráfico tradicional y Linux [18]

Elementos Tradicionales	Componentes de Linux
Shaping (ajuste)	class, ofrece capacidades de ajuste.
Scheduling (ordenación o gestión)	qdisc es en si un scheduler (ordenador de los paquetes para salgan de cierta forma)
Classifying (clasificación)	filter, realiza la clasificación, pues esta es parte del filtro.
Policing (regulación o limitación)	policer hace parte de filter, como mecanismo de regulación.
Dropping (descarte)	filter con un policer, permite descartar el tráfico.
Parking (Marcado)	dsmark, mediante esta disciplina de colas se realiza el marcaje del paquete, es decir, puede ser alterado.

Disciplinas de colas sin clases. Básicamente solo pueden reordenar, retrasar o descartar el tráfico. Existe una disciplina de colas sin clases por defecto para Linux, la cual es utilizada por este si no se ha definido ninguna otra, la cual se llama pfifo_fast, esta básicamente es una FIFO, esta disciplina tiene tres bandas con prioridades 0, 1, 2 dentro de cada banda se aplica la regla FIFO, los paquetes que vengan con mayor prioridad van a la banda 0, los siguientes a la banda 1 y el resto a la banda.

Disciplinas de colas con clases. Este tipo de disciplinas brindar un mayor nivel de control y adquieren su valor cuando se trata de gestionar trafico con diferentes prioridades, pues permiten al usuario configurarlas para dar tratamiento a cada tipo de trafico. Los paquetes IP cuando llegan a un tipo de disciplina de cola como esta deben son enviados a unas clases que la componen, son entonces clasificados por medio de unos filtros que se encargan de decidir a que clases deben ser asignados los paquetes, es posible que el paquete deba ser sometido a nuevas clasificaciones (filtros) y clases hasta alcanzar su clasificación final.

IMPLEMENTACION BASICA DE CONTROL DE TRAFICO

Se desarrolla con un ejemplo práctico el control de tráfico sobre una red, usando una máquina con sistema operativo Linux como el dispositivo que realiza dicha labor de control. La metodología usada será la de una práctica tipo laboratorio en la cual, mediante un esquema sencillo de red, se evidencia el potencial existente en el área de control de tráfico usando Linux y las herramientas con que cuenta dicho sistema operativo.

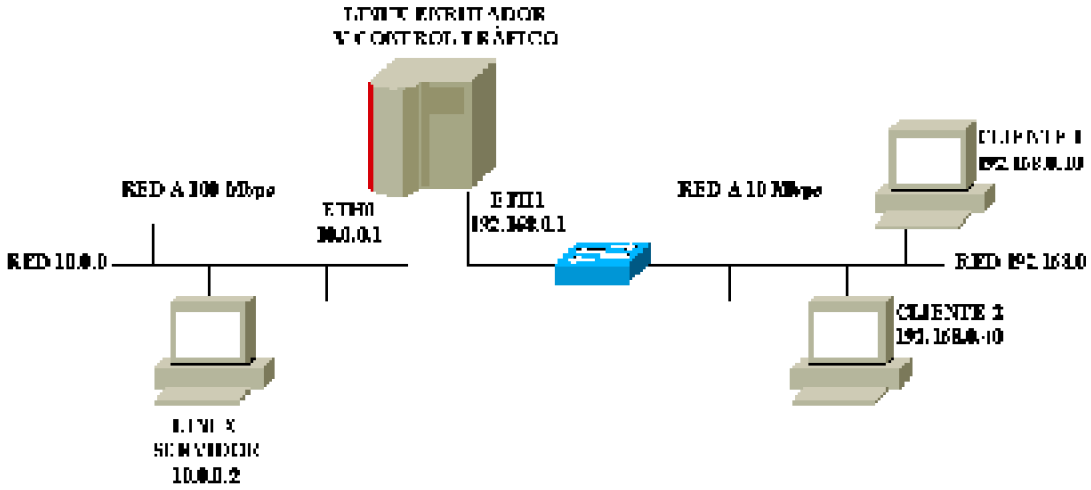


Figura 17. Esquema del montaje para el control de trafico

En la figura 17 se presenta el esquema de red utilizado: Se tienen dos segmentos de red diferentes, en uno de los cuales está ubicado un Servidor y en el otro se ubican clientes de los servicios implementados. Entre las dos redes se tiene una máquina que sirve de Gateway a ambas redes y que a su vez servirá como dispositivo de control de tráfico. Los resultados que se obtengan a partir de este esquema, a pesar de su sencillez, pueden hacerse extensivos a otro tipo de configuraciones, en donde se tiene un conjunto de clientes, accediendo a servicios ubicados en otra red (P.ej Internet), lo cual se ilustra en la figura 18..

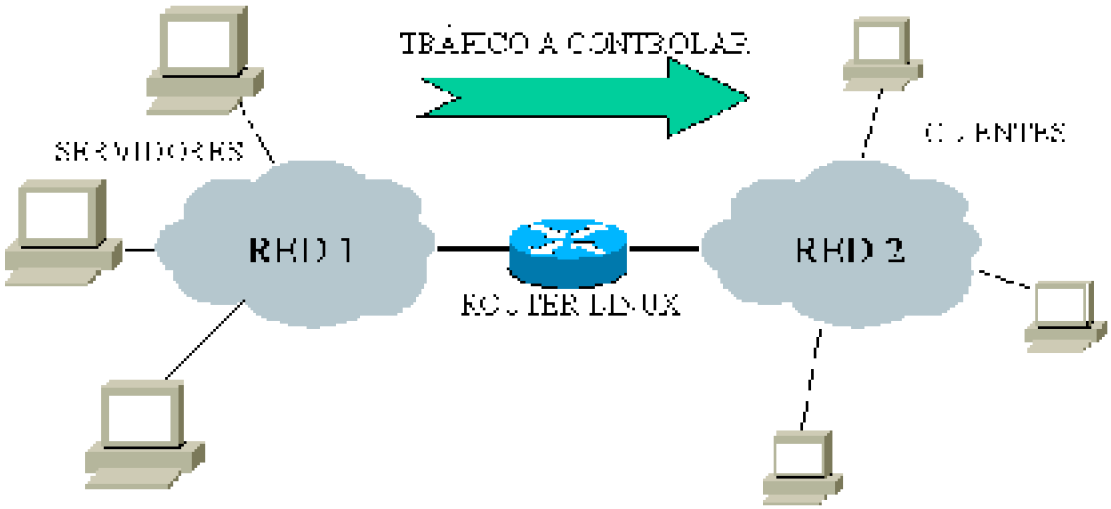


Figura 18 Esquema general para control de tráfico

Para la implementación que se muestra en la figura 17, se hará control de tráfico sobre la interfaz ETH1 de la máquina Linux, es decir, se controlará el tráfico que el servidor Linux en la red a 100 Mbps le envía a los clientes en la red a 10 Mbps. Esto de la siguiente manera, se controla el tráfico sobre la Interfaz ETH1, dependiendo del destino del mismo.

Descripción de los componentes de la red:

Red 10.0.0: Cable cruzado entre el servidor Linux y la interfaz ETH0 del Enrutador Linux.

Red 192.168.0:Swiche 10/100. La conexión del Enrutador Linux es a través de la interfaz ETH1 y es a 100 Mbps. La conexión de los clientes se hizo a 10 Mbps.

Servidor Linux:

Hardware:

Procesador Pentium IV 1.6 GHz

256 MB de RAM

Tarjeta de red integrada PRO/100 VE de Intel(R) (ETH0)

Software:

Sistema Operativo RED HAT LINUX 9.0 Versión Kernel 2.4.20-8.

Servidor WEB Apache: HTTPD versión 2.0.40-21

Servidor FTP: VSFTPD versión 1.1.3-8

Servidor SSH: OpenSSH versión 3.5p1.

NTOP: ntop-2.2-0.dag.rh90.

Cliente SNMP: net-snmp-5.0.6-17

Enrutador Linux:

Hardware.

Computador Desktop Procesador AMD Duron 1 GHz.

512 MB de RAM

Tarjeta de Red Integrada 10/100 SIS 900. (ETH0)

Tarjeta de Red 10/100 VIA Technologies. (ETH1)

Software:

Sistema Operativo RED HAT LINUX 9.0 Versión Kernel 2.4.20-8.

IPROUTE2: iproute2-ss010824.

NTOP: ntop-2.2-0.dag.rh90. Herramienta para S.O. LINUX que permite el monitoreo y análisis de tráfico.

Cliente SNMP: net-snmp-5.0.6-17

Cliente 1

Hardware

Computador Portátil con procesador Intel Pentium IV de 1.7 GHz.

256 MB RAM.

Tarjeta Integrada de Red 10/100 3Com 3C920.

Software:

Sistema Operativo Windows 2000 5.00.2195 Service Pack 4

Internet Explorer 6.0.2800.1106CO

WINFTP versión 2.52

SSH Secure Shell versión 3.2.9

Agente de captura SNMP.

WhatsUp GOLD para Windows. Versión 7.03

Cliente 2

Hardware

Computador Portátil con procesador.

MB RAM.

Tarjeta Integrada de Red.

Software:

Sistema Operativo

Internet Explorer

WINFTP versión 2.52

SSH Secure Shell versión 3.2.9

Agente de captura SNMP.

Switch 10/100

Switch nivel 2 de 8 puertos 10/100.

Descripción del software para toma de datos:

Las herramientas de software utilizado son NTOP, aplicación para Linux que permite mostrar el uso de la red, siendo capaz de listar los usuarios de la misma y reportar información concerniente al tráfico (IP y no IP) generado por cada usuario. Dicho tráfico es clasificado por NTOP de acuerdo al usuario y los protocolos.

Los protocolos que pueden ser monitoreados son:

- TCP/UDP/ICMP
- (R) ARP
- IPX
- DLC
- Decnet
- Apple Talk
- Netbios
- TCP/UDP

- FTP
- HTTP
- DNS
- Telnet
- SMTP/POP/IMAP
- SNMP
- NFS
- X11

Se requiere de un browser web para poder acceder a la información capturada por el NTOP, mientras este se esta ejecutando es posible que varios usuarios puedan acceder a la información de trafico, usando un browser web convencional, si se desea hacer un acceso seguro NTOP requiere de la librería openSSL.

La versión utilizada de NTOP, para este montaje es ntop-2.2-0.dag.rh90., en general se puede decir que NTOP es un sniffer con características que permiten visualizar las estadísticas de trafico mediante un navegador y es ideal para tareas donde se requiera conocer el grado de utilización de una red en periodos de monitoreo prolongados, debido a esto se hizo necesario el uso de una herramienta adicional que presentara gráfico en el tiempo del trafico total cursado por cada maquina en la red, dicha aplicación es la What's Up que a continuación se describe.

WhatsUp Gold: Sistema de monitoreo para Windows diseñado para redes multiprotocolo, el cual permite monitorear los dispositivos y servicios críticos en una red, generando alarmas visibles y/o audibles cuando se detecta un problema. Dos tipos de datos son guardados por la aplicación: Cambios en el estado de la red (llamados eventos), tales como que un dispositivo falló y estadísticas de poleo de cada dispositivo. Los eventos pueden filtrarse por dirección IP, nombre del dispositivo o descripción del evento y llevarse a gráficas estadísticas que muestran la disponibilidad del dispositivo.

También se cuenta con algunas características adicionales de monitoreo, tales como el monitoreo SNMP, el cual se usa para monitorear la actividad en un dispositivo y sus interfaces, lo que permite mostrar el tráfico en tiempo real.

SNMP (Simple Network Management Protocol) es un estándar de Internet que permite la gestión de datos en diferentes dispositivos de red para ser leídos y monitoreados por una aplicación. SNMP define un método mediante el cual un usuario remoto puede ver o cambiar la información de gestión de un dispositivo que está en red (Host, Gateway, Sevidor, etc). Una aplicación de gestión en un sistema remoto de usuario usa el protocolo para comunicarse con el agente SNMP en el dispositivo en el cual se quiere acceder a los datos. El agente SNMP en cada dispositivo puede proveer información acerca de la configuración y operación del dispositivo de red, tales como: las interfaces d eREd del dispositivo, tablas de enrutamiento, paquetes IP enviados, recibidos y perdidos, etc. Esta información, llamada Objetos SNMP es almacenada en un formato estándar definido en la Base de información de gestión (MIB: Management Information Base). En el MIB se definen los objetos SNMP que pueden ser gestionados y el formato

de cada objeto.

Se puede usar entonces WhatsUp Gold para ver y monitorear objetos SNMP en los dispositivos que tengan implementado el agente SNMP. Obviamente solo se podrán monitorear aquellos objetos que estén implementados en el agente SNMP remoto.

La versión utilizada para este montaje de WhatsUp Gold es la 7.03.

Con el fin de establecer un punto de referencia, se realizó un análisis de la red propuesta, en donde se monitoreó el comportamiento de la misma al generar tráfico sobre la red sin tener ninguna tarea de control de tráfico ejecutándose.

Para esto se utilizaron archivos residentes en el servidor Linux, cada uno con un tamaño de 638 Mb. Por medio de tres protocolos diferentes se hizo la transferencia de dichos archivos hacia cada uno de los clientes. Los protocolos usados fueron: HTTP, FTP y SSH, el procedimiento empleado consistió en observar el comportamiento de la red, cuando dos clientes estan haciendo uso de los tres protocolos.

Los datos obtenidos para los dos clientes al tiempo, haciendo uso de los tres protocolos:

Este es el estado “normal” de la red, donde se tienen todos los clientes accediendo a los diferentes recursos disponibles en los servidores. Ver figura 19. Se nota la distribución no equitativa del ancho de banda para los dos clientes, el tráfico promedio para cada uno es de 8.19 Mbps para el cliente 1 y de 1.40 Mbps para el Cliente 2, esta distribución hace que para la distribución global del tráfico por protocolo, prime la condición del cliente 1, es decir, se presente una distribución por protocolos con tendencia al equilibrio (Ver figura 20). En La figura 21 se observa la distribución por protocolos individual de cada Cliente, nótese que para el Cliente 2 se mantiene la distribución no uniforme entre los tres protocolos.

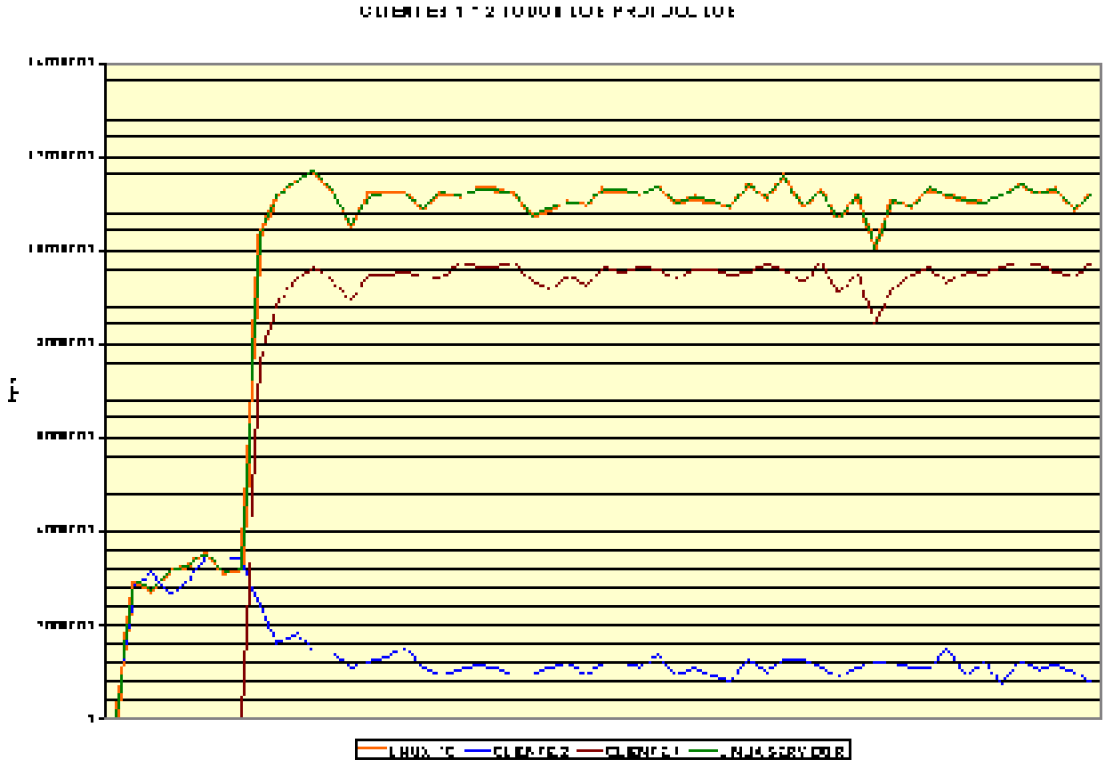


Figura 19 Tráfico en la red clientes 1 y 2 todos los protocolos whats UP

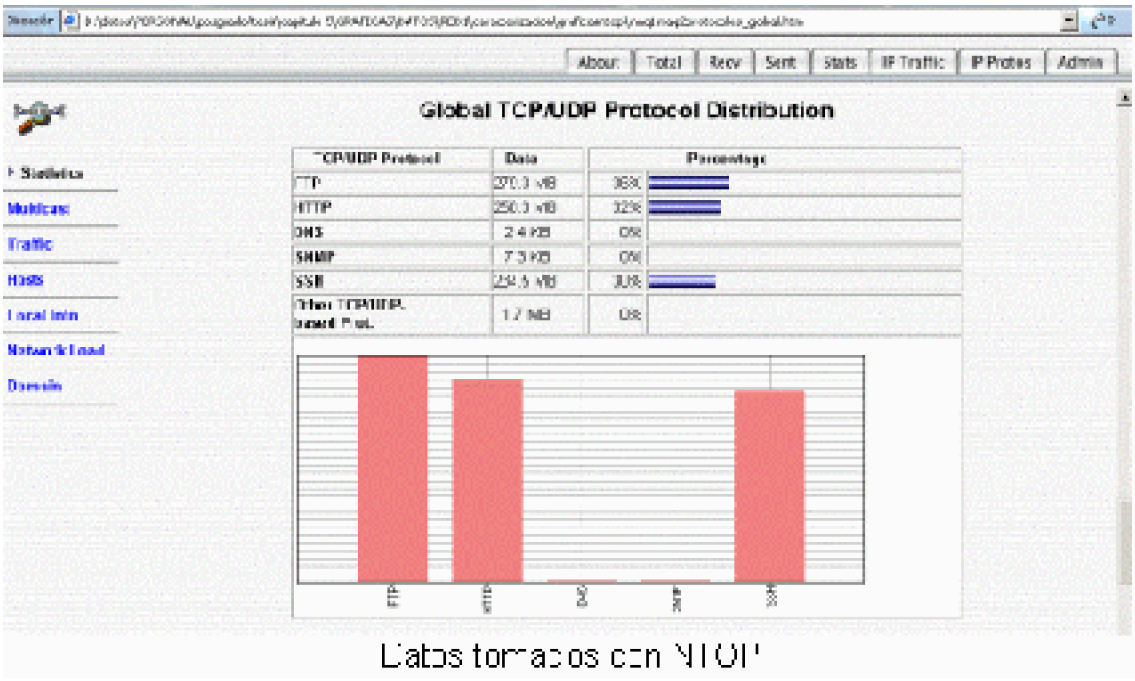


Figura 20 Distribución global de protocolos en la red, debido a los clientes 1 y 2

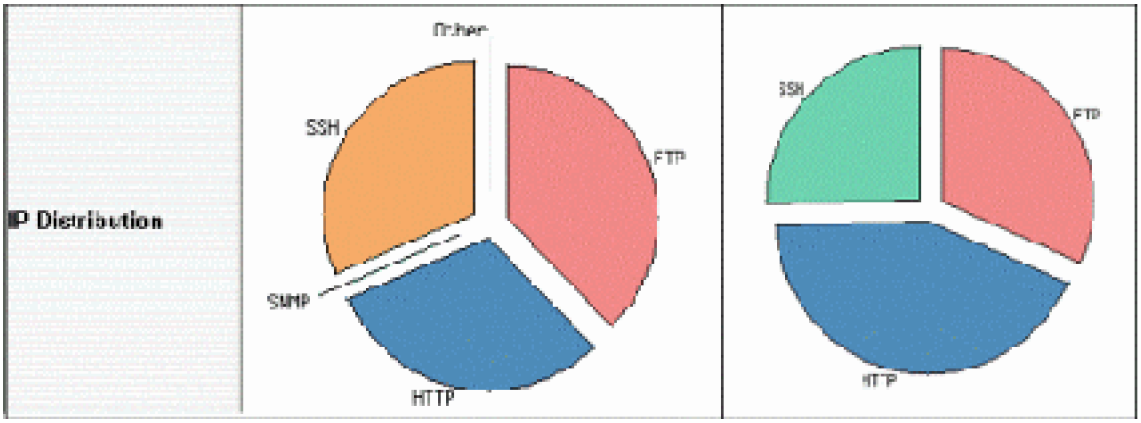


Figura 21. Distribución individual de protocolos en los clientes 1 y 2

Para controlar el trafico en la red de prueba, el control se basa en la disciplina de cola HTB, es decir, los scripts se realizan teniendo en cuenta loa parámetros definidos para realizar el control del ancho de banda al emplear este tipo de disciplina de cola, cada script consta de tres partes la primera define de la disciplina de cola raíz en este caso HTB, en la segunda se establecen las diferentes clases y finalmente se implementa el filtro.

Para realizar el control de tráfico por Dirección IP se tiene como punto de partida el script HTBBASICO, el cual se implementa para controlar el trafico por dirección IP, se requiere entonces de definir clases para lograr el objetivo, es decir, se define la disciplina de cola raíz que para toda esta implementación es la HTB, después se define una clase que permita luego construir clases hijas basadas en direcciones ip, para este caso se requiere de dos clases hijas una por cada dirección ip, pues se cuenta solo con dos clientes, a continuación se explica el script:

```
#!/bin/bash
#
# Este Script controla el ancho de banda asignado a dos maquinas
# diferentes accediendo al servidor
#
# Se realiza borrado de las disciplinas de colas que se encuentren
# configuradas, etapa egress e ingress
#
tc qdisc del dev eth1 root 2> /dev/null > /dev/null
tc qdisc del dev eth1 ingress 2> /dev/null > /dev/null
#
# Se define la disciplina HTB tipo egress para esta disciplina se
# establece una clase por defecto llamada clase 12, la cual maneja
# el otro trafico que no pasa por filtros
```

```
#
tc qdisc add dev eth1 root handle 1: htb default 12
#
# Se define la clase principal para la disciplina HTB, se limita el ancho
# de banda a 9Mbps
# Note que todo el comando esta definido en una sola línea
#
tc class add dev eth1 parent 1: classid 1:1 htb rate 9050kbit ceil 9050kbit
#
# Se definen las clases para cada maquina y se asignan los anchos de
# banda garantizado 8Mbps y máximo posible de 9Mbps Cliente 1
#
tc class add dev eth1 parent 1:1 classid 1:10 htb rate 8000kbit ceil 9000kbit
#
# Se definen las clases para cada maquina y se asignan los anchos de
# banda garantizado 1 Mbps y máximo posible 9 Mbps Cliente 2
#
tc class add dev eth1 parent 1:1 classid 1:11 htb rate 1000kbit ceil 9000kbit
#
# Se establece una clase por defecto para el resto de tráfico que no vaya
# para los clientes 1 y 2
#
tc class add dev eth1 parent 1:1 classid 1:12 htb rate 50kbit ceil 9000kbit
#
# Se asigna a cada una de las clases 10, 11 y 12, colas tipo sfq que son # del tipo de colas Justas con tiempo de reconfiguración cada 10
# segundos
#
tc qdisc add dev eth1 parent 1:10 handle 10: sfq perturb 10
tc qdisc add dev eth1 parent 1:11 handle 11: sfq perturb 10
tc qdisc add dev eth1 parent 1:12 handle 12: sfq perturb 10
#
# Se definen los filtros que permiten la escogencia de las maquinas a las # que se les controlara el ancho de banda, se trabaja nuevamente el
```

```
# filtro u32 pero esta vez sobre la dirección ip de las maquinas, para el
# Cliente 1 se le asigna la clase 10, Cliente 2 se le asigna la clase 11
# y la clase 12 para el del resto del trafico hacia la subred
# 192.168.0/24 en caso de que hubiesen otros clientes este seria su
# filtro y el ancho de banda será de 50 Kbps, garantizado
#
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.10 flowid 1:10
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0.40 flowid 1:11
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.0/24 flowid 1:12
#
# Fin del script
#
```

Con la figura 22 se puede resumir lo que se hace el script para el control de ancho de banda por dirección IP, se aprecia la definición de dos clases especificas una por cada cliente y una tercera asignada al trafico restante en la red, se debe tener en cuenta que cada clase final tiene asociado un filtro que para el caso es el filtro u32, aquí las clases finales son 1:10, 1:11 y 1:12.

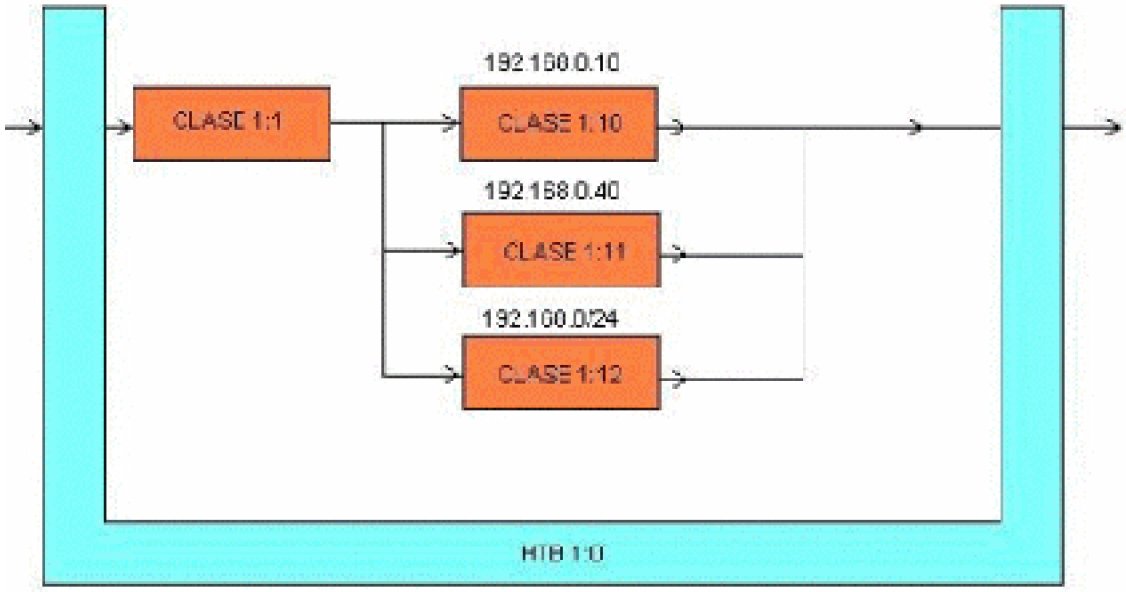


Figura 22. Representación grafica del script para control de trafico por dirección IP

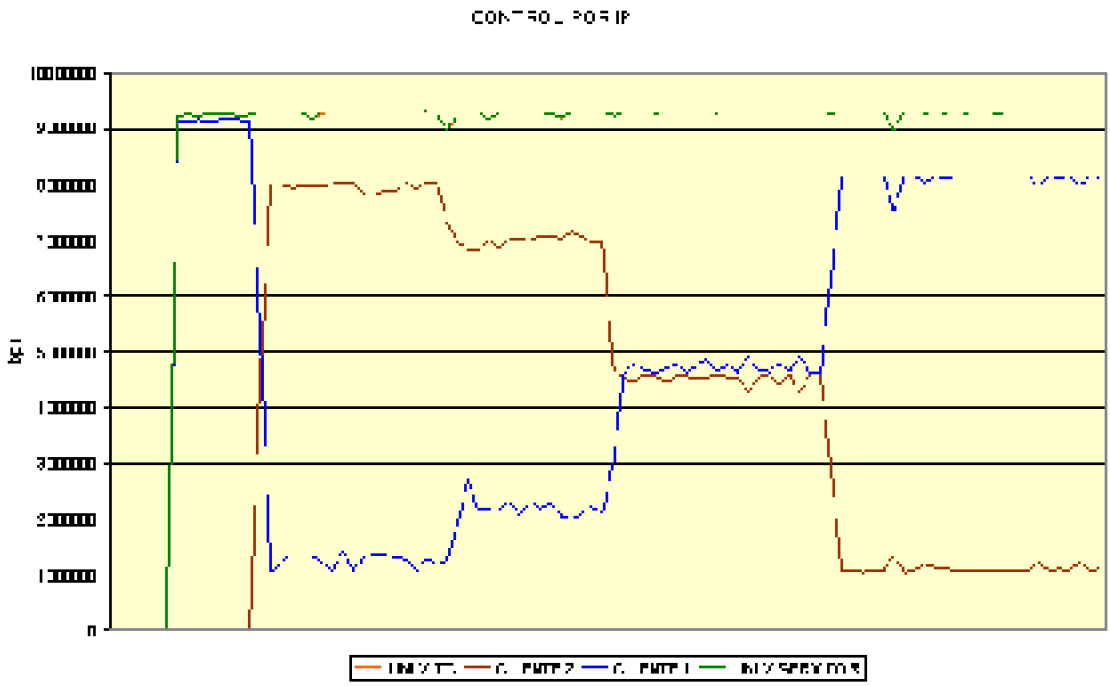


Figura 23. Control de traffic por direccion ip clientes 1 y 2

La figura 23 permite ver como se controla el ancho de banda para cada cliente, esto significa cambiar en el script el valor de rate, para el script presentado al Cliente 1 se le asigno un ancho de banda de 8Mbps, mientras el Cliente 2 cuenta con un ancho de banda de 1Mbps, en el caso de solo existir un Cliente en la red este puede hacer uso de todo el ancho de banda disponible hasta un máximo de 9Mbps, definido por el parámetro ceil.

Referencias

HARRIS Nick et al. Linux Handbook, A Guide to IBM Linux Solutions and Resources. Redbooks, 2003. <http://ibm.com/redbooks/sg247000> p. 1,9,10,12-45, 47-57, 59-69.

COOPER, Robert. Introduction to Queueing Theory. 2 ed. New York: North Holland, 1981 p. 1-2, 34-64.

GELENBE, E. ; PUJOLLE G. Introduction to Queueing Networks. Great Britain: John Wiley & Sons Ltd. 1987 p. 1-8, 10-11

VASTOLA, Kenneth. Performance Modeling and Analysis of Computer Communications Networks < http://networks.ecse.rpi.edu/~vastola/>

STALLINGS, William. Sistemas Operativos. 2 ed. España: Prentice Hall, 1997. p. 631-650.

DAIGLE, John. Queueing Theory for Telecommunications. United States of America : Addison-Wesley, 1992 p. 23-46, 214.

BROWN, Martin A. Linux Traffic Control HOWTO <http://www.tldp.org/HOWTO/Traffic-Control-HOWTO>.

MARSH, Mathew G. Policy Routing With Linux – On Line Edition <

http://www.policyrouting.org/PolicyRoutingBook>

ANDREASSON, Oskar. Iptables Tutorial 1.1.19 <http://iptables-tutorial.frozentux.net/iptables-tutorial.html>

] KUZNETSOV, Alexey N. IP Command Reference. 1999. < http://linux-ip.net/gl/ip-cref>

STANIC, Milan P. tc- traffic control – Linux QoS control tool. < http://www.rns-nis.co.yu/~mps/linux-tc.html >

Kernel Packet Traveling Diagram .<http://www.docum.or/stef.coene/qos/kptd>

RADHAKRISHAN, Saravanan. Linux – Advanced Networking Overview Version1 < http://qos.ittc.ukans.edu/howto>

ALMESBERGER, Werner. Linux Network Traffic Control-Implementation Overview . Abril 23 1999 <http://www.almesberger.net/cv/papers/tcio8 >

http://www.opalsoft.net/qos

HUBERT, Bert et al. Linux Advanced Routing & Traffic Control HOWTO <http://www.linuxdoc.org/HOWTO/Adv-Routing-HOWTO.html>