
GESTOR DE ANCHO DE BANDA PARA GENERAL LAB



Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Autor:	Alejandro Esteban Prados
Director:	Edgar Navarro Andrés
Ponente:	Anna Calveras Augé
Fecha:	Marzo 2009

AGRADECIMIENTOS

Este proyecto no se hubiera podido llevar a cabo sin la colaboración y la ayuda de un gran número de personas.

En primer lugar quiero dar las gracias a mi director Edgar Navarro, por confiar en mí y darme la oportunidad de hacer este proyecto. También agradecer a mi ponente Anna Calveras, por guiarme y aconsejarme durante todos estos meses.

Gracias también a todo el personal de General Lab, y en especial al departamento de sistemas, por su apoyo y por todas las horas que les he robado. Gracias Alberto, Marçal y Enric por haberme hecho sentir uno más entre vosotros.

Gracias a toda mi familia (y a los que están de camino) por ese apoyo incondicional que siempre me dan, y gracias a mis amigos por acompañarme en esta etapa.

No quiero acabar estas líneas sin agradecer a la persona que más me ha ayudado, apoyado, comprendido y enseñado en los últimos años. Gracias Belén.

CONTENIDOS

1.- INTRODUCCIÓN.....	1
1.1.- LA EMPRESA.....	1
1.1.1.- <i>General Lab</i>	1
1.1.2.- <i>Estructura de red</i>	2
1.1.3.- <i>Proyectos internos de General Lab</i>	4
1.2.- MOTIVACIÓN.....	7
1.3.- OBJETIVOS.....	8
2.- CALIDAD DE SERVICIO QOS.....	10
2.1.- INTRODUCCIÓN.....	10
2.1.1.- <i>Definición</i>	10
2.1.2.- <i>Parámetros</i>	10
2.2.- MODELOS DE QoS	11
2.2.1.- <i>Servicios Integrados</i>	11
2.2.2.- <i>Protocolo RSVP</i>	12
2.2.3.- <i>Servicios Diferenciados</i>	13
2.2.4.- <i>Protocolo MPLS</i>	15
2.3.- CONCLUSIONES.....	16
3.- CONTROL DE TRÁFICO EN LINUX	17
3.1.- INTRODUCCIÓN.....	17
3.2.- ELEMENTOS DE CONTROL DE TRÁFICO.....	18
3.2.1.- <i>Elementos genéricos</i>	18
3.2.2.- <i>Herramientas de control de tráfico en Linux</i>	19
3.3.- DISCIPLINA DE COLAS SIMPLES	21
3.3.1.- <i>First In First Out</i>	21
3.3.2.- <i>PFIFO_fast</i>	22
3.3.3.- <i>Stochastic Fairness Queueing</i>	23
3.3.4.- <i>Extended Stochastic Fair Queuing</i>	23
3.3.5.- <i>Token Bucket Filter</i>	24
3.3.6.- <i>Random Early Detection</i>	25
3.4.- DISCIPLINA DE COLAS CON CLASES	26
3.4.1.- <i>PRIO</i>	26
3.4.2.- <i>DSMARK</i>	27
3.4.3.- <i>Hierarchical Token Bucket</i>	28
3.5.- CORTAFUEGOS EN LINUX	33
3.5.1.- <i>Definición</i>	33
3.5.2.- <i>Funcionamiento</i>	33
3.5.3.- <i>Opciones y parámetros</i>	34
3.6.- ELECCIÓN DE LA DISCIPLINA DE COLAS	35

4.- ANÁLISIS DE POSIBLES HERRAMIENTAS	37
4.1.- REQUISITOS.....	37
4.2.- ANÁLISIS DE HERRAMIENTAS	37
4.2.1.- Soluciones en el mercado	37
4.2.2.- Creación de una aplicación personalizada.....	39
4.2.3.- Herramientas basadas en scripts	40
4.2.4.- Herramientas con interfaz Web	43
4.3.- ELECCIÓN DE LA HERRAMIENTA.....	45
5.- CONFIGURACIÓN DEL SISTEMA.....	46
5.1.- PREPARACIÓN DE LA MÁQUINA	46
5.2.- ELECCIÓN DEL SISTEMA OPERATIVO	46
5.2.1.- Análisis de sistemas operativos.....	46
5.2.2.- Conclusiones respecto al sistema operativo	48
5.3.- INSTALACIÓN Y CONFIGURACIÓN DEL SISTEMA OPERATIVO	48
5.4.- MANTENIMIENTO DEL SISTEMA	49
6.- LA APLICACIÓN <i>WEBHTB</i>.....	50
6.1.- PROCESO DE INSTALACIÓN DE <i>WEBHTB</i>	50
6.2.- FUNCIONAMIENTO DE <i>WEBHTB</i>	50
6.2.1.- Servicios adicionales necesarios.....	50
6.2.2.- Creación de los elementos HTB	51
6.2.3.- Parámetros HTB.....	53
6.2.4.- Visualización del tráfico.....	55
6.2.5.- Contribución al desarrollo de <i>WebHTB</i>	56
6.2.6.- Soluciones alternativas con el uso de iptables.....	57
7.- PROCESO DE PRUEBAS Y TESTING.....	58
7.1.- PRUEBAS EN ENTORNOS BÁSICOS SOBRE ETHERNET	58
7.1.1.- Red de pruebas básica sobre Ethernet	58
7.1.2.- Resultados con configuración básica sobre Ethernet	61
7.2.- PRUEBAS EN ENTORNOS REALES SOBRE ADSL	63
7.2.1.- Gestión en origen de un enlace ADSL	63
7.2.2.- Comportamiento del sistema con gestión en origen	64
7.2.3.- Gestión en destino de un enlace ADSL	69
7.2.4.- Control de flujo	69
7.2.5.- Comportamiento del sistema con gestión en destino	71
7.3.- ANÁLISIS DE COLAS FINALES.....	71
7.3.1.- Según latencia	72
7.3.2.- Según protocolo.....	73
7.3.3.- Conclusiones	74
7.4.- CONCLUSIONES DEL PROCESO DE PRUEBAS	75

8.- CONFIGURACIÓN PARA GENERAL LAB	76
8.1.- CONFIGURACIÓN DE LA RED DE LA EMPRESA	76
8.1.1.- Enlace a gestionar.....	76
8.1.2.- Servicios a gestionar	76
8.1.3.- Ubicación física del servidor de gestión.....	77
8.1.4.- Aclaraciones sobre la red a gestionar.....	78
8.1.5.- Modificaciones en la red debido al sistema de gestión.....	79
8.2.- CREACIÓN DE LOS ELEMENTOS DE CONTROL CON <i>WEBHTB</i>	80
8.2.1.- Creación de las clases.....	80
8.2.2.- Configuración de los filtros.....	82
8.2.3.- Comprobación de los elementos de control de tráfico creados	82
8.3.- RESULTADOS Y ANÁLISIS DEL SISTEMA EN PRODUCCIÓN	84
9.- ANÁLISIS DEL PROYECTO.....	86
9.1.- PLANIFICACIÓN DEL PROYECTO	86
9.2.- COSTE DEL PROYECTO	87
9.3.- EVALUACIÓN DE OBJETIVOS	88
10.- CONCLUSIONES Y LINEAS DE FUTURO	90
10.1.-CONCLUSIONES	90
10.2.- LINEAS DE FUTURO	91
10.2.1.- Migración del sistema operativo.....	91
10.2.2.- Configuración del sistema en alta disponibilidad	91
10.2.3.- Migración a nuevas versiones de <i>WebHTB</i>	92
10.2.4.- Análisis sobre la incorporación al sistema de <i>NAPI</i>	92
10.2.5.- Integración en el sistema de copias de seguridad de la empresa	92

ANEXOS	93
A.- INSTALACIÓN Y CONFIGURACIÓN DEL SISTEMA OPERATIVO.....	95
<i>A.1.- Instalación de CentOS</i>	95
<i>A.2.- Compilación del kernel</i>	96
B.- PERSONALIZACIÓN DEL SISTEMA	102
<i>B.1.- Gestión de usuarios</i>	102
<i>B.2.- Gestión de servicios</i>	104
<i>B.3.- Configuración de tablas de rutas</i>	106
<i>B.4.- Instalación de herramientas de utilidad</i>	107
C.- MANTENIMIENTO DEL SISTEMA.....	111
<i>C.1.- Copia de seguridad</i>	111
<i>C.2.- Control del sistema</i>	111
<i>C.3.- Logs del sistema</i>	112
<i>C.4.- Reinicio del sistema</i>	113
D.- MANUAL DE INSTALACIÓN DE <i>WEBHTB</i>	114
<i>D.1.- Instalación de la aplicación WebHTB</i>	114
<i>D.2.- Identificación y resolución de incidencias en el proceso de instalación</i>	117
E.- MANUAL DE USUARIO DE <i>WEBHTB</i>	120
<i>E.1.- Añadir interfaces</i>	120
<i>E.2.- Añadir clase</i>	121
<i>E.3.- Añadir cliente</i>	122
<i>E.4.- Aplicar cambios</i>	123
<i>E.5.- Editar clases</i>	124
<i>E.6.- Configuración remota</i>	124
<i>E.7.- Mostrar tráfico</i>	125
<i>E.8.- Detener el servicio</i>	125
<i>E.9.- Consejos para el uso de la aplicación</i>	126
F.- ELEMENTOS DE CONTROL DE TRÁFICO DEL GESTOR EN PRODUCCIÓN.....	127
G.- FORMACIÓN PARA EL PERSONAL DE GENERAL LAB	130
H.- BIBLIOGRAFÍA Y REFERENCIAS	131
I.- ÍNDICE DE FIGURAS	134

1.- INTRODUCCIÓN

1.1.- LA EMPRESA

1.1.1.- General Lab

General Lab es una empresa de servicios dedicada a la realización de análisis clínicos y biológicos. Está integrada por un grupo de laboratorios líder, en España y Portugal, en servicios, información, gestión y realización de pruebas diagnósticas de laboratorio. Su sede social está en Barcelona, en la calle Londres 28, y tiene instaladas sus oficinas de administración, para todo el grupo, en la avenida Josep Tarradellas 123-127.

La empresa en cifras en el año 2008:

- 45 laboratorios clínicos propios
- 7 empresas participadas accionarialmente.
- 28 laboratorios colaboradores.
- 740 empleados.
- 3.069.093 peticiones o solicitudes analíticas (ejercicio 2007) (Figura 1.1).
- 13.500 peticiones o solicitudes analíticas por día.
- 30.442.115 tests o análisis anuales (ejercicio 2007).
- 72.000.000 € de facturación agregada estimada para el ejercicio 2008.

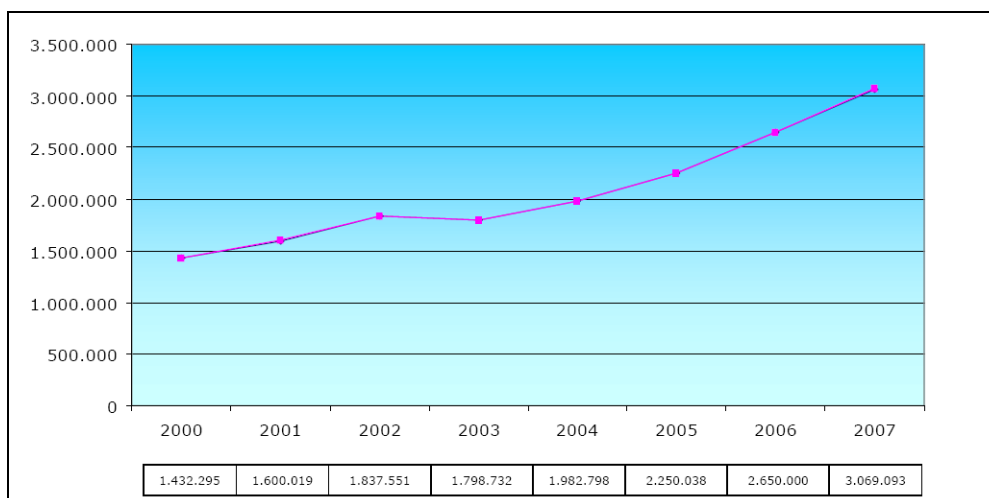


Figura 1.1.- Evolución de la actividad de la empresa en peticiones [42]

Durante el transcurso del ejercicio 2007, General Lab se fusionó con el grupo líder en Europa Labco. Labco es un grupo líder de diagnóstico médico basado en una exclusiva red europea de laboratorios clínicos locales y regionales.

La red se compone en la actualidad de una plantilla de unas 4000 personas, distribuidas en más de 250 laboratorios en 6 países europeos. Labco ofrece una variedad de 2500 ensayos clínicos de confianza y trata a más de 8 millones de pacientes por año (Figura 1.2).

A raíz de esta fusión, General Lab pasó a formar parte de Labco dentro del marco Labco Iberia, consiguiendo ser aún más líderes dentro del territorio español y portugués.

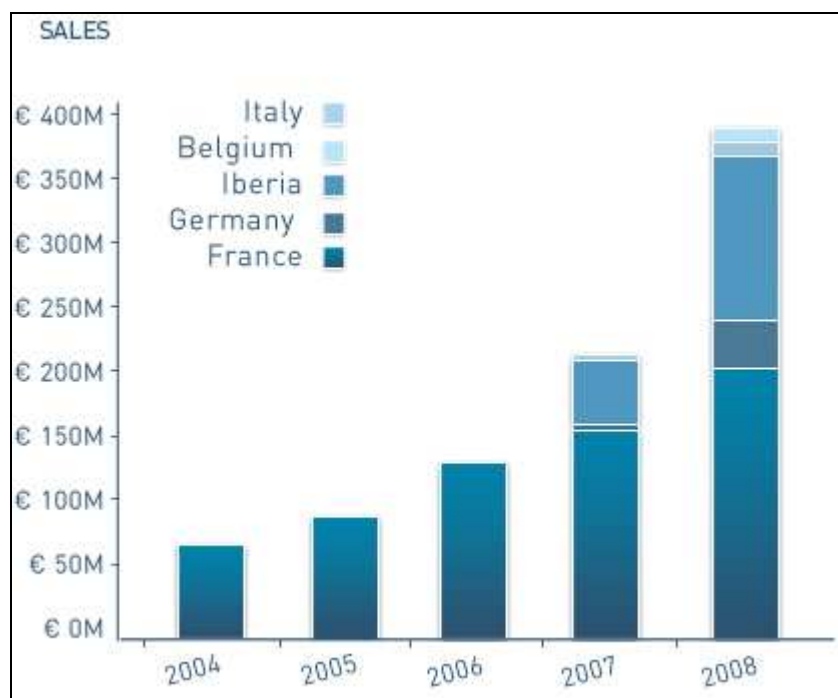


Figura 1.2.- Evolución de la facturación del grupo Labco [43]

1.1.2.- Estructura de red

Todos los centros o laboratorios que forman el grupo General Lab están estandarizados con el fin de facilitar su gestión, por lo que deben seguir algunas pautas.

En lo que se refiere a los rangos de direccionamiento IP, cada centro o laboratorio debe estar dentro del rango 128.0.0.0/8. El primer octeto es por estar dentro de la red privada de la empresa. El segundo octeto se corresponde con el número ordinal del centro de la provincia, asignado por orden de apertura. Y el tercer octeto se corresponde con los dos primeros dígitos del código postal de la provincia en la que se encuentra.

Por ejemplo, en la ciudad de Madrid existen actualmente 5 centros prestando servicio. Si se abriera otro laboratorio más, debería tener asignado el siguiente rango IP:

- Por ser un laboratorio más de la red de General Lab debería empezar con el octeto 128, de la forma 128.0.0.0/8.
- Por ser el sexto laboratorio de la ciudad, el segundo octeto debería ser el 5, quedando el rango 128.5.0.0/16.
- Por último, el tercer octeto debería ser el 28, ya que los diferentes códigos postales de Madrid empiezan todos por 28XXX, quedando finalmente el rango 128.5.28.0/24.

En cada uno de estos centros hay instalados dos servidores idénticos. El primero de ellos, identificado por la letra A y la primera dirección IP del rango asignado, es el que se encuentra en producción. El segundo, identificado por la letra B y con la segunda dirección IP, es el servidor de *backup*, que solo estará en producción en caso de caída del servidor principal. Estos servidores también actúan como servidores de impresión, de manera que siempre que sea posible se reservará el rango de direcciones IP 128.X.X.180-128.X.X.199 para impresoras.

Todos los laboratorios propios se conectan a la sede central mediante una NetLan o bien una MacroLan. Para ambos casos, se utilizan dos routers, uno encriptador (128.X.X.207) que se encarga de abrir el túnel contra otro router situado en la central, y otro que actúa simplemente como router de acceso (128.X.X.206). El segundo de ellos dispone de dos interfaces de red, uno conectado a una conexión ADSL y otro conectado a una RDSI que se utilizará solo en caso de caída de la conexión ADSL. El resto de laboratorios colaboradores se conectan mediante conexiones VPN.

Por otro lado, existen casos donde es necesaria una conexión permanente entre los laboratorios de General Lab y los hospitales o clínicas donde está prestando sus servicios. En este caso se hará uso de un router con dirección IP 128.X.X.191 para conectar la red del laboratorio con la del hospital.

Finalmente, el rango de direcciones reservado para ordenadores de trabajo es el comprendido entre 128.X.X.31–128.X.X.180. Se reserva el margen 128.X.X.1–128.X.X.30 para servidores y otro tipo de equipos no estándar, como por ejemplo dispensadores de tickets automático.

En la figura 1.3 se observa el mapa general de comunicaciones. En este esquema se detallan todos los elementos de red que forman parte de la empresa, así como la distribución y los rangos IP de todos los centros y laboratorios. Todos los servidores, routers y *firewalls* que aparecen en la parte inferior del esquema, están físicamente ubicados en la sede central en Barcelona.

1.1.3.- Proyectos internos de General Lab

Actualmente, General Lab está llevando a cabo distintos proyectos internos con el fin de mejorar el servicio prestado a los clientes y conseguir un crecimiento sostenido. Todos estos proyectos se están desplegando de manera gradual, y consisten en la migración de la aplicación de negocio de gestión de laboratorios, el despliegue de un sistema de planificación de recursos ERP y una migración de la red NetLan a una MacroLan para centralizar las comunicaciones.

1.1.3.1.- Migración de la aplicación de negocio de gestión de laboratorios

Para coordinar el servicio prestado por los distintos laboratorios, General Lab dispone de distintas aplicaciones internas de negocio. Uno de los proyectos que se están llevando a cabo actualmente es la migración de todas estas aplicaciones a una nueva herramienta llamada WinLabNet, que permite facilitar la gestión de los distintos centros y englobar todas las antiguas aplicaciones en una sola. Esta nueva aplicación presenta más prestaciones y está desarrollada en entorno gráfico, y en consecuencia, requiere mayor capacidad del ancho de banda.

Otra diferencia importante es que, a diferencia de las anteriores aplicaciones de negocio, los servidores que soportan WinLabNet se encuentran ubicados en la sede central de Barcelona. Esto hace que todo el tráfico referente a esta aplicación deben transmitirse a través de las redes internas de la empresa, lo que supone un aumento considerable del tráfico en estas redes.

1.1.3.2.- Despliegue de un sistema de planificación ERP

Otro de los proyectos que implica un aumento de tráfico es el despliegue de un sistema de planificación ERP. Este sistema pretende facilitar las tareas de logística de la empresa, y de la misma manera que la anterior aplicación, también tiene sus servidores centralizados en Barcelona.

1.1.3.3.- Migración a una MacroLan y centralización de las comunicaciones

Debido al aumento de tráfico propiciado por el despliegue progresivo de los proyectos comentados anteriormente, las líneas de comunicaciones de la red NetLan que se utilizaban para conectar los distintos centros había llegado a situaciones críticas. La solución que la empresa ha utilizado para conseguir unas comunicaciones fluidas ha sido empezar una migración progresiva de la red NetLan a una red MacroLan con mayores prestaciones.

La red NetLan implantada en la empresa es una solución de Telefónica que se basa en conexiones ADSL asimétricas en cada uno de los centros para conectar con la sede central. Estas conexiones pueden variar su capacidad según los requisitos de cada centro, y la conexión ADSL de la sede central cuenta con un CIR de 2Mbps tanto de subida como de bajada para canalizar todo el tráfico centralizado.

La nueva red MacroLan de Telefónica utiliza fibra óptica en cada centro, lo que supone una mejora considerable en cuanto a latencia. Estas nuevas conexiones son simétricas y de mayor capacidad, siendo la conexión de la sede central de 10Mbps. Además, cuenta con un mecanismo de transporte MPLS con la posibilidad de configurar Calidad de Servicio.

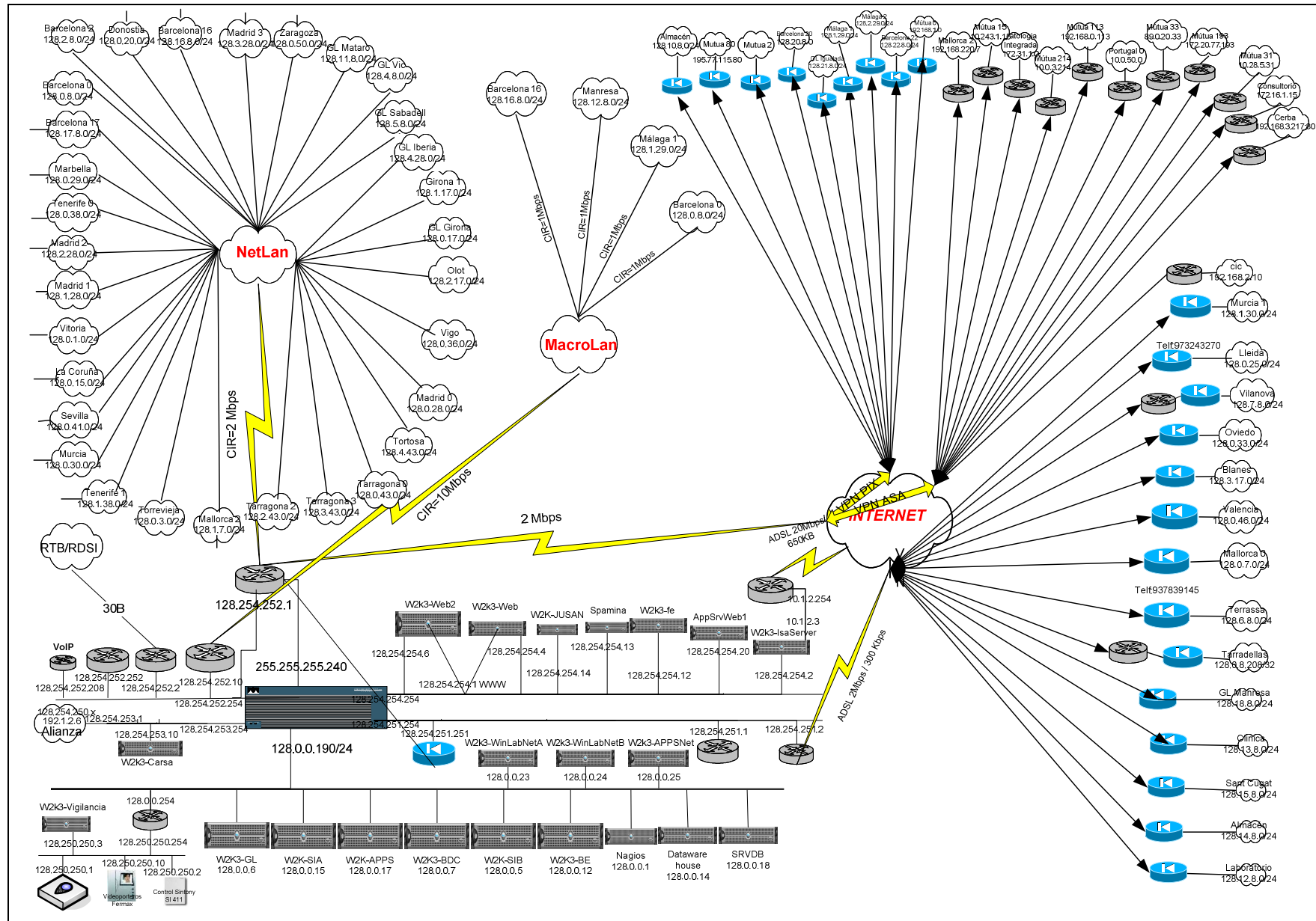


Figura 1.3.- Esquema general de comunicaciones de General Lab.

1.2.- MOTIVACIÓN

A causa de los distintos proyectos internos que está llevando a cabo la empresa, la cantidad de tráfico que se debe transportar a través de la red ha sufrido un aumento importante. Además, en los últimos años General Lab ha experimentado un crecimiento considerable, lo que implica un mayor número de ordenadores y de empleados, lo que da como resultado otro fuerte incremento del tráfico.

La red NetLan que comunica los distintos centros no es capaz de soportar el incremento del caudal y ha quedado en situación crítica, por lo que se ha empezado una migración y centralización de las comunicaciones para consolidar una base de datos centralizada de mayores prestaciones.

Actualmente, el tráfico de negocio no presenta ningún tipo de prioridad respecto el resto de tráfico, teniendo que compartir el enlace de manera equitativa con el resto de servicios (correo, Internet, Web interna, etc.). Esto hace menos eficiente el uso del ancho de banda, lo que hace necesaria una solución eficiente que permita optimizar los recursos de la empresa.

En vistas a que la centralización de las comunicaciones, la migración de la aplicación de negocio y el despliegue de ERP sirvan para conseguir un crecimiento sostenido, surge la necesidad de dotar a la red MacroLan de Calidad de Servicio (QoS). Aunque la red MacroLan se basa en el protocolo MPLS que permite QoS, se prefiere que la gestión del tráfico se haga por parte de la empresas y no del proveedor de servicios de red.

Esta calidad debe centrarse en proveer a la red de un sistema capaz de controlar y gestionar el ancho de banda disponible, diferenciando entre flujos de tráfico y priorizando el intercambio de tráfico de negocio frente el resto, con el fin de optimizar el uso del ancho de banda disponible.

1.3.- OBJETIVOS

El objetivo principal del proyecto es dotar a la empresa de un sistema personalizado que dote a la red MacroLan de QoS (Calidad de Servicio) y que gestione el ancho de banda disponible.

Para llegar a este objetivo final, es necesario desarrollar los siguientes puntos:

1. Obtención de conocimientos teóricos sobre gestión de ancho de banda en entorno GNU/Linux.
2. Análisis de distintas herramientas disponibles para gestionar anchos de banda.
3. Instalación y configuración de la aplicación elegida.
4. Facilitar a la empresa de un manual de instalación y de la documentación necesaria sobre el uso y el mantenimiento del sistema.
5. Análisis y ejecución de un procedimiento para la puesta en producción del servicio.

Para conseguir estos objetivos de una manera eficiente, se hace una estimación de los recursos y del tiempo necesario para llevar a cabo el proyecto. Esta planificación se puede ver en forma de diagrama de Gantt en la figura 1.4.

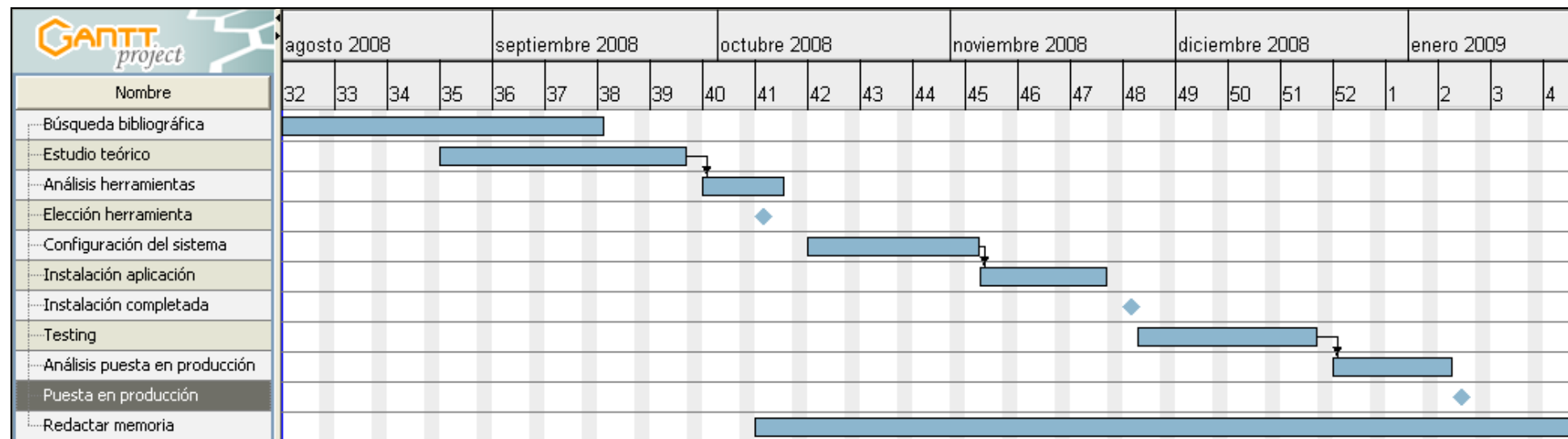


Figura 1.4.- Diagrama de Gantt de planificación del proyecto

2.- CALIDAD DE SERVICIO QoS

2.1.- INTRODUCCIÓN

2.1.1.- Definición

Para cumplir con las necesidades de la empresa es necesario dotar a la red de calidad de servicio extremo a extremo. Este término consiste en la capacidad de la red para reservar algunos de los recursos disponibles para un tráfico concreto con la intención de proporcionar un determinado servicio. Es posible conseguir QoS en todas las partes o tramos de la red, ya sea dentro de la red local (LAN) o en la red proporcionada por el proveedor (WAN).

Existen diferentes clasificaciones para definir los parámetros que conforman la calidad del servicio. Una clasificación posible podría ser la que tiene en cuenta la disponibilidad, la latencia, el *jitter*, la pérdida de paquetes y el ancho de banda.

2.1.2.- Parámetros

Disponibilidad: Porcentaje de tiempo que la red es capaz de realizar las funciones requeridas, es decir, el tiempo en el que la red está activa y en funcionamiento.

Latencia: Retardo entre el envío de paquetes y su recepción, pero desde un punto de vista del usuario se definiría como el vacío en la conversación producido por retardos acumulados durante la transferencia de paquetes y los retardos de procesamiento.

Jitter: Variación de los retardos en la llegada de los paquetes entre su origen y el destino, usualmente producida por congestión de tráfico en algún punto de la red. La solución usual es utilizar un *buffer* (*jitter buffer*) que almacene los paquetes antes de entregarlos al destino asegurándose así de que lleguen todos en orden, aunque esto introduce un retardo adicional.

Pérdida de Paquetes: Porcentaje de paquetes que no llegan a su destino. Esta pérdida puede producirse por errores en alguno de los equipos que permiten la conectividad de la red o por sobrepasar la capacidad de algún *buffer* de algún equipo o aplicación en momentos de congestión.

Ancho de banda: Cantidad mínima de ancho de banda requerido por un flujo de una aplicación. Debe especificarse el intervalo de tiempo, ya que diferentes intervalos conllevan diferentes resultados.

2.2.- MODELOS DE QoS

Existen distintos modelos de calidad de servicio, y para cada modelo se diferencian varios tipos de servicios. Los tipos de servicio más importantes para QoS extremo a extremo son los servicios integrados y los diferenciados.

2.2.1.- Servicios Integrados

2.2.1.1.- Definición

La arquitectura de red integrada de servicios (IntServ [22]) define un conjunto de modificaciones al tradicional modelo Best Effort utilizado en Internet con el fin de permitir calidad de servicio extremo a extremo. En este modelo, los puntos extremos clave son las aplicaciones emisoras y receptoras que solicitan un nivel de servicio deseado a la red para un conjunto de flujos, definidos por sus direcciones, sus puertos y por el protocolo de transporte. La definición del servicio que está dando el emisor está descrita en el Tspec (Transmisión Specification). De la misma manera, la información referente al servicio que espera recibir el receptor está incluida en el Rspec (Receiver Specification). Comparando esta arquitectura de servicio con el correo ordinario, IntServ equivaldría al correo certificado, con ciertos parámetros en lo que respecta a pérdidas y entrega garantizada.

2.2.1.2.- Tipos de servicios

IntServ describe tres tipos principales de servicio que puede solicitar una aplicación:

- **Servicio Garantizado** (Guaranteed Services [14]): Provee límites matemáticamente probables sobre las demoras de encolado de paquetes extremo a extremo, haciendo lo posible por proveer un servicio que garantice el ancho de banda y el retraso.
- **Carga Controlada** (Controlled Load [13]): Provee al flujo de aplicación una calidad de servicio equivalente al que recibiría el mismo flujo en un elemento de red que se encuentra bajo muy poca carga, pero utiliza control de capacidad para asegurar que el servicio es recibido aún si el elemento de red está sobrecargado.
- **Mejor Esfuerzo** (Best Effort): No provee ni garantiza ningún tipo de servicio.

2.2.1.3.- Características

Ventajas del modelo de red IntServ:

- Simplicidad conceptual
- Calidad de servicio por flujo. Esto hace que sea útil para el control de llamadas de voz, ya que permite avisar a los puntos finales si el ancho de banda necesario está disponible.

Inconvenientes del modelo de red IntServ:

- Todos y cada uno de los elementos de la red debe intercambiar tráfico de estado y señalización por cada flujo, lo que significa un aumento del tráfico.
- Todos los nodos intermedios deben implementar RSVP u otro protocolo IntServ.

Aunque RSVP es el protocolo más utilizado y conocido para implementar este mecanismo, la arquitectura IntServ está diseñada para poder soportar otros protocolos.

2.2.2.- Protocolo RSVP

RSVP (Resource Reservation Protocol [15]) es un protocolo de señalización que las aplicaciones pueden utilizar para solicitar recursos a la red. La red responde explícitamente con la admisión o el rechazo de estas solicitudes RSVP. Algunas aplicaciones que han de cuantificar las necesidades de recursos expresan estas necesidades utilizando parámetros IntServ definidos en las correspondientes especificaciones. Como se ha comentado anteriormente, el protocolo RSVP y la arquitectura IntServ son separables.

El modelo más reciente de este protocolo está basado en una combinación RSVP/IntServ. En este modelo, las señales RSVP indican los requerimientos del flujo emisor a los elementos de la red utilizando parámetros IntServ. Estos elementos de red aplican el control de admisión IntServ para indicar la respuesta a las solicitudes RSVP. Además, el mecanismo de control de tráfico en la red está configurado para asegurar que cada flujo recibe una respuesta de servicio de una manera aislada e independiente.

Los siguientes factores han impedido el despliegue tanto del protocolo RSVP como de la arquitectura IntServ en Internet:

- Debido a la escasa escalabilidad para grandes redes.

- Debido al escaso número de elementos de la red que generan señalización RSVP.
- Debido a la necesidad de una política de control de acceso, de autenticación y de contabilidad.

2.2.3.- Servicios Diferenciados

2.2.3.1.- Definición

La arquitectura de red de servicios diferenciados (DiffServ [16]) se basa en la división del tráfico en diferentes clases, asignándole a cada una de ellas diferentes prioridades. En este modelo es importante la configuración del router de acceso a la red, ya que es el que se debe encargar de clasificar y controlar el tráfico que entra a la red del proveedor e informar a los distintos routers del cumplimiento de las especificaciones requeridas.

Los paquetes que pertenecen a una determinada clase se marcan con un código específico denominado DSCP (DiffServ Code Point). Este código es todo lo que se necesita para identificar una clase de tráfico. Por otro lado, la diferenciación de servicios se consigue mediante la definición de comportamientos específicos para cada clase de tráfico entre dispositivos de interconexión siguiendo el protocolo PHB (Per Hop Behavior). Los nodos intermedios son los encargados de analizar estos paquetes y tratarlos según sus necesidades.

Si se vuelve a comparar la arquitectura de red con el envío de correo, DiffServ equivaldría a distintos servicios de correo, como podría ser el correo simple, certificado o exprés. Cada servicio ofrece parámetros de entregas diferentes y particulares, y la carta se sella en cada punto de la red para asegurar que recibe el servicio requerido. Esta arquitectura tan solo determina un servicio diferenciado en un sentido del tráfico, por lo que se puede considerar asimétrica.

2.2.3.2.- Tipos de servicios

Es posible conseguir distintos servicios diferenciados según las especificaciones contenidas en el PHB:

- **Reenvío Expeditivo** (Expedited Forwarding [18]): Provee un servicio de estricta prioridad.
- **Reenvío Asegurado** (Assured Forwarding [17]): Provee una garantía de entrega calificada.
- **Selectores de Clases** (Class Selectors): Provee puntos de código (*code points*) que pueden ser usados para compatibilidad con el modelo IP precedente.
- **Mejor Esfuerzo** (Best Effort): No da garantía de transporte.

2.2.3.3.- Disciplinas de colas

Para diferenciar el tráfico de salida a la red del WAN es posible utilizar disciplinas de colas en el router de acceso. Estas disciplinas permiten diferenciar y clasificar el tráfico según cualquier dato de su cabecera IP (direcciones IP, puertos, TTL, etc.) y separarlas en distintas colas gestionadas por software. Esto permite un mayor margen de diferenciación que con el campo DSCP y una mayor personalización de las distintas clases de tráfico.

Como consecuencia de que en el campo de la cabecera IP hay toda la información referente al paquete, este modelo de QoS permite diferenciar entre usuarios, máquinas, servicios, aplicaciones, etc, lo que lo convierte en una herramienta muy versátil.

2.2.3.4.- Características

Ventajas del modelo de red DiffServ:

- **Escalabilidad:** No es necesario mantener información de estado o flujos.
- Permite un despliegue gradual.
- **Performance:** El contenido del paquete solo se inspecciona una sola vez para clasificarlo. En ese momento, el paquete es marcado y todas las decisiones de QoS posteriores se hacen de acuerdo al valor en un campo fijo de la cabecera, reduciendo los requerimientos de procesamiento.
- **Interoperabilidad:** Todos los fabricantes ya están ejecutando IP.
- **Flexibilidad:** El modelo DiffServ no dictamina que se implemente ninguna funcionalidad particular en un nodo de red. El nodo puede utilizar cualquier técnica que optimice su hardware y arquitectura, mientras sea consistente con el comportamiento esperado.

Inconvenientes del modelo de red DiffServ:

- No es capaz de garantizar de forma determinista una determinada calidad de servicio. Tan solo garantiza un mejor tratamiento a ciertos flujos (calidad relativa) por parte de la red.
- **Starvation:** Este fenómeno se produce cuando una clase de prioridad inferior es servida a una velocidad muy inferior a sus necesidades a causa de que siempre hay un tráfico de prioridad superior esperando ser servido.

- No hay reservas de ancho de banda extremo a extremo, por lo tanto, las garantías de servicio pueden ser imparciales en los nodos de la red que no implementen los PHBs correctamente sobre enlaces congestionados o por nodos que no están correctamente diseñados para el volumen de tráfico esperado de una clase específica.
- La ausencia de control de admisión por flujo o por sesión hace posible que las aplicaciones se congestionen unas con otras.

2.2.4.- Protocolo MPLS

El protocolo MPLS (Multi Protocol Label Switching [20]) es un mecanismo para utilizar etiquetas en lugar de las direcciones IP de los paquetes para enrutar el tráfico y conseguir QoS de manera diferenciada. Estas etiquetas permiten definir rutas predeterminado con el fin de conseguir balanceo de carga y calidad de servicio. En este mecanismo, las rutas se denominan LSP (Label Switching Paths) y se consideran túneles MPLS entre los extremos de la red, y los routers LSR (Label Switching Routers).

Para conseguir diferenciar el tráfico, el protocolo MPLS utiliza el campo de 3 bits llamado *experimental* que se encuentra en la cabecera MPLS (Figura 2.1). Estos bits permiten diferenciar entre 8 clases distintas de tráfico, aunque en la práctica se quedan en 7 clases debido a que una clase es utilizada para el tráfico por defecto.

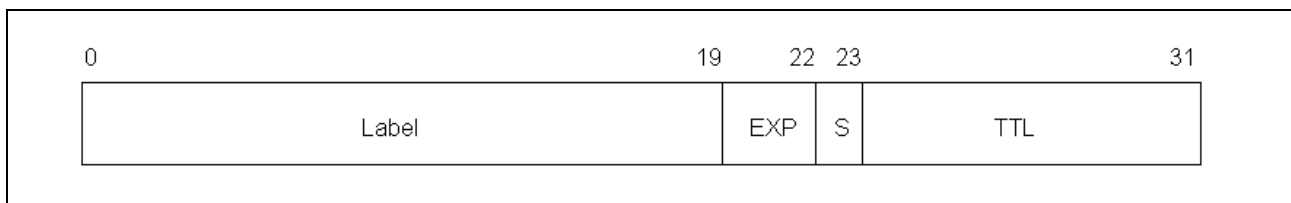


Figura 2.1.- Cabecera MPLS

La clasificación o descarte de este tráfico según la etiqueta MPLS la efectúan los routers de acceso (LSR) antes de enviar el tráfico a la red WAN. El algoritmo de clasificación más sencillo utilizado por este tipo de protocolo es el llamado modo uniforme, que consiste en copiar el campo DSCP en el campo *experimental* de la cabecera MPLS. Sin embargo, existen otros modos más complejos como son el *short pipe* y el *long pipe*, que definen el tipo de tráfico según las necesidades o requisitos del administrador de la red.

2.3.- CONCLUSIONES

Los modelos de calidad de servicio han ido evolucionando con el paso del tiempo según las necesidades de las redes privadas y empresariales. A medida que las infraestructuras de red han ido mejorando, los requerimientos han sido cada vez más estrictos, con lo que se han ido consiguiendo modelos más sofisticados y complejos. Los modelos IntServ y DiffServ han sido los principales y han sentado las bases para el resto de modelos, pero tan solo son unos ejemplos, y existen muchos otros que no se han comentado.

En la actualidad, las características de los distintos modelos de QoS pueden ser muy diversas y pueden moldearse en gran medida a las necesidades de cada caso. Como se verá más adelante, existen herramientas sencillas para entornos GNU/Linux capaces de proporcionar servicios diferenciados extremo a extremo.

Sin embargo, para el caso de General Lab, se requiere un modelo de QoS diferenciado extremo a extremo y que controle el tráfico según disciplina de colas, donde los parámetros necesarios para diferenciar los distintos flujos no se encuentran en el campo DSCP ni en la etiqueta MPLS, sino en cualquier campo de la cabecera del paquete IP, como pueden ser la dirección IP o el puerto. De la misma manera, el tipo de servicio que se requiere no es fijo sino que puede ser configurado por el administrador de la red, basándose en un servicio de prioridades similar al reenvío expeditivo de DiffServ.

Finalmente, estos modelos de QoS se deben utilizar para diferenciar el tráfico y conseguir reservar un determinado porcentaje del ancho de banda disponible para determinados flujos de tráfico. Estas reservas se deben hacer dependiendo de la prioridad definida para cada tipo de tráfico, con el fin de minimizar la latencia de los paquetes con mayor prioridad.

3.- CONTROL DE TRÁFICO EN LINUX

3.1.- INTRODUCCIÓN

Para poder controlar y verificar los diferentes parámetros de QoS, es importante poder gestionar el tráfico que pasa a través de la red. Existen diferentes herramientas hardware y software que dan este tipo de servicios. Sin embargo, las últimas versiones del núcleo o *kernel* de sistemas operativos GNU/Linux, añaden paquetes y herramientas que permiten una gran versatilidad a la hora de controlar el tráfico, tanto en enrutamiento como en filtrado y clasificación de paquetes.

El núcleo de estos sistemas puede llegar a ser un potente enrutador de paquetes IP. Dentro de este *kernel*, existen dos etapas importantes: la cola *ingress* y la cola *egress*. Configurando convenientemente estas colas es posible conseguir una política de clasificación, marcado, descarte, retraso y reenvío de paquetes. La cola de entrada (*ingress*) tan solo permite una etapa de tratamiento de paquetes, como descarte, control de tráfico de entrada, marcado, etc. En cambio, la cola de salida (*egress*) puede adoptar una estructura de árbol de manera que puede contener distintos filtros que distribuyan el tráfico en colas y clases, lo que supone una potente herramienta a la hora de controlar el tráfico de salida (Figura 3.1).

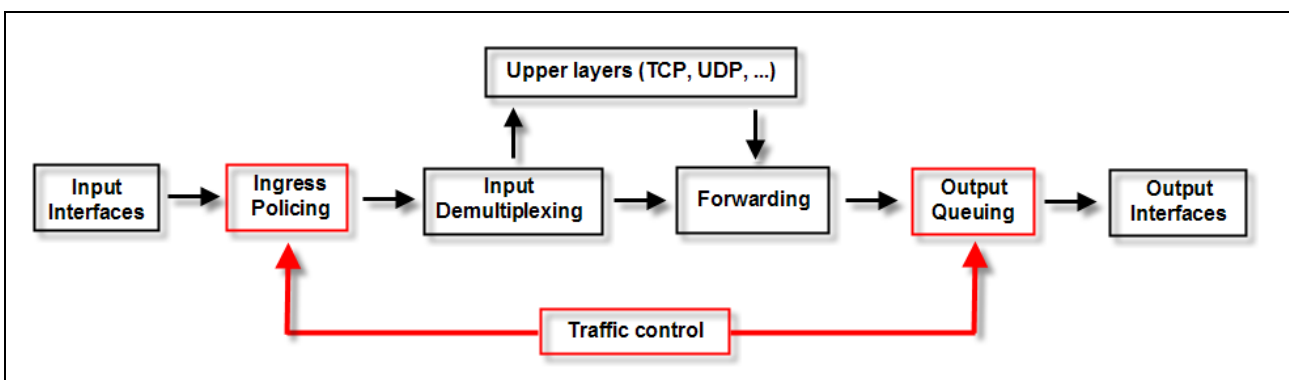


Figura 3.1.- Procesamiento de paquetes en el *kernel*.

En la figura anterior se puede observar de manera general como el *kernel* procesa los paquetes que recibe de la red por una de sus interfaces y como genera nuevos datos para enviarlos por la interfaz de salida. En primer lugar, los paquetes llegan una interfaz de entrada, donde pueden ser susceptibles de vigilancia (*policing*). La vigilancia descarta paquetes indeseados, como por ejemplo, si el tráfico de entrada es demasiado rápido. Una vez pasada esta etapa, los paquetes que la hayan salvado pueden ir o bien de vuelta a la red (ya sea por la misma interfaz o por una diferente, si la máquina está trabajando como router) o se puede pasar a un protocolo superior para realizar algún procesamiento adicional.

Una vez que este procedimiento ha sido realizado, los paquetes son encolados¹ en la correspondiente interfaz de salida. En este punto del proceso es donde se define el control de tráfico. Aquí es posible decidir si los paquetes son encolados o si son descartados, se puede decidir el orden en que son enviados, se pueden retrasar, etc. Una vez se ha liberado el paquete para ser enviado, el controlador del dispositivo lo recoge y lo emite sobre la red.

3.2.- ELEMENTOS DE CONTROL DE TRÁFICO

3.2.1.- Elementos genéricos

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

Shaping: Se encarga de retardar los paquetes con al finalidad de satisfacer una velocidad deseada. Es uno de los requisitos más comunes a la hora de gestionar el ancho de banda disponible. Se utiliza generalmente para suavizar el tráfico a ráfagas y limitar el tráfico.

Scheduling: Su función es la de ordenar los paquetes antes de ser desencolados, tanto en la cola de entrada como en la de salida.

Classifying: Se encargan de ordenar o separar el tráfico en distintas colas. Es el mecanismo por el cual los paquetes son separados para conseguir diferente tratamiento.

Policing: Su función es la de medir y limitar el tráfico en una cola en concreto. Normalmente se requiere su uso en un extremo de la red para asegurar que un elemento no está consumiendo más ancho de banda del que debe. Un *policer* tan solo toma una decisión según la velocidad con que el tráfico está entrando en la cola. Si este tráfico entra a una velocidad inferior al umbral, se toma una decisión. En cambio, si este tráfico entra con una velocidad superior, se toma otra decisión distinta.

Dropping: Se encarga de descartar un paquete, un flujo o una clasificación.

Marking: Es el mecanismo por el cual un paquete es alterado o marcado. Es importante distinguir esta marca de la utilizada por los *firewall* de Linux (*iptables*). Los mecanismos de control de tráfico introducen una marca en el propio paquete, la cual es utilizada y respetada por los demás elementos de la red dentro de un dominio administrativo (DiffServ).

¹ El término encolar se refiere a introducir elementos en la cola. Desencolar se refiere a sacar elementos

3.2.2.- Herramientas de control de tráfico en Linux

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

La herramienta *iproute2* es la encargada de este nuevo paquete. Este fue creado por Alexey Kuznestov y consiste es una colección de utilidades para controlar el tráfico en Linux. Dentro de estas utilidades se pueden encontrar ejecutables como *arpd*, *cstat* o *ifcfg*, pero sin duda las más importantes son *ip* y *tc*. El ejecutable *ip* es el principal, y tiene diferentes funciones de gestión de rutas e interfaces. El ejecutable *tc* (Traffic Control) es el que permite controlar el tráfico con implementaciones de QoS. Para información detallada sobre esta utilidad se aconseja visitar su *manpage*. Aquí se van a comentar algunas de sus herramientas principales:

- **tc qdisc**: Permite establecer disciplinas de colas.
- **tc class**: Permite establecer clases basadas en la planificación de las disciplinas de colas.
- **tc estimator**: Permite hacer una estimación del flujo de una red.
- **tc filter**: Permite configurar el filtrado de paquetes.
- **tc policy**: Permite establecer las diferentes políticas QoS.

Gracias a este paquete es posible configurar los diferentes elementos de control de tráfico en Linux, los cuales están estrechamente relacionados con los elementos genéricos detallados en el punto anterior (Figura 3.2).

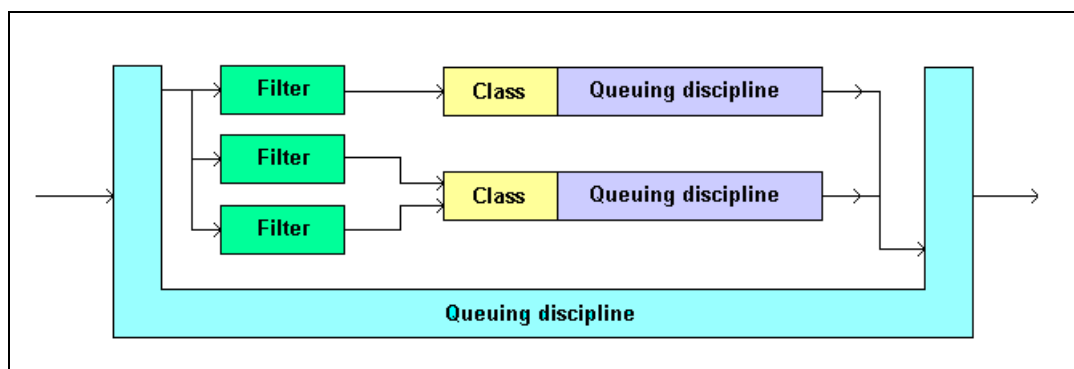


Figura 3.2.- Disposición de los elementos de control de tráfico en Linux [6]

Qdisc (Queue Discipline): Una *qdisc* se puede definir como un *scheduler*, y toda interfaz de salida necesita uno. Si no se determina lo contrario, el *scheduler* por defecto es una cola FIFO_fast. Este es el elemento fundamental sobre el que se basa el control de tráfico en Linux, y se puede distinguir generalmente en dos tipos: colas con clases o colas simples.

Clases: Las clases solo se pueden definir dentro de una cola con clases. Cada una de estas clases puede contener a su vez una o varias clases hijas, lo cual permite modelar escenarios de control de tráfico relativamente complejos. Cada clase puede tener un número ilimitado de filtros asociados. Una clase especial es la *leaf class*, que se puede definir como una clase final, es decir, que no tiene clases hijas. Por defecto se le asigna una disciplina de colas FIFO.

Filtro: Un filtro es sin duda el elemento más complejo y moldeable dentro del control de tráfico. Básicamente se encarga de clasificar los paquetes y asignarlos bien a una clase o bien a una cola con clases. Un filtro necesariamente debe contener un elemento *classifier*, sin embargo, es opcional que contenga un elemento *policer*. Los filtros actúan sobre los paquetes que atraviesan la cola *root* (*root qdisc*), debido a que por este punto entran todos los paquetes correspondientes al tráfico de entrada.

Handle: Cada una de las clases y de las colas del sistema de control de tráfico requiere un identificador único. Este identificador lleva el nombre de *handle*, y se compone por un número mayor y un número menor separado por el símbolo “:”. Este identificador puede ser escogido arbitrariamente, aunque se suele escoger siguiendo las algunas pautas. El identificador ffff:0 está reservado para la *ingress qdisc*.

- **Número mayor:** Define a la clase madre. Todos los objetos provenientes de la misma madre deben compartir el mismo número mayor.
- **Número menor:** Este número identifica a una *qdisc* si es igual a 0. Cualquier otro valor define a una clase, sabiendo que todas las clases de una misma madre deben tener un número menor diferente.

En la tabla de la figura 3.3 se observa la relación entre los elementos de control de tráfico genéricos y los que se encuentran en el núcleo de los sistemas Linux más recientes.

ELEMENTO GENÉRICO	ELEMENTO LINUX
<i>Shaping</i>	Una clase puede actuar como <i>shaper</i>
<i>Scheduling</i>	Una <i>qdisc</i> es un <i>scheduler</i> .
<i>Classyfing</i>	Un filtro utiliza elementos <i>classifier</i> para filtrar tráfico. En Linux, un elemento <i>classifiers</i> no tiene sentido fuera de un filtro.
<i>Policing</i>	Tan solo existe como parte de los filtros.
<i>Dropping</i>	Para poder implementar este elemento, es necesario un filtro con un <i>policer</i> que rechace paquetes.
<i>Marking</i>	La disciplina de cola <i>dsmark</i> utiliza <i>marking</i> .

Figura 3.3.- Relación entre elementos de control de tráfico

3.3. - DISCIPLINA DE COLAS SIMPLES

Las disciplinas de colas simples son aquellas que no contienen ninguna clase asociada, por lo que se limitan principalmente a ordenar, retrasar o descartar los paquetes que le llegan. Normalmente, este tipo de colas están asociadas a la clase principal o a una clase final, ya que se corresponde al último paso antes de entregar los paquetes a la interfaz de red correspondiente.

3.3.1.- First In First Out

La disciplina de cola FIFO (Figura 3.4) no ordena ni modifica los paquetes, tan solo los transmite tan pronto como le es posible. Sin embargo, su longitud es limitada para prevenir desbordamientos en caso de que no sea capaz de desencolar paquetes tan rápido como los recibe. En Linux se implementan dos colas FIFO básicas, una basada en bytes (BFIFO) y la otra basada en paquetes (PFIFO). Independientemente del tipo utilizado, la longitud de la cola se define en el parámetro *limit*.

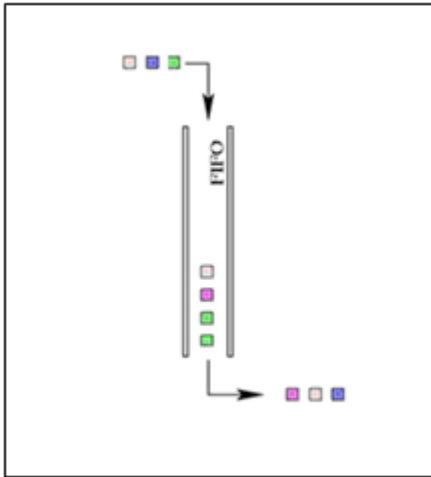


Figura 3.4.- Esquema FIFO [4]

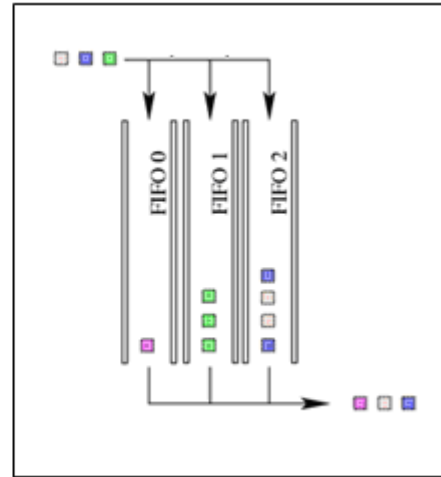


Figura 3.5.- Esquema PFIFO_fast [4]

3.3.2.- PFIFO_fast

La disciplina de cola PFIFO_fast (Figura 3.5) es la que viene por defecto en todas las interfaces de red basadas en Linux. Está desarrollada a partir del algoritmo FIFO, y la única diferencia es que PFIFO_fast es capaz de otorgar cierta priorización. Esto lo hace diferenciando en tres bandas para separar el tráfico, donde cada banda se corresponde con una FIFO distinta. El tráfico más prioritario, como puede ser el tráfico interactivo, se encola en la banda 0 y es la que siempre se transmite primero. De manera similar, la banda 1 se encarga de servir los paquetes pendientes de la banda 0, y lo mismo sucede con la banda 2. De este tipo de cola no existe ningún parámetro configurable. La elección de la prioridad se hace según el valor del campo ToS (Type of Service). En la tabla de la figura 3.6 se muestran algunos de los valores de este campo.

Binario	Decimal	Significado
1000	8	Minimizar retraso (md)
0100	4	Maximizar transferencia (mt)
0010	2	Maximizar fiabilidad (mr)
0001	1	Minimizar coste (mcc)
0000	0	Servicio normal

Figura 3.6.- Significado de algunos valores del campo ToS [23]

3.3.3.- Stochastic Fairness Queuing

La disciplina de cola SFQ (Figura 3.7) pretende distribuir el tráfico de una manera justa entre un número arbitrario de flujos. Para conseguirlo, utiliza una función *hash* para separar internamente el tráfico entre distintas colas FIFO, las cuales son descoladas siguiendo una distribución Round Robin. Existe la posibilidad de que la elección del código *hash* cree cierta injusticia, y por esto es alterado periódicamente. Esta periodicidad se configura con el parámetro *perturb*. Además, el parámetro *quantum* define cuantos bytes son descolados de cada cola FIFO, sabiendo que este valor no puede ser nunca menor al MTU².

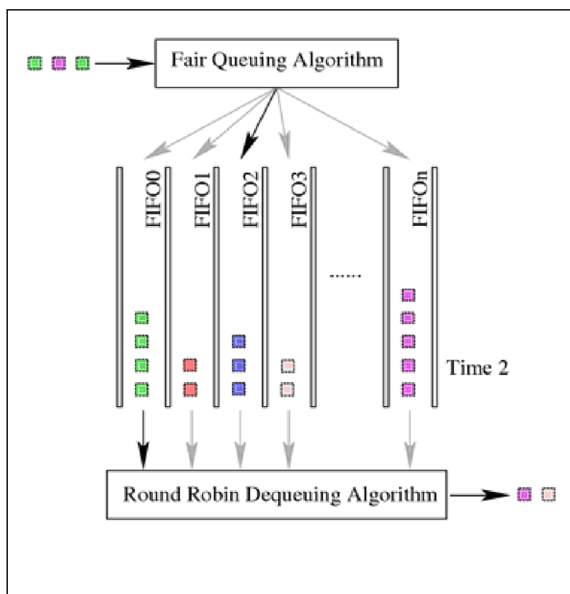


Figura 3.7.- Esquema SFQ [4]

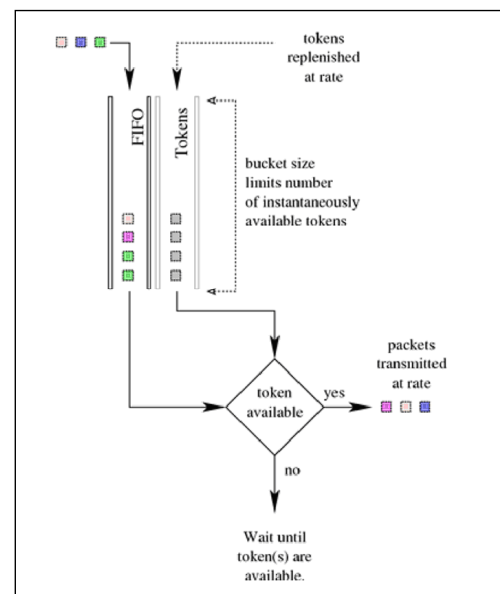


Figura 3.8.- Esquema TBF [4]

3.3.4.- Extended Stochastic Fair Queuing

Conceptualmente, esta cola no difiere en gran medida de SFQ. Sin embargo, esta permite controlar más parámetros lo que la hace más moldeable. ESFQ fue creada para resolver la desventaja comentada anteriormente que ofrecía SFQ, ya que en esta cola es posible escoger el algoritmo de *hash* a utilizar, lo que supone una mejora importante en cuanto a justicia a la hora de gestionar el ancho de banda.

² MTU (Maximum Transfer Unit) es el tamaño en bytes de la unidad de datos más grande que puede enviarse por la red.

3.3.5.- Token Bucket Filter

Esta disciplina de cola tan solo se limita a dejar pasar los paquetes que lleguen a una tasa que no exceda la establecida previamente, pero con la posibilidad de permitir ráfagas cortas que excedan esta velocidad (Figura 3.8). La cola TBF es muy precisa y amigable para la red, y es considerada muy útil si la función que se requiere es la de ralentizar la interfaz. La implementación consiste en una cola (*bucket*) que se llena constantemente con piezas virtuales de información denominadas *tokens* a una velocidad específica (*token rate*), de manera que los paquetes se desencolan tan solo si existen *tokens* disponibles en el *bucket*. Un parámetro importante en esta cola es el tamaño del *bucket*, que es el número de *tokens* que puede llegar a almacenar. En esta disciplina de cola se pueden dar tres casos:

- **Caso 1:** Los paquetes llegan al mismo ritmo que los *tokens*. En este caso, cada paquete es asignado automáticamente a una de las bandas que lo sacará de la interfaz.
- **Caso 2:** Los paquetes llegan a un ritmo mayor que el de los *tokens*. En este caso, los paquetes tendrán que esperar durante un tiempo a que haya disponible un *token* para poder servir el paquete. Si esta situación se prolonga en el tiempo, parte de los paquetes que esperan empezarán a ser descartados.
- **Caso 3:** Los paquetes llegan a un ritmo menor que el de los *tokens*. En este caso, cada paquete será asignado automáticamente a un *token* que lo sacará de la interfaz. Además los *tokens*, que no han sido utilizados para sacar ningún paquete IP, serán almacenados en el *buffer (bucket)* hasta alcanzar el límite del mismo. De esta forma, si cambiara la tendencia y empezaran a llegar paquetes a un mayor ritmo, se podrían utilizar estos *tokens* almacenados para servir los nuevos paquetes.

Los parámetros configurables de esta disciplina de cola son:

- **Limit:** Número de bytes a encolar antes de que los paquetes sean eliminados.
- **Burst:** Tamaño del *bucket* expresado en bytes.
- **Rate:** Velocidad límite especificada por el *scheduler*.
- **Peakrate:** Máxima tasa de agotamiento del *bucket*.
- **Latency:** Máximo periodo de tiempo que un paquete puede permanecer en la cola.

3.3.6.- Random Early Detection

RED es una disciplina de cola sin clases que limita su tamaño de una manera inteligente. Como se ha visto, el resto de colas simples descartan paquetes del final de la cola cuando han llegado a su capacidad máxima, lo cual puede no ser el comportamiento óptimo. Mientras, la cola RED hace este descarte de paquetes de manera gradual. Cuando la cola se acerca a su capacidad promedio, los paquetes encolados tienen una probabilidad configurable de ser marcados, lo que puede terminar finalmente en el descarte del paquete. Esta probabilidad aumenta linealmente hasta un punto llamado el máximo promedio de la capacidad de la cola, que no se corresponde con el tamaño máximo de la cola.

El objetivo principal es mantener una cola de tamaño relativamente pequeño para favorecer la interactividad y la retransmisión de paquetes en ráfagas de tráfico. Dependiendo de como se ha configurado el ECN (Explicit Congestion Notification), el marcado puede significar directamente el descarte del paquete o simplemente una notificación conforme ha sobrepasado su límite de velocidad. El tamaño promedio de la cola se utiliza para determinar la probabilidad de marcado, y se calcula utilizando un promedio exponencial móvil ponderado.

- **Min:** Tamaño promedio de la cola para el cual se considera la posibilidad de marcado.
- **Max:** Tamaño promedio de cola donde la probabilidad de marcado es máxima. Debe ser al menos el doble que *min* para prevenir retransmisiones sincronizadas.
- **Probability:** Probabilidad máxima de marcado. Se recomiendan valores como 0.01 o 0.02.
- **Limit:** Tamaño máximo de la cola. Los paquetes que lleguen por encima de este valor serán descartados. Debe ser mayor que la suma de *max* y *burst*.
- **Burst:** Indica el número máximo de paquetes que pueden pasar en una ráfaga de tráfico. Experimentos de casos reales sugieren la siguiente regla orientativa:

$$\boxed{burst = \frac{2 \cdot min + max}{3 \cdot avpkt}} \quad (3.1)$$

- **Avpkt:** Se utiliza con *burst* para determinar la constante de tiempo para los cálculos del tamaño promedio de la cola.
- **Bandwidth:** Se debe configurar con el valor del ancho de banda de la interfaz.

3.4.- DISCIPLINA DE COLAS CON CLASES

Este tipo de colas se caracterizan por tener una subdivisión interna de su estructura configurable, lo cual es de gran utilidad cuando se tienen diferentes tipos de tráfico que requieren diferentes tratamientos.

Cuando los paquetes llegan a una disciplina con clases, necesitan ser clasificados hacia una de las clases que la componen. Para realizar esta clasificación se consultan los filtros asociados a la disciplina de colas, que devuelven un resultado que permite a la disciplina determinar a que clase debe ser enviado el paquete. Además, cada clase puede tener asociada una nueva disciplina de colas con o sin clases, con lo que es posible que se consulten diferentes filtros hasta que el paquete esté completamente clasificado en una clase.

3.4.1.- PRIO

Esta disciplina de colas con clase recuerda a la disciplina de cola simple PFIFO_fast, aunque es mas completa y permite mayores posibilidades. De la misma manera que la cola simple, esta disciplina define tres clases por defecto, y cada una de ellas tiene asociada una nueva disciplina de colas FIFO. El método de desencolado y las prioridades entre las tres bandas sigue el mismo modelo que la cola simple PFIFO_fast, aunque con algunas diferencias. La primera de las diferencias es que en esta clase es posible definir los filtros necesarios, de forma que no se limita a hacer una clasificación de los paquetes en función del campo ToS, sino que esta clasificación puede ser todo lo compleja que se requiera. Otra diferencia es que se puede definir a cada una de las bandas una disciplina de colas que no necesariamente tiene que ser FIFO, con lo que puede dar lugar a una nueva disciplina de colas con clases y filtros asociados (Figura 3.9).

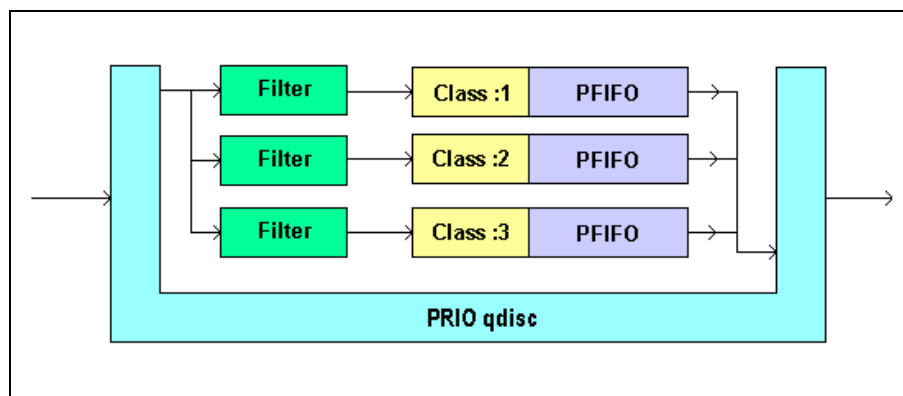


Figura 3.9.- Esquema PRIO [6]

Los parámetros para esta disciplina son:

- **Bands:** Permite especificar el número de clases que se requieren. Por defecto se tienen tres clases o bandas.
- **Priomap:** Define la prioridad de un paquete dentro de una clase. Puede ser directamente configurada por el usuario o definida por el campo ToS. Indica como la prioridad de los paquetes se asignan a una banda o a otra.

3.4.2.- DSMARK

Esta disciplina de colas con clases fue especialmente diseñada para cumplir con las especificaciones de la arquitectura DiffServ (Figura 3.10). Se encarga del marcado de paquetes dentro de la implementación de DiffServ en sistemas Linux, y su comportamiento es simple comparado con el resto de disciplinas de colas. La disciplina DSMARK no prioriza, retarda, ordena o descarta paquetes, tan solo los marca en base al campo DSCP.

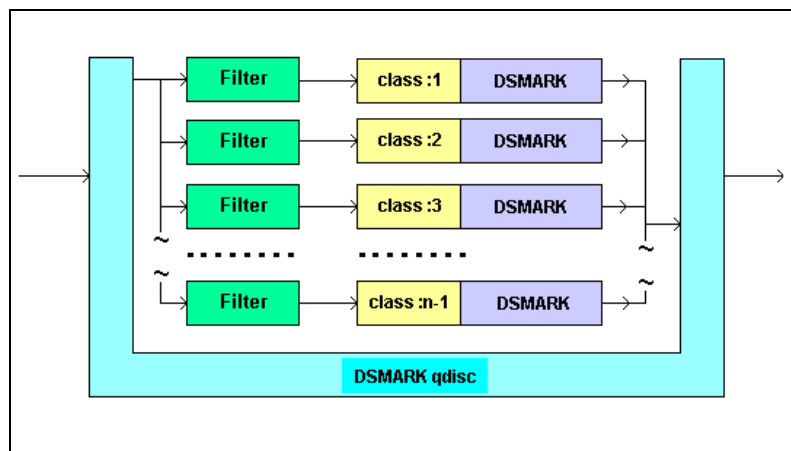


Figura 3.10.- Esquema DSMARK [6]

Las diferentes clases que forman la disciplina son numeradas (1, 2, 3,..., n-1), donde n es un parámetro que define el tamaño de una tabla interna necesaria para implementar el comportamiento de la cola. Este parámetro se denomina índice. Siendo q el número mayor de la cola, el elemento con identificador $q:0$ es la cola principal misma. Elementos a partir de $q:1$ a $q:(n-1)$ son las clases de la disciplina de cola.

Los diferentes filtros envían los paquetes a su respectiva clase. Esta clasificación se lleva a cabo según la marca del campo DSCP, donde la marca consiste en un valor entero que va asociada a una clase en concreto.

3.4.3.- Hierarchical Token Bucket

La disciplina de colas HTB surge como una disciplina sencilla de entender, intuitiva y rápida. Esta nueva disciplina permite controlar el uso del ancho de banda de salida de un enlace. También permite simular varios enlaces de manera jerárquica en una misma interfaz de red y definir que tipo de tráfico debe pasar por cada uno de ellos. Su principal característica es que es posible pedir prestado tráfico de un enlace a otro si este último no lo está utilizando.

3.4.3.1.- Árbol de preferencias

Esta disciplina de colas permite gestionar de manera jerárquica el ancho de banda disponible. Para ello, cada clase puede estar dividida en diferentes clases hijas, y así sucesivamente. Esta distribución es lo que se denomina árbol de preferencias (Figura 3.11). HTB asegura que la capacidad provista a cada clase sea al menos el mínimo entre el ancho de banda asignado y el tráfico solicitado. Cuando una clase solicita menos que la cantidad asignada, el ancho de banda excedente se distribuye a las otras clases que soliciten servicio según unos parámetros de prioridad.

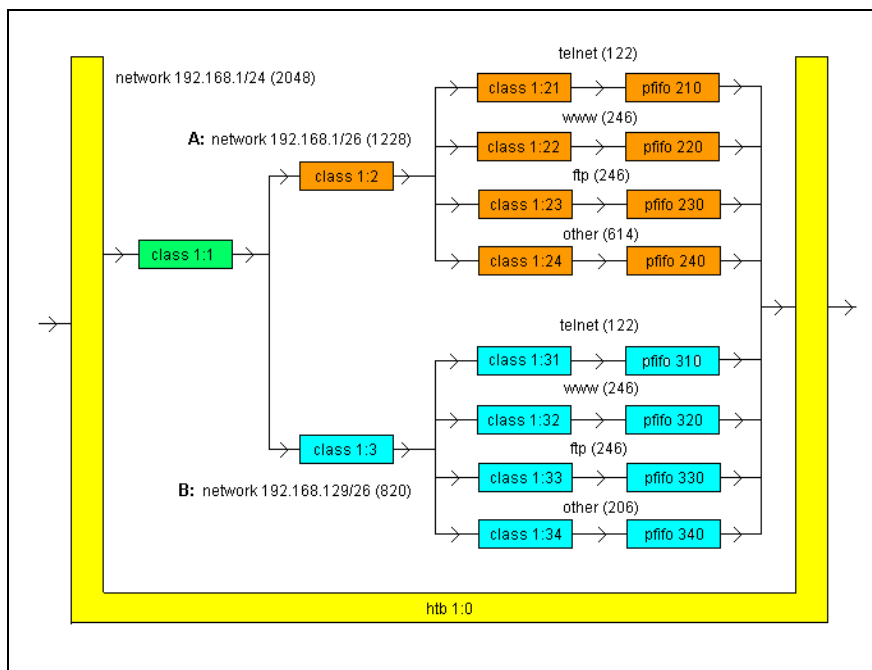


Figura 3.11.- Árbol de preferencias para HTB [6]

En este árbol de preferencias (Figura 3.11) se puede ver un ejemplo de una posible estructura jerárquica de una interfaz de red. Está dividido entre la clase *root*, las clases madre y las clases hijas. Cada una de estas clases se define con su número de identificación o *handle*.

3.4.3.2.- Creación de clases

La metodología a seguir para la creación de cada una de las clases se puede ver en el siguiente ejemplo:³

```
# tc qdisc add dev eth0 root handle 1:0 htb default 4
```

Este comando conecta una disciplina de cola HTB a la interfaz eth0, le asigna el *handle* 1:0 y define que por defecto el tráfico será asignado a la clase 4.

Una vez asignada la disciplina a la interfaz, es posible crear el árbol de preferencias. A continuación se muestra como crear algunas de sus clases siguiendo con el ejemplo anterior:

```
# tc class add dev eth0 parent 1:0 classid 1:1 htb rate 2048kbit ceil 2048kbit
# tc class add dev eth0 parent 1:1 classid 1:2 htb rate 1228kbit ceil 1228kbit
# tc class add dev eth0 parent 1:2 classid 1:21 htb rate 122kbit ceil 1228kbit
# tc class add dev eth0 parent 1:2 classid 1:22 htb rate 246kbit ceil 1228kbit
```

El primer comando crea la clase *root*. Esta clase se define como la clase que tiene como madre la disciplina de cola HTB. Esta clase es necesaria ya que una clase *root* permite que sus clases hijas se presten el ancho de banda excedente unas a otras, pero no es posible que una clase *root* pida prestado a otra clase del mismo tipo. Si se tiene esto en cuenta, sería posible crear diferentes clases directamente debajo de la disciplina de cola HTB, pero en el caso de que existiera tráfico excedente no cabría la posibilidad de prestarlo de unas clases a las otras.

El comando *rate* y *ceil* tienen que ver con el ancho de banda asignado y el ancho de banda máximo que puede llegar a ocupar una determinada clase. Un flujo asignado a una clase podrá contar con el ancho de banda definido en *rate* siempre que lo requiera. Además, si el resto de clases no están utilizando todo su ancho de banda asignado, este flujo podrá pedir prestado al resto de clases hasta llegar a un máximo especificado en *ceil*. Por defecto, el *ceil* es igual que el *rate* definido para una clase en concreto. Si se desea configurar manualmente este parámetro, se debe tener en cuenta que el *ceil* de una clase debe ser al menos el valor del *rate*, y que el *ceil* debe ser al menos igual que el *ceil* máximo de cualquiera de sus clases hijas.

El resto de clases hijas se han creado de la misma manera que la clase *root*, teniendo en cuenta quien es el *parent* o clase madre, el *handle* y los distintos parámetros referentes al ancho de banda.

³ Se ha seguido como criterio tratar como figura a las imágenes, las tablas y los archivos, y no a los comandos ni a las salidas de los comandos. Sin embargo existen algunas excepciones.

Opcionalmente, es posible asignar una disciplina de cola simple a las clases finales. Por defecto a estas clases se les asigna una cola PFIFO.

```
# tc add dev eth0 parent 1:21 handle 210:0 pfifo limit 5
# tc add dev eth0 parent 1:21 handle 211: sfq perturb 10
```

3.4.3.3.- Parámetros opcionales

A parte de los parámetros comentados anteriormente, existen otros menos comunes pero que es posible configurarlos en caso que sea requerido.

Quantum

Cuando se da el caso de que distintas clases quieren tomar prestado ancho de banda excedente, se les otorga a cada una de ellas un determinado número de bytes antes de atender a otra clase. Este parámetro se denomina *quantum*. El valor de este parámetro debe ser mayor que el MTU siempre que sea posible. Normalmente, no es necesario configurarlo manualmente, ya que HTB permite seleccionar valores predeterminados. De esta manera, la disciplina de cola define el valor de *quantum* de las distintas clases dividiendo su *rate* por un parámetro global llamado *r2q*. Para velocidades relativamente altas (valores de *rate* superiores a 15kBps⁴) el valor por defecto que toma *r2q* es 10. De lo contrario, para velocidades menores se toma como valor 1. En el caso de que el *quantum* se especifique de manera explícita en el comando, el valor de *r2q* es ignorado.

Si se diera el caso de que el *quantum* es menor que el MTU, es decir, que se envían paquetes con una longitud mayor a *quantum*, se registra un error en el registro */var/logs/messages* que indica que el *quantum* es demasiado pequeño, pero el paquete se sirve sin ningún tipo de problema ni penalización para la clase. También se registran errores en este log cuando el retraso al desencolar paquetes como consecuencia de usar HTB es superior a 5 segundos.

Ráfagas

Por otro lado está el concepto de ráfagas. La interfaz de red solo permite enviar paquete a paquete y a la velocidad que el hardware soporte. Por lo tanto, los valores de *rate* y *ceil* comentados anteriormente no se consideran medidas instantáneas, sino promedios durante un periodo de tiempo determinado.

⁴ Si no se dice lo contrario, y a diferencia del binario *tc*, se sigue el siguiente criterio:
1 kbps=1024 bps (bits por segundo)
1 kBps=1024 Bps (Bytes por segundo)

Lo que realmente sucede es que se permite que una clase desencole paquetes a la velocidad máxima de la interfaz durante un periodo de tiempo, y seguidamente se hace lo mismo con otras clases. De aquí surgen los parámetros *burst* y *cburst*, que se encargan de controlar la cantidad de datos que pueden ser enviados a velocidad máxima sin prestar servicio a otras clases.

Más concretamente, el valor de la velocidad *rate* y *ceil* se controlan con dos colas o *buckets*, uno para el *rate* (*token*) y el otro para *ceil* (*ctoken*). El tamaño de estas colas se define con los parámetros *burst* y *cburst* respectivamente, y su funcionamiento es idéntico al descrito para la disciplina con clases TBF, manteniendo la condición de que los valores *burst* y *cburst* de una clase deben ser iguales o mayor que el de cualquiera de sus clases hijas.

Prioridad

El último concepto que permite configurar la disciplina de colas HTB es el de prioridad, gracias al parámetro *prio*. Cuanto menor sea este parámetro mayor será la prioridad de la clase, siendo *prio*=0 la clase más prioritaria. La utilidad de este parámetro es que el ancho de banda excedente es ofrecido primero a las clases con mayor prioridad, siempre respetando los valores de *rate* y *ceil*.

3.4.3.4.- Filtros

Una vez se han creado las clases, es necesario clasificar el tráfico en su correspondiente clase. Para esta función se utilizan los filtros. Existen gran variedad de filtros para disciplinas de colas con clases:

- **Filtro fw:** Basa la decisión en la marca del paquete hecha por el *firewall iptables*.
- **Filtro route:** Basa la decisión en la ruta que debe seguir el paquete.
- **Filtro tcindex:** Es el que se utiliza para las disciplinas DSMARK.
- **Filtro rsvp:** Solo es útil para redes que soporten el protocolo RSVP.
- **Filtro u32:** Basa la decisión en los diferentes campos del paquete IP.

Por su versatilidad y por ser el más avanzado se suele utilizar el filtro u32. Por definición, este filtro permite filtrar en función de cualquier conjunto de bits, ya sea de la cabecera del paquete IP o del segmento de datos.

Para hacerlo, compara un valor determinado con cualquier segmento del paquete IP de la siguiente manera:

```
#tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32\
match u8 64 0xff at 8 \
flowid 1:21
```

Este filtro clasificará a la clase 1:21 el tráfico cuyo octavo byte coincida con el valor 64. Si se mira la distribución de bytes de un paquete IP se observa que el octavo bytes es el TTL. En los filtros, de la misma manera que en las clases, es importante la prioridad. El tráfico irá pasando a través de los diferentes filtros en orden de prioridad, y cuando un paquete coincida con uno de ellos ya no se comparará con el resto.

Pero esta manera de configurar los filtros es algo compleja de interpretar, con lo que existe otro método para hacerlo de manera más intuitiva. Este filtro también permite hacer coincidir valores con información del paquete, sin necesidad de indicar en que byte se encuentra información.

```
#tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32\
match ip src 128.0.0.50/32 \
match ip dport 23 0xffff \
flowid 1:21
```

Como se puede observar, este nuevo filtro es mucho más intuitivo. En concreto, se enviará a la misma clase los paquetes cuya dirección IP origen sea la 128.0.0.50 y el puerto destino el 23.

3.4.3.4.- Otros parámetros

El comando *tc* también permite visualizar los elementos de control de tráfico creados en una interfaz (clases, colas y filtros) con la opción *show*.

```
# tc class show dev eth0
# tc qdisc show dev eth0
# tc filter show dev eth0
```

En el caso de que no exista ninguna configuración, se muestran las disciplinas de colas por defecto *PFIFO_fast*, con la asignación de las distintas bandas.

```
# tc qdisc show
qdisc pfifo_fast 0: dev eth0 root bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1
```

3.5.- CORTAFUEGOS EN LINUX

3.5.1.- Definición

Netfilter [24] es un *framework* disponible en el núcleo de sistemas Linux que manipula paquetes de red en distintos estados del procesamiento. Esta herramienta es parte del proyecto *netfilter* que se encarga de soluciones libres para *firewall* de Linux.

El componente más popular de este proyecto es *iptables*, una herramienta de *firewall* que permite filtrar paquetes y traducir direcciones de red para IPv4. Con este paquete es posible configurar las tablas, cadenas y reglas de *netfilter*, y solo puede ser utilizado por el súper-usuario

3.5.2.- Funcionamiento

Gracias a *iptables*, es posible definir reglas y filtros para manipular los paquetes que llegan a una interfaz. Estas reglas están agrupadas en cadenas (*chains*), donde cada una de estas cadenas es una lista ordenada de reglas donde importa el orden. Estas cadenas se agrupan en tablas, donde cada tabla está asociada con un procesamiento de paquetes distinto. Existen tres cadenas básicas: INPUT, OUTPUT y FORWARD, asociadas a las tablas FILTER, NAT y MANGLING.

Como se ha dicho anteriormente, una regla no es más que una decisión sobre un atributo del paquete. Este paquete irá pasando secuencialmente por cada una de las reglas de la cadena hasta coincidir con una de ellas. Si después de recorrer toda la cadena el paquete no cumple ninguna de las reglas, se ejecutará la acción por defecto asignada a esa cadena en concreto. La tabla por defecto es FILTER.

Para configurar la tabla sobre la que se quiere trabajar basta con introducir el comando *iptables* con la opción *-t*.

```
# iptables -t TABLA
```

NAT: Esta tabla se utiliza para configurar el protocolo NAT (Network Address Protocol). Cuando un flujo de paquetes atraviesa la tabla, el primer paquete es admitido y el resto son identificados como parte de este mismo flujo y se llevan a cabo sobre ellos las operaciones NAT. Existen tres cadenas sobre la que se pueden añadir las reglas: PREROUTING, OUTPUT y POSTROUTING. La cadena PREROUTING se utiliza para modificar los paquetes tan pronto como llegan al núcleo.

La cadena OUTPUT se utiliza para modificar los paquetes generados localmente en el núcleo antes de tomar ninguna decisión de enrutado. Finalmente, la cadena POSTROUTING se utiliza para modificar los paquetes que acaban de salir del núcleo.

MANGLE: Esta tabla permite manipular los diferentes campos de los paquetes (ToS, TTL, etc.) excepto el NAT, que se realiza en la tabla anterior. Tiene asociado dos cadenas: PREROUTING y OUTPUT.

FILTER: En esta otra tabla se lleva a cabo el filtrado de paquetes. Tiene asociada las cadenas INPUT, FORWARD y OUTPUT. La primera hace referencia a los paquetes entrantes cuyo destino es el propio núcleo. La segunda cadena se utiliza para decidir la ruta de los paquetes que tienen como destino otro equipo. La cadena OUTPUT se utiliza para filtrar paquetes generados en el propio equipo que el destino es externo.

3.5.3.- Opciones y parámetros

Para conocer detalladamente el uso del comando *iptables* se recomienda consultar *manpage*. A continuación se comentarán algunos de los parámetros utilizados en este proyecto.

Para configurar las cadenas se tienen los siguientes parámetros:

- **iptables -N:** Crear una nueva cadena vacía.
- **iptables -X:** Elimina una cadena vacía.
- **iptables -F:** Elimina todas las reglas de una cadena.
- **iptables -P:** Cambia la política por defecto de una cadena.
- **iptables -L:** Lista las reglas de una cadena.
- **iptables -Z:** Pone a cero las variables de auditoría de una cadena.
- **iptables -I:** Inserta una nueva regla al comienzo de la cadena.
- **iptables -R:** Reemplaza una regla de una cadena.
- **iptables -D:** Elimina una regla de una cadena.

Para configurar las reglas:

- **iptables -s**: Indica un dominio o una dirección IP de origen.
- **iptables -d**: Indica un dominio o una dirección IP de destino.
- **iptables -i** (*--in-interface*): Indica la interfaz de entrada.
- **iptables -o** (*--out-interface*): Indica la interfaz de salida.
- **iptables -p**: Especifica el protocolo del datagrama.

Los nombres válidos de protocolos son TCP, UDP, ICMP. Cada protocolo lleva asociados sus propios modificadores a través de las extensiones correspondientes. Para el protocolo TCP se tienen las siguientes extensiones:

--sport: Especifica el puerto origen.

--dport: Especifica el puerto destino.

Además, existe la posibilidad de marcar los paquetes que cumplan una determinada regla. Esta marca tan solo será visible dentro del cortafuego del núcleo. Para hacerlo hay que utilizar la opción *mark*. En el siguiente ejemplo se marcan con el valor 1 a los paquetes cuyo puerto destino sea el 25.

```
#iptables -A PREROUTING -i eth0 -t mangle -p tcp --dport 25 -j MARK --set-mark 1
```

3.6.- ELECCIÓN DE LA DISCIPLINA DE COLAS

En lo que se refiere a las disciplinas de colas con clases, por definición, todas ellas son capaces de desglosarse en distintos niveles jerárquicos. Sin embargo, de entre las disciplinas analizadas tan solo HTB permite gestionar el ancho de banda excedente de tal manera que se comparte de unas clases a otras. Si bien es cierto que existen otras disciplinas que cumplen este requisito, como es el caso de CBQ (Class Based Queueing), se ha decidido no incluirlas en el análisis por ser demasiado complejas de configurar y de interpretar.

Por otro lado, la disciplina HTB permite utilizar distintos tipos de filtros para clasificar el tráfico. Del análisis de todos estos filtros se concluye que el u32 es el que optimiza el proceso de filtrado, ya que permite hacer sus decisiones en base a todos los campos del paquete IP, y no solo en función del campo DSCP como el filtro *tcindex* de la disciplina DSMARK. Esto ofrece grandes ventajas a la hora de diferenciar y clasificar el tráfico de una manera eficiente.

La disciplina HTB es también la más elaborada, la más completa y es más sencilla de utilizar y de comprender que otras como TBF, DSMARK o la misma CBQ, con lo que se facilita en gran medida el proceso de configuración.

Sin embargo, el uso de esta herramienta también presenta algunos inconvenientes. El primero de ellos es que, de la misma manera que todas las demás disciplinas, el hecho de que el tráfico deba ser clasificado y filtrado implica un aumento obligado de la latencia. Otro inconveniente es la aparición del fenómeno *starvation*, que hace que el tráfico correspondiente a las clases menos prioritarias nunca pueda ser desencholado si siempre existe otro tráfico más prioritario que se lo impide.

En base a todos estos criterios, la disciplina HTB es la que más ventajas presenta respecto el resto de disciplinas de colas con clases, con lo que es la que se va a utilizar para crear el gestor de ancho de banda.

En lo que se refiere a las disciplinas de colas simples o terminales, en base a la teoría todas ellas cumplen las condiciones necesarias para ser utilizadas en el sistema de gestión. Más adelante se comprobará que no todas ellas son útiles en situaciones reales.

4.- ANÁLISIS DE POSIBLES HERRAMIENTAS

4.1.- REQUISITOS

A la hora de escoger la herramienta que se va a utilizar para crear el sistema de gestión, se han determinado unos requisitos previos. Uno de los más importantes es que fuera libre y gratuito. Esta premisa deja fuera soluciones hardware ya implementadas o soluciones software que requiera algún tipo de licencia.

Otro requisito es que la solución escogida sea configurable por el administrador de la red en de una manera sencilla, transparente para el usuario y sin necesidad de parar el servicio durante un gran periodo de tiempo. Sería positivo que pudiera ser usado a través de un entorno Web de forma remota.

La solución que se busca debe ser capaz de priorizar los servicios y gestionar el ancho de banda excedente. Según el capítulo anterior, este requisito implica que la solución final debe estar basada en disciplinas de colas HTB o similares.

En base a estos requisitos, se ha realizado la búsqueda en soluciones software dentro de un entorno GNU/Linux, ya que debido a su tipo de licencia GNU GPL existen soluciones interesantes y versátiles. Sin embargo, se han analizado tanto herramientas que cumplen con estos requisitos como herramientas que no lo hacen. De esta manera ha sido posible compararlas y encontrar una solución final que se ajuste lo máximo posible a las necesidades de la empresa.

4.2.- ANÁLISIS DE HERRAMIENTAS

4.2.1.- Soluciones en el mercado

4.2.1.1.- Eduna Enterprise 100Mb

E-duna [25] es una solución para empresas de la compañía española especializada en soluciones de telecomunicaciones INOLABS. Para el caso de General Lab sería necesaria la solución Eduna Enterprise 100Mb. Esta es una solución hardware que permite gestionar el ancho de banda de distintos grupos de usuarios, además de ofrecer servicios de monitorización de tráfico y usuarios.

Ventajas:

- Dispone de asistencia técnica.
- Es capaz de gestionar el tráfico filtrando por servicio y usuario.
- Configuración vía Web.
- Dimensiones reducidas y enrackable ocupando un espacio de 1U.

Inconvenientes:

- Realiza tareas de monitorización y filtrado de tráfico. La empresa ya dispone de estos tipos de servicios.
- Orientado a gestión de tráfico con salida a Internet.
- Número limitado de usuarios y de tráfico.
- Precio: 4.500€.

4.2.1.2.- PacketShaper 3500

PacketShaper [27] es una solución hardware basada en un sistema de administración de tráfico y ancho de banda para aplicaciones en redes WAN. Forma parte de uno de los productos de la compañía Packeteer, especializada en gestión de redes empresariales. Existen diversas versiones del producto, pero para las especificaciones de la empresa la mejor opción es PacketShaper 3500.

Ventajas

- Dispone de asistencia técnica.
- Orientado a redes de área local.
- Permite gestión remota.
- Posibilidad de enrackar en el armario de servidores, ocupando un espacio de 2U.

Inconvenientes

- Número limitado de usuarios y de tráfico.
- Precio aproximado: 4.000€.

4.2.1.3.- SoftPerfect Bandwidth Manager

SoftPerfect Bandwidth Manager [26] es una solución software de la compañía australiana SoftPerfect. Permite gestionar el tráfico creando reglas en un entorno Windows, con la posibilidad de configurar el gestor de manera remota.

Ventajas:

- Sistema Operativo Windows. Posibilidad de instalarlo en algún servidor en producción de la empresa.
- Permite gestión remota.
- Es capaz de gestionar el tráfico filtrando por servicio y usuario.

Inconvenientes:

- Asistencia técnica a través de correo electrónico, no presencial.
- Orientado a gestión de tráfico con salida a Internet.
- Licencia Microsoft Windows.
- Precio: 250\$ con licencia para 10 servidores.

4.2.2.- Creación de una aplicación personalizada

En un primer momento se estudió la posibilidad de crear un software propio y personalizado para configurar las funciones de gestión de ancho de banda. Esta aplicación sería implementada estrictamente según las necesidades de la empresa, haciéndolo moldeable a sus requerimientos.

Sin embargo, como se verá en este análisis de posibles herramientas, existen diferentes aplicaciones que ya cumplen con estas necesidades, de manera que se prefirió partir de un software ya creado e intentar mejorarlo y ampliarlo.

Cabe decir que si en este análisis no se hubiera encontrado ninguna herramienta que cumpliera con los requisitos deseados, hubiera sido necesario crear una aplicación personalizada.

4.2.3.- Herramientas basadas en *scripts*

4.2.3.1.- Solución HTB.init

El *script* HTB.init [28] se basa en un fichero ejecutable desde la consola que facilita de forma considerable la tarea de gestionar el tráfico con clases HTB. Su principal propósito es el de permitir gestionar grandes configuraciones de manera individual para cada clase HTB a partir de sencillos ficheros.

Cada una de estas clases está descrita en un fichero ubicado por defecto en la ruta `/etc/sysconfig/htb`, o de lo contrario donde indique la variable `$HTB_PATH`. El nombre de estos ficheros identifica cada clase con la interfaz a la que está asociada, el identificador de la clase, el identificador de la madre y, opcionalmente, una descripción de la forma:

```
$HTB_PATH/<ifname>-<clsid>(:<clsid>).<description>
```

Algunos nombres válidos de ejemplo podrían ser:

- **eth0-2**: Clase *root* con identificador 2 en la interfaz eth0.
- **eth0-2:3**: Clase hija con identificador 3, madre 2 en la interfaz eth0
- **eth0-2:3:4**: Clase hija con identificador 4, madre 3 en la interfaz eth0
- **eth1-2.root**: Clase *root* con identificador 2 en la interfaz 1. Como descripción se ha puesto el nombre de la clase.

La principal ventaja de esta solución es que permite gestionar gran parte de los parámetros de las clases HTB y de sus colas de una manera sencilla asignando valores a ciertas variables. Además, permite implementar árboles de preferencia tan complejos como sea necesario gracias al tratamiento individual de cada clase, y permite filtrar el tráfico por franja horaria o por día de la semana haciendo uso de reglas *iptables*.

Sin embargo, no es una herramienta ampliamente extendida, por lo que la documentación al respecto es bastante escasa. La información necesaria acerca del proceso de instalación y la configuración se pueden encontrar en el mismo fichero HTB.init.

Una posible configuración de ejemplo podría ser la mostrada en la figura 4.1.

Nombre	eth0	eth0-2.root	eth0-2:10.www	eth0-2:20.smtp
CONTENIDO	DEFAULT=30	RATE=5Mbit BURST=15k	RATE=5Mbit BURST=15k LEAF=sfq RULE=*:80,	RATE=3Mbit CEIL=5Mbit BURST=15k LEAF=sfq RULE=*:25

Figura 4.1.- Ejemplo de configuración de HTB.init

Ventajas:

- Tratamiento de cada clase por separado.
- Creación de árboles de preferencia tan complejos como se requiera.
- No existe límite de usuarios ni de tráfico.
- Permite filtrar por franja horaria.
- Software libre.

Inconvenientes:

- Escasa documentación.
- No permite configurar todos los parámetros.

4.2.3.2.- Solución con HTB-tools

El HTB-tools [29] es un software más completo que el anterior, que contiene distintas herramientas que ayudan a simplificar el proceso de gestión de ancho de banda, tanto de subida como de bajada. Para su funcionamiento, es necesario copiar los binarios de la carpeta *tc* del paquete *iproute* en la carpeta correspondiente, normalmente */sbin*.

De la misma manera que HTB.init, HTB-tools se basa en la generación de ficheros que al ejecutarlos forman las clases, colas y filtros deseados, creando un fichero por cada interfaz, lo que deja más compacta la información (Figura 4.2). Como posible inconveniente respecto al caso anterior, se tiene que el árbol de preferencias es más limitado, pudiendo disponer tan solo de una clase madre, acotando el árbol a un máximo dos niveles jerárquicos.

Además, no es posible configurar todos los parámetros de las clases HTB, sino que solo se puede cambiar el valor de los principales. También permite marcar paquetes con la opción *mark* de *iptables* para después tratarlos según su marca.

Como novedad, esta solución incluye diferentes herramientas de gran utilidad, como por ejemplo, comprobar si los parámetros configurados son sintácticamente correctos antes de ejecutarlo, controlar el tráfico para una interfaz determinada y visualizar el tráfico en tiempo real.

```
class class_1 {
    bandwidth 192;
    limit 256;
    burst 2;
    priority 1;
    que sfq;

client client1 {
    bandwidth 48;
    limit 64;
    burst 2;
    priority 1;
    mark 20;
    dst {
        192.168.100.4/32;
    };
};

client client2 {
    bandwidth 48;
    limit 64;
    burst 2;
    priority 1;
    mark 20;
    dst {
        192.168.100.5/32;
    };
};

class default { bandwidth 8; };
};
```

Figura 4.2.- Ejemplo del script *eth1-qos.sh*

Ventajas:

- Manuales y amplia documentación en el enlace oficial.
- Software libre.
- Permite comprobar el código antes de ejecutarlo.
- Permite ver el tráfico en tiempo real.
- Permite filtrar según marcas de *iptables*.

Inconvenientes:

- Árbol de preferencias limitado a dos niveles.
- No permite configurar todos los parámetros.

4.2.4.- Herramientas con interfaz Web

La evolución de las anteriores herramientas ha dado paso a otras más completas y que permiten utilizarlas a partir de una interfaz Web. De esta forma se conservan las mismas ventajas y se facilita su uso gracias al entorno Web, que aporta más agilidad, más comodidad y una visualización más agradable.

4.2.4.1.- Solución con módulos de *webmin*

De la unión entre HTB.init y *webmin* surge el módulo *webmin-HTB* [30]. Se trata de un módulo de *webmin* con el objetivo de facilitar la configuración de QoS utilizando clases HTB bajo plataformas Linux. La función de este módulo es el de crear los archivos de configuración de cada clase a partir de un entorno Web. Permite además crear y ver gráficamente el árbol de preferencias y las reglas utilizadas, así como arrancar o parar el servicio.

De la misma manera que *webmin* en general, este módulo en concreto intenta facilitar la tarea de crear los archivos de configuración, con lo que la interfaz Web es muy sencilla. Como inconvenientes podemos encontrar que éste módulo es una versión *beta*, con lo que todavía está en periodo de pruebas. Además, resulta complicado interpretar las reglas de filtrado del tráfico, ya que se muestran en código hexadecimal.

Ventajas:

- Interfaz Web facilita su uso.
- Permite arrancar o parar el servicio.
- Conserva ventajas de HTB-init.

Inconvenientes:

- Módulo *webmin* todavía en versión de pruebas.
- Filtros difíciles de interpretar.
- Conserva inconvenientes de HTB-init.

4.2.4.2.- Solución con *WebHTB*

Para facilitar el uso de HTB-tools, se implementó la aplicación *WebHTB* [31]. Es un software que permite editar y comprobar los ficheros de configuración a partir de un entorno Web muy cuidado basado en un menú de pestañas desplegadas. También es posible ver el tráfico asociado a cada clase en tiempo real, tal y como lo hace HTB-tools.

Durante el periodo de análisis de herramientas para crear el gestor, se han publicado distintas versiones de esta aplicación. La primera de ellas fue la versión v1.5, la cual requería tener instalado y configurado HTB-tools. Las siguientes fueron las versiones v2.x, las cuales no se basan en esta herramienta, sino que se crea directamente un fichero y lo ejecutan.

Ventajas:

- Permite arrancar o parar el servicio a través de la interfaz Web.
- Facilita la interpretación de los filtros.
- No necesita HTB-tools.
- Interfaz Web más manejable y visualmente mejor.
- Aplicación traducida a español.

Inconvenientes:

- Aunque sean aplicaciones independientes, conserva inconvenientes de HTB-tools.

4.3.- ELECCIÓN DE LA HERRAMIENTA

En este análisis se han estudiado herramientas que cumplen alguno de los requisitos impuestos al principio del capítulo. Aunque no todas cumplen estas premisas, se ha intentado dejar constancia de las ventajas y de los inconvenientes de cada una de ellas.

En la tabla de la figura 4.3 se muestra una comparativa de las distintas herramientas analizadas en base a los requisitos necesarios.

REQUISITOS	E-DUNA	PACKETSHAPER	SOFTPROJECT	WEBMIN	WEBHTB
Sistema GNU/Linux	?	?	✗	✓	✓
Disciplina HTB	?	?	✗	✓	✓
Gestión remota	✓	✓	✓	✓	✓
Entorno Web	✓	✓	✓	✓	✓
Gratuito	✗	✗	✗	✓	✓
Apta para redes LAN	✗	✓	✗	✓	✓
Software	✗	✗	✓	✓	✓
Usuarios y tráfico ilimitado	✗	✗	✓	✓	✓

Figura 4.3.- Tabla comparativa de herramientas para gestión de tráfico

En esta tabla se puede observar como todas las herramientas analizadas pueden ser administradas remotamente a partir de una interfaz Web. Sin embargo, el mayor inconveniente de las herramientas en el mercado y que hace que queden directamente descartadas es que sean no gratuitas. Otro requisito imprescindible en vistas de la expansión de la empresa es que sea ilimitado en cuanto a número de usuarios y volumen de tráfico.

Si se descartan las herramientas analizadas que no cumplen con los principales requisitos comentados anteriormente, tan solo quedan como posibles candidatas para configurar el gestor de ancho de banda el módulo *webmin*-HTB y la aplicación *WebHTB*. Sin embargo, el módulo *webmin*-HTB todavía es una herramienta en periodo de pruebas, lo que supone que todavía no se considera una solución estable y no cuenta con una amplia documentación al respecto.

Por estos motivos, se decide elegir la aplicación *WebHTB* como la herramienta a utilizar para crear el gestor de ancho de banda. *WebHTB* es una solución software para sistemas GNU/Linux que se basa en la disciplina de colas HTB, y aunque su árbol de preferencias se limita a dos niveles, se consideran necesarios para las necesidades del proyecto.

5.- CONFIGURACIÓN DEL SISTEMA

5.1.- PREPARACIÓN DE LA MÁQUINA

En la documentación consultada no se ha encontrado ningún tipo de requisito mínimo a la hora de instalar *WebHTB* en una máquina. Sin embargo, teniendo en cuenta las funciones que debe proporcionar el equipo y en vista a posibles ampliaciones, se cree conveniente que el equipo donde se instale cumpla algunos requisitos. Estos requisitos consisten en un mínimo de 512Mbytes de RAM, un procesador de más de 2GHz, capacidad para utilizar varias interfaces de red a la vez y una capacidad mínima de 80GBytes.

Por motivos de estandarización y seguridad, se recomienda el uso de un servidor DELL, ya que es el proveedor mayoritario a nivel de hardware de la empresa. Con el uso de este modelo de servidor se consigue ubicar físicamente el servidor en el armario de servidores, disponer de asistencia técnica y estandarizar el material informático de la empresa.

Con los requisitos especificados, se estudian distintas opciones en el mercado, y finalmente se opta por la compra de un servidor *Dell Power Edge 1950* con las siguientes características:

- Procesador Quad-Core Xeon 2.8GHz.
- Memoria 4GB 667MHz FBD.
- Dos discos de 160GB SAS 15k 3.5”.
- Alimentación de red redundante.
- Servicio de 3 años de garantía.

5.2.- ELECCIÓN DEL SISTEMA OPERATIVO

5.2.1.- Análisis de sistemas operativos

La elección del sistema operativo debe tener en cuenta las necesidades y los requisitos del sistema y de sus funcionalidades. Debido a la naturaleza de las funciones de gestión de tráfico expuestas anteriormente, el sistema operativo a utilizar debe ser una distribución GNU/Linux. Como no todas las distribuciones en este entorno se adecuan a las necesidades del proyecto, a continuación se muestra un análisis de algunas de ellas.



La distribución Debian o Debian GNU/Linux es una de las más antiguas, y está plenamente mantenida por la comunidad y por voluntarios. Se caracteriza por su alta calidad en cuestión de paquetes y en su estabilidad, ya que el software publicado ha sido exhaustivamente testado. Por otro lado, esta distribución no mantiene una política regular de emisión de versiones, pero suele tener una media de 1 a 3 años entre lanzamientos. Sin embargo, después de una nueva versión, la anterior sigue teniendo versiones de seguridad por un periodo de un año aproximadamente. Es la que sigue siendo más fiel a su política de software libre y la que tiene más derivados. La última versión de esta distribución es Debian 5.0.



El proyecto openSUSE es una distribución comunitaria patrocinada por Novell y que se originó en la antigua SUSE con una fuerte presencia en Europa. OpenSUSE se caracteriza por sus gráficos y por su agradable acabado, tanto en gestores de ventana KDE como en GNOME. También utiliza como base para Novell SUSE Linux Enterprise, un sistema similar al adoptado por Red Hat que patrocina la distribución y el uso de Fedora como base para su versión empresarial. La política de openSUSE es proporcionar actualizaciones de seguridad durante dos años.



CentOS es una distribución de Linux a nivel de empresa totalmente derivada de Red Hat Enterprise Linux.

Esta distribución sigue completamente la política de redistribución de Red Hat y pretende ser totalmente compatible con sus binarios. CentOS está mantenido por un pequeño equipo de desarrolladores cuya principal función es recopilar las fuentes proporcionadas por Red Hat y limpiarlas de marcas y logotipos. Por estos motivos, conserva la estabilidad y la popularidad de su distribución madre, la cual es una de las más aceptadas en el mundo empresarial. El periodo de mantenimiento de seguridad es de hasta siete años. La versión más reciente es CentOS 5.2.

5.2.2.- Conclusiones respecto al sistema operativo

De esta lista se han descartado previamente distribuciones basadas en Mandriva o en Gentoo por ser demasiado complejas tanto de instalar como de gestionar, o por no estar enfocadas al entorno empresarial ni al uso en servidores. Dentro de las que se han seleccionado, podemos mencionar a CentOS como la que más de acerca a las necesidades de la empresa. Además de contar con un gran periodo de mantenimiento de seguridad, el hecho de estar vinculado con Red Hat hace que exista mucha documentación al respecto y que si algún día se considerase migrar el sistema a una distribución Red Hat el impacto sería menor. Además, Red Hat es una distribución oficialmente soportada por los sistemas DELL, el cual es el principal proveedor de la empresa a nivel de hardware.

Paralelamente, se pidió consejo a un antiguo trabajador de la empresa experto en sistemas Linux. Tras estudiar el uso que se quería hacer del sistema y teniendo en cuenta sus conocimientos del funcionamiento de la empresa, llegó a la conclusión de que la distribución que mejor cumplía las necesidades requeridas debía ser una basada en Red Hat como CentOS.

5.3.- INSTALACIÓN Y CONFIGURACIÓN DEL SISTEMA OPERATIVO

Una vez se dispone del servidor enrackado en el armario de servidores, se procede a la instalación del sistema operativo. Tal y como se ha concluido en el apartado anterior, se elige la distribución Centos y la versión v5.2.

Del proceso de instalación se pueden destacar los siguientes puntos:

- El proceso se lleva a cabo a partir del asistente Anaconda.
- Las particiones del sistema se hacen de manera personalizada según los servicios que tiene que soportar.

- El nombre definitivo del servidor es **centos5-gestorA@general-lab.com**.
- Se instala el escritorio GNOME.

El sistema debe soportar funciones y servicios específicos de control de tráfico, con lo que es necesario compilar el *kernel* para configurar ciertos parámetros. Esta compilación se aprovecha para actualizar el núcleo a la última versión disponible y personalizarlo según el hardware instalado.

Durante este proceso de instalación y personalización han ido surgiendo incidencias que se han ido resolviendo. Todo este proceso, así como la identificación y la resolución de las distintas incidencias, se explica de manera detallada en el anexo *A.- Instalación y configuración del sistema operativo*.

Finalmente, se personaliza el sistema. Esta personalización se detalla en el anexo *B.- Personalización del sistema* y consiste en la creación de usuarios para evitar el uso abusivo del súper-usuario *root*, gestionar los servicios que deben arrancar automáticamente e instalar aplicaciones de utilidad para el sistema. Estas aplicaciones son:

- **Webmin**: Permite administrar el sistema a partir de un entorno Web.
- **Wireshark**: Un analizador de protocolos que permite analizar el tráfico que pasa a través del servidor.
- **Cacti**: Un monitorizador del sistema, que se va a utilizar principalmente para monitorizar en tiempo real el tráfico.

5.4.- MANTENIMIENTO DEL SISTEMA

Con el fin de optimizar el funcionamiento del servidor y de sus servicios, se recomienda realizar periódicamente tareas de mantenimiento.

Estas tareas recomendadas consisten principalmente en realizar copias de seguridad, controlar la capacidad de las distintas particiones del sistema y analizar los distintos ficheros de logs para solucionar posibles problemas. Todas estas tareas se detallan en el anexo *C.- Mantenimiento del sistema*.

6.- LA APLICACIÓN *WEBHTB*

6.1.- PROCESO DE INSTALACIÓN DE *WEBHTB*

Una vez el proceso de configuración del sistema está finalizado, se da comienzo al proceso de instalación de la aplicación *WebHTB*. Durante el transcurso de esta instalación han surgido distintas incidencias que ha sido necesario solucionarlas para poder instalar la aplicación de forma correcta.

En el anexo *D.- Manual de instalación de WebHTB* se describe detalladamente todo este proceso y la solución a los problemas surgidos.

6.2.-FUNCIONAMIENTO DE *WEBHTB*

WebHTB es la aplicación que se ha elegido para implantar el gestor de ancho de banda en la empresa. Esta herramienta cumple con todos los requisitos descritos, y no es necesario desarrollar una aplicación personalizada. Esto permite dedicar el tiempo que se utilizaría en desarrollar la nueva aplicación en intentar mejorar y personalizar *WebHTB*, ya que como se verá en este capítulo, esta aplicación todavía está en desarrollo y todavía presenta aspectos a mejorar.

WebHTB es una aplicación Web que permite crear, editar y comprobar de manera ágil y sencilla las clases, colas y filtros necesarios para la gestión de tráfico. Las versiones más recientes prescinden del uso de HTB-tools, lo que facilita la tarea de interpretar el código creado. El entorno Web consiste en un menú de pestañas muy intuitivo y cuidado. Además de permitir crear las reglas necesarias para gestionar el tráfico, también es posible visualizar por pantalla el tráfico que pasa por cada clase en tiempo real.

6.2.1.- Servicios adicionales necesarios

WebHTB se gestiona en entorno Web mediante *Apache*, está construido en lenguaje PHP, Java, Ajax y la base de datos es gestionada con MySQL. Para utilizar *WebHTB* es necesario tener arrancados y configurados algunos servicios:

- **MySQL:** Servicio que utiliza *WebHTB* para gestionar sus bases de datos.
- **PHP:** Servicio que permite interpretar código PHP, utilizado para programación de entornos Web como HTML.

- **Apache (httpd):** Servidor http Web donde corre *WebHTB* y todos los demás servicios con entorno Web. La ruta donde se encuentran estos archivos se especifica en el archivo de configuración de *Apache /etc/httpd/conf/httpd.conf*. Este ruta puede ser cambiada por el administrador editando este fichero, pero por defecto es */var/www/html* (Figura 6.1).

```
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/var/www/html"
```

Figura 6.1.- Copia parcial del fichero */etc/httpd/conf/httpd.conf*

6.2.2.- Creación de los elementos HTB

Con estos servicios arrancados y configurados, es posible utilizar *WebHTB*. En el anexo *E. Manual de usuario de WebHTB* se encuentra un manual detallado sobre el uso de la aplicación.

Antes de empezar a insertar elementos, es importante definir el árbol de preferencias que se quiere conseguir. En la figura 6.2 se muestra un sencillo árbol de ejemplo, donde se tiene la clase *root*, la clase *default*, y un cliente asociado a una clase.

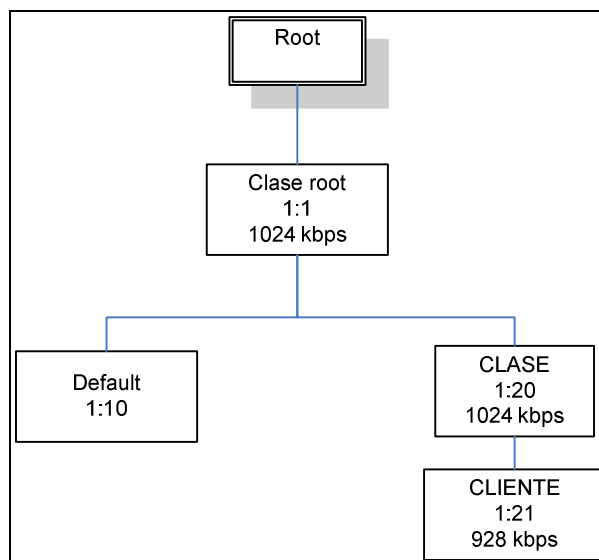


Figura 6.2.- Árbol de preferencias simple de ejemplo

Una vez creadas todas las clases y los filtros necesarios en base al árbol de preferencias, *WebHTB* genera y ejecuta un archivo llamado *qos.sh* en el directorio */tmp*. Este archivo se ejecuta de manera independiente para cada una de las interfaces, y se superponen unos con otros, de manera que tan solo se almacena la configuración de la última interfaz configurada.

En la figura 6.3 se muestra el archivo con el conjunto de elementos HTB correspondiente al árbol de preferencias del ejemplo anterior.

```
#!/bin/sh
DEV=eth1
TC=`which tc`
U32="filter add dev $DEV protocol ip parent 1:0 prio 1 u32"
U32u="filter add dev $DEV protocol ip parent ffff: prio 1 u32"

#Delete previous root qdisc

$TC qdisc del dev $DEV root >/dev/null 2>&1
$TC qdisc del dev $DEV ingress >/dev/null 2>&1

#Add root qdisc
$TC qdisc add dev $DEV handle ffff: ingress; >/dev/null 2>&1
$TC qdisc add dev $DEV root handle 1: htb default 10

#Add root class

$TC class add dev $DEV parent 1: classid 1:1 htb rate 128000kbps ceil 128000kbps
burst 307200k prio 0

#Add default class

$TC class add dev $DEV parent 1:1 classid 1:0x10 htb rate 8kbps ceil 1kbps
$TC qdisc add dev $DEV parent 1:0x10 handle 0x10: pfifo limit 5

#Add class CLASE

$TC class add dev $DEV parent 1:1 classid 1:0x20 htb rate 128kbps ceil 128kbps
burst 307k prio 0

#Add client CLIENTE

$TC class add dev $DEV parent 1:0x20 classid 1:0x21 htb rate 116kbps ceil
128kbps burst 307k prio 0
$TC ${U32} match ip src 128.16.8.0/24 match ip dst 128.0.0.23/32 match ip dport
2034 0xffff flowid 1:0x21
```

Figura 6.3.- Ejemplo del fichero *qos.sh*

De este archivo se puede destacar:

- Se crea un *script* con el mismo nombre para cada una de las interfaces.
- Al inicio del *script* se eliminan todas las clases creadas anteriormente.
- Se utiliza el filtro *u32* y por defecto todos los filtros tienen la misma prioridad (*prio 1*). Esto hace que si diferentes filtros comparten alguna condición y deben estar ordenados, se debe realizar este orden de manera alfanuméricamente.
- Se asigna el *handle ffff:0* a la disciplina de colas *ingress*.
- La clase *default* tiene asignada una disciplina de cola *pfifo* con un *buffer* de 5 paquetes.

- La clase *default* tiene el *handle* 1:10, por lo que las distintas clases empiezan por la numeración 1:20.
- El archivo aparece automáticamente comentado para facilitar su interpretación.

6.2.3.- Parámetros HTB

Antes de comentar los distintos parámetros en *WebHTB*, es conveniente definir las unidades que utiliza el binario *tc* y la disciplina HTB para cuantificar los distintos parámetros. Cabe destacar que son distintos al criterio seguido al inicio del proyecto. Todos estos parámetros se componen de valores en coma flotante seguidos de la unidad correspondiente.

Anchos de banda o capacidades:

- **kbps**: Kilobytes por segundo.
- **mbps**: Megabytes por segundo.
- **kbit**: Kilobits por segundo.
- **mbit**: Megabits por segundo.
- **bps o sin unidad**: Bytes por segundo.

Volumen de datos:

- **kb o k**: Kilobytes.
- **mb o m**: Megabytes.
- **mbit**: Megabits.
- **kbit**: Kilobits.
- **b o sin unidad**: Bytes.

Periodos de tiempo:

- **s, sec o secs**: Segundos completos.
- **ms, msec o msecs**: milisegundos.
- **us, usec, usecs o sin unidad**: microsegundos.

A continuación se muestran, a partir del comando *tc* y la opción *show*, la configuración de las clases y de las disciplinas de colas creadas en el ejemplo (Figuras 6.4 y 6.5).

```
[informatica@centos5-gestorA etc]$ tc class show dev eth0

class htb 1:1 root rate 1024Mbit ceil 1024Mbit burst 300Mb cburst 129408b
class htb 1:10 parent 1:1 leaf 10: prio 0 rate 64000bit ceil 8000bit burst 1607b
cburst 1600b
class htb 1:20 parent 1:1 rate 1024Kbit ceil 1024Kbit burst 307Kb cburst 1727b
class htb 1:21 parent 1:20 leaf 21: prio 0 rate 928000bit ceil 1024Kbit burst
307Kb cburst 1727b
```

Figura 6.4.- Configuración de los parámetros de las clases de ejemplo

```
[informatica@centos5-gestorA etc]$ tc qdisc show dev eth0

qdisc ingress ffff: parent ffff:fff1 -----
qdisc htb 1: root r2q 10 default 10 direct_packets_stat 0
qdisc pfifo 10: parent 1:10 limit 5p
qdisc sfq 21: parent 1:21 limit 127p quantum 1514b
```

Figura 6.5.- Configuración de los parámetros de las disciplinas de colas de ejemplo

La aplicación *WebHTB* deja algunos parámetros libres para que la disciplina de colas HTB los calcule de manera automática. En cambio, se encarga de definir otros en base a pruebas realizadas por el grupo que ha desarrollado la aplicación.

En concreto, *WebHTB* propone valores para los parámetros *burst*, y deja a HTB que defina los de *cburst* y *quantum*.

- **burst:** *WebHTB* tiene la opción de calcular este parámetro si se indica con un 0 en la casilla correspondiente. Este parámetro lo calcula en función del límite de la clase (*ceil*), multiplicándolo por un factor 2,4. El grupo encargado de desarrollar la aplicación ha demostrado empíricamente que este valor de *burst* es óptimo para minimizar la latencia (Ecuación 6.1). Este valor se diferencia del que utiliza la disciplina HTB para calcular el valor de *burst* a partir del *rate* de la clase (6.2).

$$\text{burst [Bytes]} = \lfloor \text{ceil [kBytes/seg.]} \times 2.4 \rfloor \quad (6.1)$$

$$\text{burst[Bytes]} = \text{rate[kBytes/seg.]} + 1600[\text{Bytes}] - 1[\text{Byte}] \quad (6.2)$$

- **cburst:** Este parámetro lo calcula automáticamente la disciplina HTB también en función del límite *ceil*. Este valor no puede ser nunca menor que 1600 Bytes, de tal manera que el valor mínimo de *ceil* se corresponde con *cburst*=1600 Bytes (Ecuación 6.3).

$$\text{cburst[Bytes]} = \text{ceil[kBytes/seg.]} + 1600[\text{Bytes}] - 1[\text{Byte}] \quad (6.3)$$

- **quantum para clases:** La disciplina HTB hace distinción entre el parámetro *quantum* para clases y para colas finales. Para el caso de las clases, HTB define este parámetro en función de $r2q$, de manera que esté comprendido entre MTU y 6000 Bytes (Ecuación 6.4).

$$\text{quantum[Bytes]} = \begin{cases} \text{rate[kBytes/seg.]/10, rate > 15kBps} \\ \text{rate[kByte/seg.]/1, rate} \leq 15kBps \end{cases} \quad (6.4)$$

- **quantum para colas finales:** La aplicación elegida no interviene en el valor de este parámetro, sino que es HTB quien define por defecto que este valor sea de 1514 Bytes. El parámetro *quantum* puede ser como mínimo igual que el MTU. Si para redes como Ethernet este valor es por defecto 1500 Bytes, se escoge un valor cercano al mínimo para asegurar que todos los flujos de una misma clase final sirvan paquetes de una manera ágil.

Los parámetros propuestos por *WebHTB* han sido calculados con la premisa de minimizar la latencia en base a pruebas realizadas por los creadores de la aplicación. En el proyecto se ha aceptado esta propuesta y no se ha entrado a validar ni comprobar si este cálculo es realmente beneficioso para el sistema. De todos modos, siempre es posible definir un valor propio para cada clase y no dejar que la aplicación defina ningún parámetro por si sola.

6.2.4.- Visualización del tráfico

Una característica importante de *WebHTB* es que es posible visualizar el tráfico que pasa por cada una de las clases activas en tiempo real. Esto permite comprobar el funcionamiento y la carga a la que es sometida la red y el gestor, con el fin de optimizar su uso y solucionar posibles problemas de configuración.

CLASS	CLIENT	SPEED	BANDWIDTH	LIMIT	TOKENS	CTOKENS
BARCELONA16		414.63	1024	1024	2377866	-7388
	1.WinlabNet_bcn16	48.75	920	1024	2669157	12817
	2.Otros_bcn16	374.40	104	1024	-56661	4219
MANRESA		0	1024	1024	2398437	13183
BARCELONA0		0	1024	1024	2398437	13183
MALAGA1		0	1024	1024	2398437	13183
default		0	1024	1024	23615383	13183

Figura 6.6.- Ventana para visualizar el tráfico en *WebHTB*

El valor de este tráfico se calcula utilizando una ventana temporal de 1 segundo y con un retraso aproximado de 5 segundos. Esto hace que los cambios en la capacidad de la red no se contemplen de forma inmediata en la ventana de tráfico de *WebHTB*.

En la figura 6.6 se puede ver la ventana que utiliza la aplicación para mostrar el tráfico en tiempo real. En esta ventana tan solo se muestran las clases activas, es decir, las que tienen tráfico asociado en el momento de de la captura. El resto de clases aparecen sin desplegar.

Para cada clase activa, se muestra la capacidad del enlace que está utilizando, el ancho de banda que tiene asignado, el máximo ancho de banda del que puede disponer, el número de *tokens* del *buffer burst* y el número de *tokens* del *buffer cburst*.

6.2.5.- Contribución al desarrollo de *WebHTB*

Una parte significativa de este proyecto se ha dedicado a intentar mejorar la aplicación *WebHTB*. Durante el proceso de pruebas y testing se detectó que para que la aplicación fuera más completa y útil para este proyecto, era conveniente depurar algunos errores y mejorar algunos servicios.

Esta contribución ha sido posible gracias al contacto continuo con el grupo de desarrolladores de la aplicación a través de correo electrónico. Debido a esto, hemos sido los encargados de testear las diferentes versiones antes de que fueran publicadas, dando paso a actualizaciones y nuevas versiones.

Más concretamente, se ha contribuido a las siguientes mejoras y ampliaciones:

- La aplicación tan solo muestra en tiempo real las clases activas, y oculta las que no están gestionando ningún tipo de tráfico. Esto permite simplificar y clarificar la ventana que muestra el tráfico.
- Se soluciona un error que no permitía filtrar por puertos que no estuvieran asignados a una dirección IP. Esta mejora permite filtrar independientemente por puertos, sin necesidad de asociar estos puertos a una dirección o rango IP.
- Se soluciona un error que no permitía añadir más de un puerto por filtro, tanto en origen como en destino. De esta manera, se simplifica el proceso de crear filtros, ya que con este cambio ya no es necesario un filtro para cada puerto, sino que varios puertos pueden estar asignados a un mismo filtro si se separan por comas.

- Se habilita una pestaña en la interfaz Web que permite arrancar o parar el servicio. Anteriormente era necesario hacerlo a través de la consola de comandos (`service webhtb start/stop/restart`)
- Se traduce el entorno Web al idioma español. La aplicación estaba definida en varios idiomas (inglés, portugués y rumano). Después de esta traducción, es posible utilizar *WebHTB* en español.

6.2.6.- Soluciones alternativas con el uso de iptables

Para poder seguir utilizando *WebHTB* antes de que se solucionaran los distintos problemas y errores que se iban descubriendo, se estudió el uso conjunto de *WebHTB* e *iptables*. Esta posibilidad permite utilizar el *firewall* de sistemas Linux para marcar paquetes, para después tratarlos en *WebHTB* según su marca. De esta manera es posible clasificar el tráfico de maneras tan complejas como se requieran sin necesidad de utilizar los filtros de *WebHTB*.

Para analizar esta alternativa, se estudia el caso particular de clasificar a una clase en concreto el tráfico cuyo puerto destino sea el 23.

```
[root@centos5-gestorA]# iptables -t mangle -A POSTROUTING -p tcp --dport 23 -j MARK --set-mark 1
```

Ahora tan solo habría que reiniciar el servicio *iptables* para que las reglas tuvieran efecto y utilizar el campo *mark* en lugar de puerto destino como se muestra en la figura 6.7.

Figure 6.7 shows a web interface titled "AÑADIR CLIENTE A LA INTERFAZ eth1". It contains a red warning message: "¡IMPORTANTE! No es posible utilizar espacios en blanco y separe los puertos con comas. Los campos rojos son obligatorios!". Below the warning, there is a dropdown menu for "ELIJA UNA CLASE:" with "CLASE" selected. A table of configuration fields follows, with the "MARK" column circled in red. The table has columns: CLIENT, BANDWIDTH, LIMIT, BURST, PRIORITY, UPLOAD, and MARK. The values are: CLIENT (empty), BANDWIDTH (1024), LIMIT (1024), BURST (0), PRIORITY (3), UPLOAD (empty), and MARK (1). Below the table, there are fields for SRC IPS, SRC PORTS, DST IPS, and DST PORTS. At the bottom, there is a button "Presione aquí para un nuevo src, dst.." and two buttons "GUARDAR" and "BORRAR".

CLIENT	BANDWIDTH	LIMIT	BURST	PRIORITY	UPLOAD	MARK
	1024	1024	0	3		1

Figura 6.7.- Ejemplo de filtrado utilizando *mark*

Este tan solo es un ejemplo de las posibilidades que presenta el uso conjunto de *WebHTB* e *iptables*. Puede llegar a ser útil en caso de que no sea posible clasificar el tráfico con los filtros *WebHTB*.

7.- PROCESO DE PRUEBAS Y TESTING

7.1.- PRUEBAS EN ENTORNOS BÁSICOS SOBRE ETHERNET

Una vez se ha completado el proceso de instalación, es necesario comprobar si el comportamiento del gestor es el esperado. En este primer apartado no se busca analizar si se gestionan correctamente los recursos, ni tampoco comparar el comportamiento de las distintas disciplinas de colas, si no comprobar si el servidor es capaz de gestionar el tráfico de salida de sus interfaces de red en un entorno amigable.

7.1.1.- Red de pruebas básica sobre Ethernet

7.1.1.1.- Configuración de la red

Para hacer esta primera comprobación se configura un banco de pruebas que consta de unos ordenadores clientes, el gestor y un servidor, unidos entre sí a partir de la tecnología Ethernet. Por sus altas prestaciones en cuanto a capacidad y a su transmisión bidireccional se considera que esta tecnología favorece la gestión de ancho de banda. Previamente se ha configurado las distintas tarjetas de red para que trabajen a una velocidad de 100Mbps. Todos los elementos de la red disponen de sistema operativo Windows XP, a excepción del servidor centos5-gestorA (Figura 7.1)

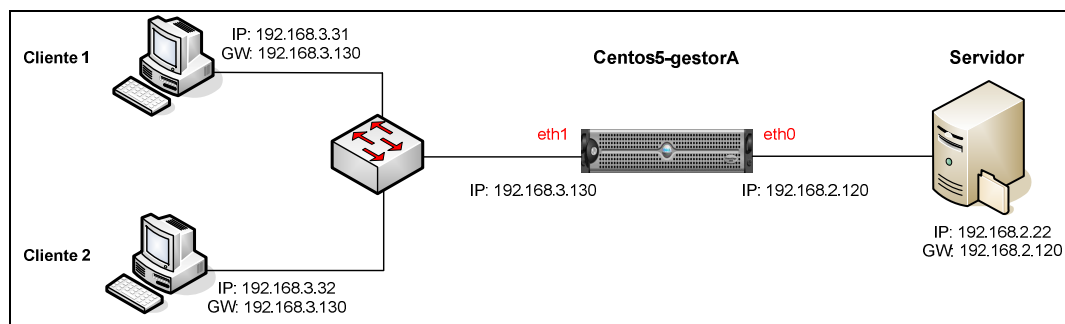


Figura 7.1.- Red de pruebas básica sobre Ethernet

Para simplificar el proceso de testing, durante todas las pruebas realizadas en este apartado los clientes generan tráfico al servidor en sentido ascendente. Esto supone que solo será necesario gestionar el tráfico que sale por la interfaz eth0. Si se quisiera comprobar la otra interfaz de red, habría que cambiar el sentido del tráfico y configurar la interfaz eth1 de manera simétrica.

Para todo el proceso de testing se utiliza la herramienta *Network Traffic Generator and Monitor*, una aplicación que genera tráfico de una máquina que actúa de cliente a otra que lo hace de servidor.

En la figura 7.1, así como en los posteriores esquemas de red, se muestran los distintos elementos que forman parte del banco de pruebas. Cada uno de estos elementos está identificado con su nombre y con la configuración de sus interfaces de red.

En todos estos esquemas, los clientes están conectados al gestor mediante un switch. En primera instancia se pensó unirlos a través de un router y hacer que cada cliente correspondiera a una red diferente, pero debido a que *WebHTB* permite diferenciar entre direcciones IP, es posible gestionar el tráfico de cada cliente de manera inequívoca, como si cada uno correspondiera a un rango IP distinto.

7.1.1.2.- Creación del árbol de preferencias

Una vez configurada la red y comprobada la conexión entre los diferentes elementos, se diseña un árbol de preferencias como el de la figura 7.2 para llevar a cabo las pruebas. En este árbol se quiere representar una posible situación en la empresa, donde se tienen varios centros o clientes y se quieren gestionar servicios reales con diferentes prioridades.

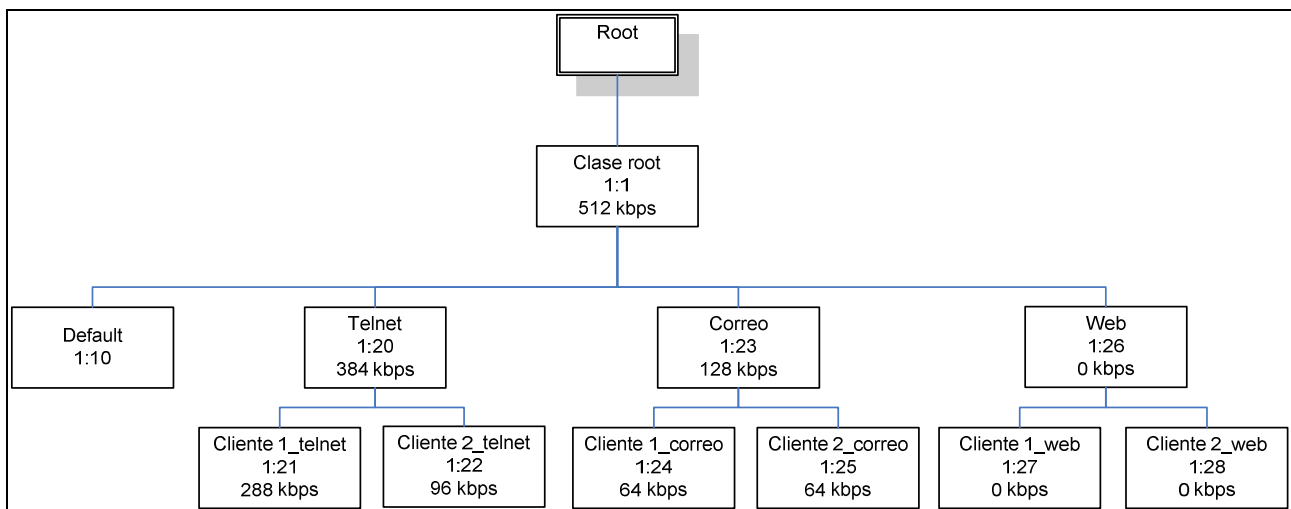


Figura 7.2.- Esquema del árbol de preferencias de pruebas sobre Ethernet

Durante todo el proceso de pruebas se lleva a cabo el criterio de limitar el ancho de banda de la interfaz de salida del gestor a un máximo de 512kbps. Esta limitación se debe a que en pruebas posteriores se utilizará una conexión ADSL con una velocidad real de subida de 512kbps. Con la intención de facilitar este proceso manteniendo el mismo esquema de configuración, se decide limitar en todas las pruebas el ancho de banda a esta capacidad.

Por defecto, la aplicación *WebHTB* crea la clase *root* con un ancho de banda de 1Gbps. Con el fin de facilitar y estandarizar el uso del sistema, se creió conveniente incluir en la aplicación *WebHTB* la posibilidad de modificar los parámetros de la clase *root* para poder limitar el enlace.

Como se ha hecho en otras ocasiones, se propuso esta mejora al grupo de desarrollo de la aplicación y se colaboró en las modificaciones necesarias. Finalmente se creó una nueva pestaña donde era posible configurar distintos parámetros de la clase root, entre ellos *rate* y *ceil*, dando paso a la versión *WebHTB* v2.3.

En el árbol de preferencias de la figura 7.2 se definen tres clases asignadas a tres servicios con diferentes prioridades y capacidades (clases servicio), y todos ellos pueden disponer de todo el ancho de banda en el caso de que ningún otro servicio más prioritario lo esté utilizando (*ceil*=512kbps). Las reglas de los filtros de cada clase se encuentran resumidas en el diagrama de la figura 7.3.

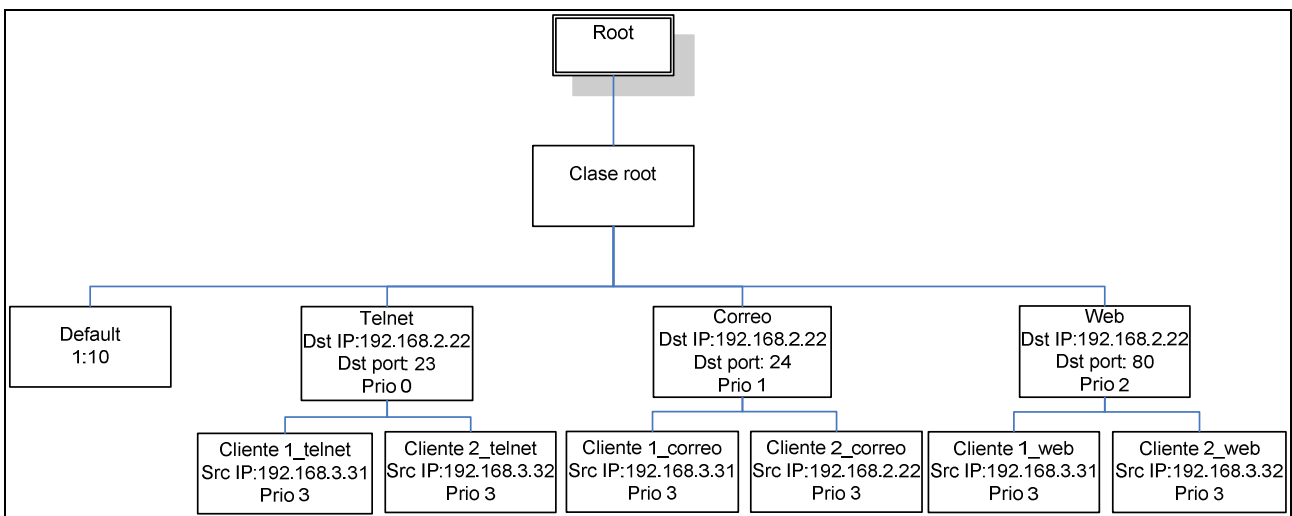


Figura 7.3.- Configuración de los filtros de ejemplo

El servicio TELNET es el más prioritario, y utilizará las $\frac{3}{4}$ partes del ancho de banda siempre que lo necesite. Por otro lado, el servicio CORREO es menos prioritario que el anterior, y tiene a su disposición $\frac{1}{4}$ parte del ancho de banda total. Finalmente, el servicio WEB es el menos prioritario de todos, y tan solo utilizará el ancho de banda excedente. La clase DEFAULT es la que gestiona el tráfico que no se corresponde con ninguno de los casos anteriores, y viene configurada por defecto por la aplicación.

Por lo que se refiere a los clientes, se asignan dos clases hijas a cada clase servicio (clases cliente). De la misma manera que se ha hecho con las clases servicio, cada clase cliente tiene un ancho de banda asignado distinto según el servicio al que corresponda, pero todas ellas pueden disponer de toda la capacidad disponible.

7.1.2.- Resultados con configuración básica sobre Ethernet

En este apartado se van a mostrar las pruebas que se han realizado para comprobar el funcionamiento del gestor en una red bajo tecnología Ethernet. A causa de que una prueba donde se comprobara el comportamiento de todos los servicios y de todos los clientes daría como resultado un gráfico demasiado denso y complejo de interpretar, se han dividido entre pruebas entre clientes y pruebas entre servicios.

Para presentar los resultados de manera visual, se ha utilizado la aplicación *wireshark*. Es una aplicación muy útil porque permite representar de manera gráfica todos los paquetes que pasan a través de una interfaz. Además de esta, existen otras maneras de comprobar el funcionamiento local del gestor. Existe una herramienta llamada *ethloop* desarrollada por el mismo equipo que trabajó en la creación de HTB. *Ethloop* [33] es un generador de paquetes y una herramienta de medida, en el que se puede simular el funcionamiento de las reglas HTB utilizando la interfaz de red local *lo*, sin necesidad de generar tráfico real ni estar conectado a ninguna red.

7.1.2.1.- Prueba entre clientes

La prueba entre clientes consiste en comprobar que los clientes de una misma clase se reparten el ancho de banda correctamente y gestionan el tráfico excedente según se ha configurado. En este ejemplo en concreto se ha elegido la clase correspondiente al servicio TELNET, y se ha monitorizado la interfaz *eth0* para diferentes situaciones de tráfico (Figura 7.5).

Representación de resultados

Todos los gráficos conseguidos a partir de datos extraídos de *wireshark* son presentados de la misma manera. En primer lugar, se tiene una leyenda donde se relaciona cada tipo de tráfico con un color. Otra característica es que el gráfico se divide en distintos tramos separados por flechas rojas, donde cada tramo corresponde a unas condiciones diferentes. Encima de cada tramo, se tiene información sobre el tipo y la cantidad de tráfico generados.

Por ejemplo, el gráfico de la figura 7.4 se divide en tres tramos, que corresponden con tres distribuciones de tráfico distintas. Siguiendo con el ejemplo, en el primer tramo el cliente 1 está generando un flujo de tráfico TELNET a una velocidad de 1Mbps. Naturalmente, la interfaz tan solo podrá servir un máximo de 512kbps, pero en ocasiones es conveniente generar más tráfico del que se puede desencolar para someter a la red a situaciones de congestión.

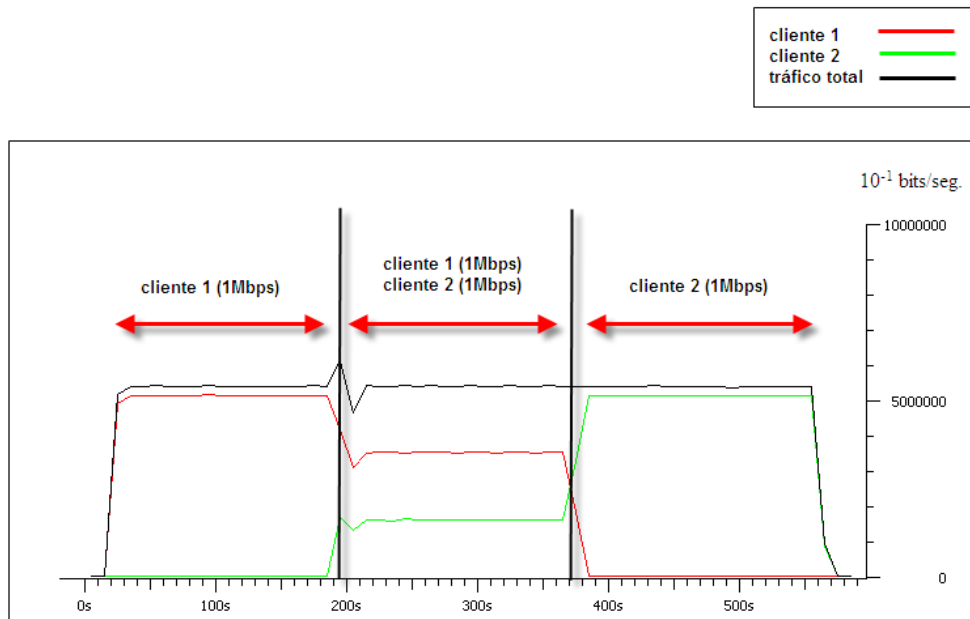


Figura 7.4.- Gráfico de prueba entre clientes sobre Ethernet

En lo referente al comportamiento del gestor, en esta figura se puede observar como en el primer tramo tan solo genera tráfico el cliente 1, lo que hace que disponga todo el recurso. Ya en el segundo tramo, generan tráfico ambos clientes, y se comprueba como se reparten el ancho de banda total según se ha configurado en el árbol de preferencias (3/4 del recurso para el cliente 1 y el resto para el cliente 2). Finalmente, cuando el cliente 1 cierra su conexión con el servidor, el cliente 2 aprovecha todo el enlace. Se observa que durante los diferentes tramos la ocupación total del enlace es constante y se corresponde con el límite fijado de 512kbps.

7.1.2.2.- Prueba entre servicios

De la misma manera, en la figura 7.5 se representan diferentes ejemplos de gestión de tráfico entre servicios y sin tener en cuenta el cliente que lo genera, y se monitorizan en la interfaz eth0.

Con los resultados de esta prueba, se puede observar como el servicio WEB tan solo puede disponer de ancho de banda cuando ninguno de los servicios más prioritarios lo estén utilizando, como sucede en el segundo y el tercer tramo. En el primero, aunque este servicio esté generando tráfico, el servicio TELNET dispone de todo el ancho de banda disponible, pero una vez este servicio cesa su flujo, el WEB aprovecha para utilizar todo el recurso. En el tercer tramo se comprueba como el servicio CORREO dispone de su capacidad asignada siempre que lo necesite, con lo que al tráfico WEB se le asigna el tráfico excedente. En el momento en el que cualquier otro servicio más prioritario requiera este ancho de banda, como sucede en el cuarto tramo, el tráfico WEB es nulo.

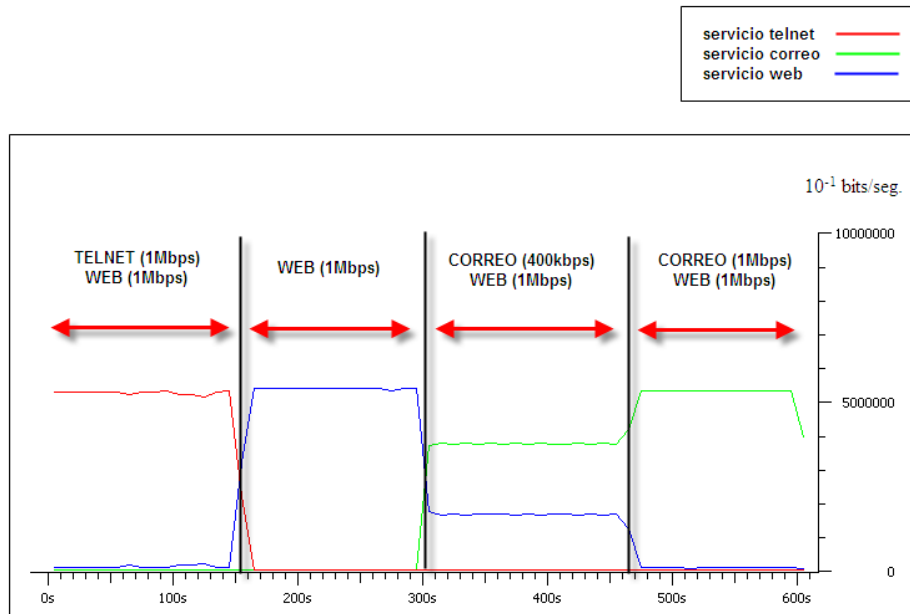


Figura 7.5.- Gráfico de prueba entre servicios sobre Ethernet

7.2.- PRUEBAS EN ENTORNOS REALES SOBRE ADSL

Llegados a este punto, se puede asumir en base a las pruebas realizadas que el gestor es capaz de gestionar el tráfico de salida de una interfaz en una red basada en tecnología Ethernet. En este apartado se va a comprobar este mismo funcionamiento pero ya no en la red básica utilizada anteriormente, sino en entornos más parecidos al objetivo final. Para emular este entorno se va a utilizar una línea ADSL con una capacidad real de 512kbps⁵ de subida. En este caso se va a analizar la gestión que hace el gestor de este recurso, es decir, como es capaz de gestionar un enlace limitado y compartido.

Para realizar estas pruebas se sigue conservando el árbol de preferencias creado en el apartado anterior, y el tráfico generado continuará siendo en sentido ascendente (cliente→servidor). Por otro lado, según la ubicación física del gestor se puede estar hablando de dos tipos de gestión. Si lo colocamos antes del recurso compartido se trata de gestión en origen. En cambio, si se ubica detrás del recurso se habla de gestión en destino.

7.2.1.- Gestión en origen de un enlace ADSL

El banco de pruebas utilizado para comprobar el comportamiento del gestor en entornos con tecnología ADSL es algo más complejo que el anterior. En esta configuración de red la interfaz eth0 del gestor, que es la que se quiere gestionar, está conectada a un router de acceso ADSL.

⁵ Debido a factores externos, esta velocidad de subida puede llegar a variar algunos kbps. En ningún caso esta diferencia ha superado los 10kbps durante le proceso de pruebas.

Por otro lado, el servidor se encuentra al otro extremo de Internet. Para que haya conexión entre los clientes y el servidor, es necesario hacer sendos NAT tanto en el router de acceso del extremo de los clientes como en el del servidor (Figura 7.6).

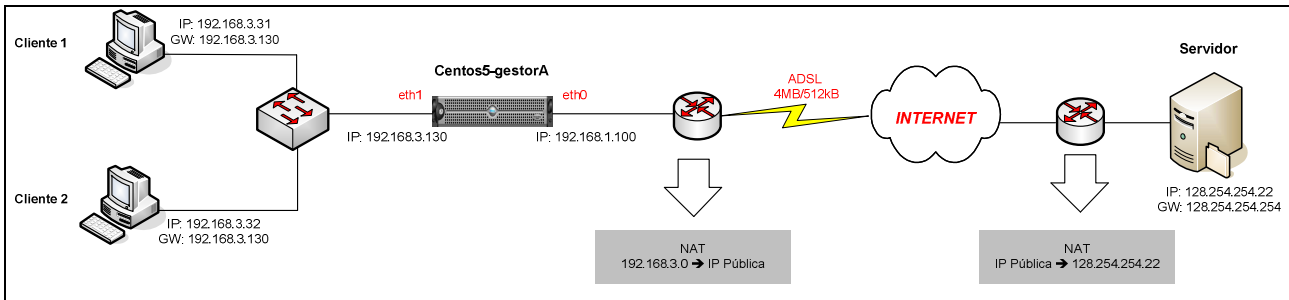


Figura 7.6.- Red de prueba con gestión en origen sobre ADSL

Como consecuencia de estos NAT, todo el tráfico de salida hacia Internet tiene como dirección IP origen la dirección estática del router de acceso. Para tener conexión con el servidor desde cualquiera de los clientes, este tráfico debe tener como dirección IP destino la IP pública definida para el servidor de datos. Esto es un punto a tener en cuenta a la hora de monitorizar el tráfico que pasa por cada interfaz.

En este nuevo banco de pruebas, el único parámetro de los filtros que hay que cambiar es la dirección IP destino de los paquetes, ya que tanto la dirección IP origen y el puerto destino se mantienen. En lo que se refiere a la monitorización, se vuelve a utilizar la interfaz de salida del gestor (eth0). Otra modificación importante es que, como esta conexión ADSL tiene una capacidad real máxima de subida de 512kbps, no será necesario limitar la capacidad de la clase *root* a esta velocidad, tal y como se ha hecho para el caso anterior de tecnología Ethernet.

Cuando se gestiona el tráfico en origen, los paquetes llegan al router de acceso en el mismo orden en que han sido descolados por la interfaz eth0 del gestor. Debido a que este router no mantiene ningún criterio para controlar el tráfico, el enlace ADSL se debería comportar según se haya especificado en el sistema de gestión.

7.2.2.- Comportamiento del sistema con gestión en origen

Una vez se vuelve a comprobar la conexión entre los distintos elementos de la red, se ejecutan las mismas pruebas que en el apartado anterior. Estas pruebas consisten en someter al sistema a situaciones de congestión de red y comprobar si se gestiona correctamente el ancho de banda de la conexión ADSL, tanto entre clases cliente como entre clases servicio. Al hacerlo, se observa como en ninguno de los casos la ocupación del enlace ADSL es la esperada.

Por un lado, el gestor es capaz de identificar correctamente a que clase corresponde cada paquete. Esto se puede ver a través de la ventana que muestra el tráfico en tiempo real asociado a cada clase en la aplicación *WebHTB*. Pero en cambio, no aplica las reglas correctamente. Más concretamente, se observa que el sistema de gestión reparte el tráfico de manera equitativa para todas las clases, es decir, que funciona de la misma manera que si el servicio estuviera detenido.

Para intentar identificar la causa de este comportamiento, se configura un nuevo banco de pruebas con una estructura de red que emula la red sobre ADSL con tecnología Ethernet (Figura 7.7). En este nuevo esquema se han sustituido los dos routers ADSL por un PC con sistema operativo Centos5 (Centos5-gestorB). De esta manera, el enlace 1 corresponde al anterior enlace entre la interfaz eth0 del gestor y el router ADSL, y el enlace 2 corresponde con la conexión ADSL al servidor.

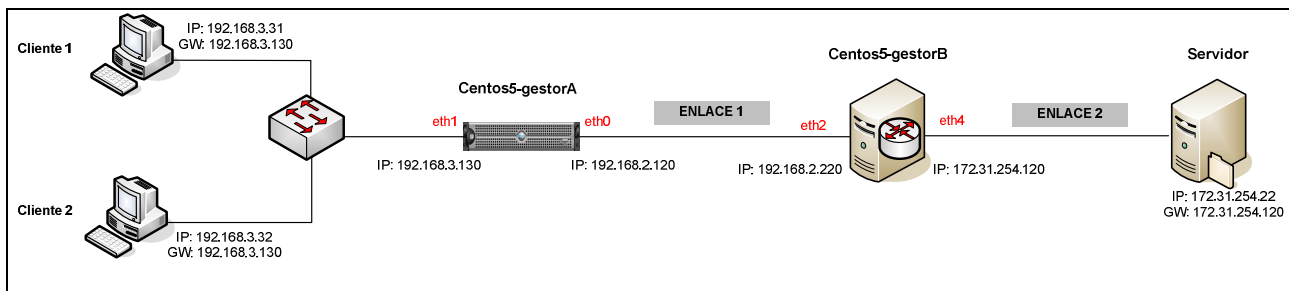


Figura 7.7.- Red de pruebas con diversos enlaces

Cabe destacar que ha sido necesario cambiar la configuración de red del servidor para así poder conservar las reglas y los filtros. Por su lado, el nuevo servidor centos5-gestorB ha sido configurado siguiendo los pasos del anexo *A.- Instalación y configuración del sistema operativo*.

Una vez configurada la nueva red, si se vuelve a hacer la misma prueba y se monitoriza el enlace 1 se observa como en este caso el gestor si que es capaz de gestionar el tráfico de salida de la interfaz eth0. En este punto, se intenta identificar la causa del problema en base a las principales diferencias entre las dos redes, como son el cambio de direccionamiento IP de origen (NAT) y el límite del ancho de banda del enlace 2.

7.2.2.1.- Análisis del cambio de dirección de origen (NAT)

En un primer momento se pensó que el problema podría estar ocasionado por el cambio de dirección IP origen que se efectúa en el router ADSL. Por este motivo, en esta nueva máquina que hace las veces de router se configura el mismo NAT.

El cambio de direccionamiento IP en sistemas Linux se hace usando la herramienta *iptables*. En la figura 7.8 se muestra un posible fichero para hacer este cambio.

```
#!/bin/sh

DEV=eth4
SOURCE=192.168.3.0/24
DEST=172.31.254.22/32
NEW_SOURCE=172.31.254.120

iptables -t nat -A POSTROUTING -o eth4 -p tcp -s $SOURCE -d $DEST -j SNAT --to
$NEW_SOURCE
```

Figura 7.8.- Contenido del *script* para hacer un NAT *nat.sh*

En esta nueva situación se vuelven a realizar pruebas entre clientes y entre servicios, teniendo en cuenta que al hacer el NAT se debe cambiar la dirección IP de origen de los filtros. En los resultados se observa que en este caso si que se consigue una gestión del enlace, demostrando que el comportamiento del gestor es el mismo tanto si el router hace NAT como si no lo hace. Con esto se descarta que el problema sea causado por el cambio de la dirección IP de origen.

7.2.2.2.- Análisis de la relación entre enlaces

Otra de las diferencias entre las dos redes es la capacidad del enlace 2, que anteriormente estaba limitado por la conexión ADSL y ahora lo está por el protocolo Ethernet. Debido a que el PC centos5-gestorB no dispone de la aplicación *WebHTB*, se crea y se ejecuta un *script* que contiene los comandos necesarios para crear los elementos de control de tráfico (Figura 7.9).

```
#!/bin/sh

DEV=eth4
RATE=560kbit
CEIL=560kbit

#Eliminamos las clases previas
tc qdisc del dev $DEV root
tc qdisc del dev $DEV ingress

#Creamos la qdisc root
tc qdisc add dev $DEV handle ffff: ingress
tc qdisc add dev $DEV root handle 1: htb

#Creamos la clase root
tc class add dev $DEV parent 1: classid 1:1 htb rate $RATE ceil $CEIL

#Creamos la clase
tc class add dev $DEV parent 1:1 classid 1:20 htb rate $RATE ceil $CEIL

#Creamos el filtro
tc filter add dev $DEV protocol ip parent 1:0 prio 1 u32 match ip dst
172.31.254.22/32 flowid 1:20
```

Figura 7.9.- Contenido del *script qos_start.sh*

Al limitar el ancho de banda del enlace 2 a una capacidad de 512kbps, similar a la de la conexión ADSL, se observa que el sistema vuelve a actuar como si el servicio estuviera parado, es decir, que reparte equitativamente el tráfico entre las distintas clases. Esto hace llegar a la conclusión de que el posible origen de este problema esté en la relación de los distintos enlaces de la red del banco de pruebas.

7.2.2.2.- Resolución del problema debido a la capacidad de la clase *root*

Una vez encontrada la posible causa del problema, se comprueba que para que el sistema de gestión funcione correctamente debe existir una cierta relación entre los distintos enlaces que forman la red, desde el origen del flujo hasta el destino final.

En este banco de pruebas que se ha implementado se tiene el caso concreto de una red con dos enlaces críticos (Figura 7.7). En este punto se van a analizar los distintos casos que se pueden dar en cuanto a relación de anchos de banda para el caso de dos enlaces, y se va a generalizar para cualquier configuración de red. En la tabla de la figura 7.10 se puede ver el comportamiento del sistema de gestión para distintas relaciones.

ENLACE 1 (kbps.)	ENLACE 2 (kbps.)	GESTIÓN
No limitado	No limitado	✓
No limitado	512	✗
512	512	✗
512	480	✗
512	560	✓

Figura 7.10.- Tabla de relación entre enlaces

Partiendo de la tabla anterior, se representan gráficamente situaciones similares en el caso de una prueba entre servicios (Figura 7.11). En todos los casos, se tiene un flujo de la clase TELNET y otro de la clase CORREO congestionando el enlace a una velocidad de 1Mbps cada uno. En base al árbol de preferencias definido, el sistema debería gestionar este enlace repartiendo $\frac{3}{4}$ partes a la clase TELNET y $\frac{1}{4}$ a la clase CORREO.

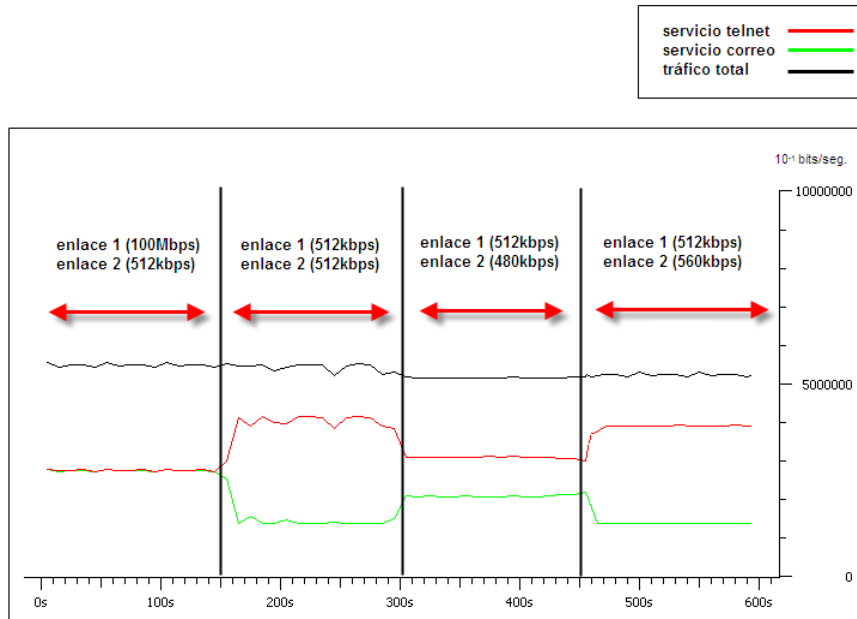


Figura 7.11.- Gráfico de relación entre enlaces

Según los resultados de la figura 7.11, cuando el ancho de banda del enlace 1 es mayor que el del enlace 2, el servidor no es capaz de gestionar el tráfico según su configuración previa. Esta situación se da en el primer y en el tercer tramo. En el primero, el ancho de banda disponible se reparte a partes iguales entre los dos servicios. En el tercero, el servicio más prioritario ocupa más ancho de banda, pero no se respeta la relación establecida en el árbol de preferencias. Cuando los dos anchos de banda son iguales, como pasa en el segundo tramo, el sistema empieza a gestionar el tráfico correctamente, pero la precisión es baja. Finalmente, en el último de los tramos el gestor reparte el enlace correctamente y mejora notablemente la precisión.

A partir de estos resultados, se observa que la relación que se debe cumplir entre ambos enlaces es que el enlace 2 sea mayor que el enlace 1. Si se generaliza este comportamiento, equivale a decir que la capacidad de la interfaz de red donde se aplica la gestión debe ser menor que el enlace más restrictivo de toda la red (Ecuación 7.1).

$$\boxed{\text{capacidad root} < \text{capacidad más restrictiva}} \tag{7.1}$$

Se ha buscado información y se ha consultado bibliografía especializada, pero no se han conseguido referencias que traten este comportamiento. Tampoco la base teórica que se ha adquirido a lo largo del proyecto ofrece respuestas robustas. Se han analizado los paquetes que pasan a través de las diferentes interfaces del gestor con el analizado de protocolos *wireshark*, y no se ha descubierto ningún flujo fuera de lo normal que de lugar a este fenómeno.

Esta limitación puede ser un inconveniente importante si se tiene en cuenta que nunca va a ser posible utilizar todo el ancho de banda del que se dispone. En cambio, en la práctica se ha comprobado que un margen de 5bps entre enlaces es suficiente, lo que supone unas cifras casi despreciables para anchos de banda del orden de varios Mbps.

7.2.3.- Gestión en destino de un enlace ADSL

Esta nueva ubicación del gestor en la red implica algunas modificaciones en el árbol de preferencias de ejemplo. Al hacerse la gestión después de atravesar el enlace ADSL, el gestor no será capaz de diferenciar entre los distintos clientes a causa del NAT que se ejecuta en el router de acceso ADSL. Debido a esta modificación, en este punto se realizarán tan solo pruebas entre servicios y se pondrá como dirección IP origen la dirección estática del router de acceso (Figura 7.12).

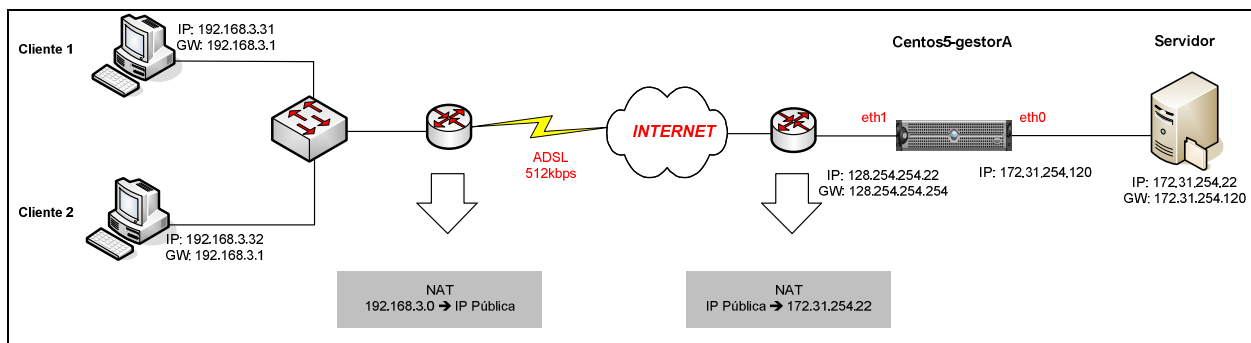


Figura 7.12.- Red de prueba en destino sobre ADSL

A diferencia de la gestión del tráfico en origen, en este caso los paquetes desencolados por la interfaz del gestor no llegan al router ADSL. Con gestión en destino es necesario acudir a los mecanismos de control de flujo para entender como se gestiona el enlace compartido.

7.2.4.- Control de flujo

En la gestión en origen, era sencillo entender que controlando la salida de la interfaz eth0 se gestionaba también la conexión ADSL, ya que la salida de esta interfaz correspondía con el acceso al recurso y los paquetes que no eran servidos esperaban su turno en la interfaz del gestor. En cambio, en este tipo de gestión en destino es el recurso el que realmente gestiona el ancho de banda disponible gracias a los mecanismos de control de flujo.

Las conexiones TCP tienen distintos mecanismos para que el emisor determine la capacidad de la red al iniciar una conexión y para ajustarse a los cambios de capacidad de manera dinámica [21]. Estos métodos se basan en la observación y en el cálculo de algoritmos basándose principalmente en los ACK, los RTT (Round Trip Time) y en el tamaño de las ventanas de congestión.

Los principales métodos son *slow start* y *additive increase* para conocer el ancho de banda disponible al iniciar una conexión, y *fast recovery* y *fast retransmit* para reaccionar a los cambios de capacidades. El principio fundamental de estos métodos es que TCP necesita provocar pérdidas de paquetes para estimar el ancho de banda disponible.

En el gráfico de la figura 7.13 se puede comprobar el funcionamiento de estos mecanismos de control de flujo extraídos de una conexión real. Este ejemplo se corresponde con una conexión entre un cliente y el servidor monitorizada desde el extremo del cliente, y se pueden diferenciar distintas etapas o tramos.

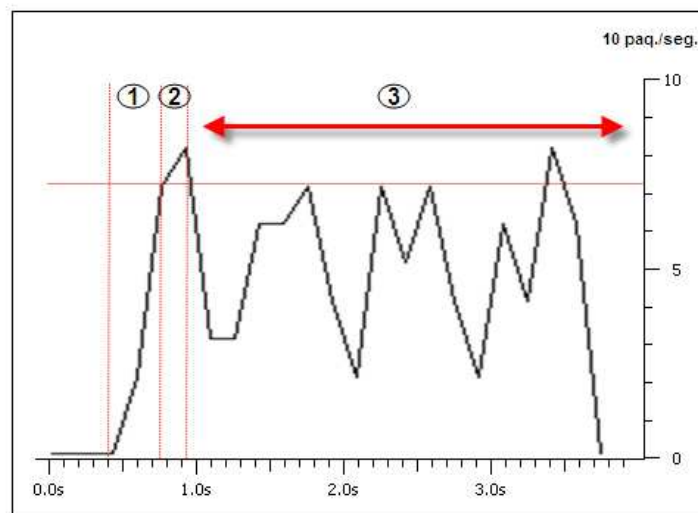


Figura 7.13.- Gráfico de mecanismos de control de congestión

- **Tramo 1:** Este primer tramo se corresponde con el mecanismo *slow start*. En este caso, el emisor (cliente) pretende conocer el ancho de banda disponible a partir de un protocolo en el que se aumenta el envío de paquetes de manera exponencial.
- **Tramo 2:** Este tramo se corresponde con el mecanismo *additive increase*. Este protocolo se utiliza cuando el número de paquetes enviados está más cerca del máximo, y consiste en aumentar el número de paquetes enviados de manera lineal hasta que se tiene la pérdida de un paquete.
- **Tramo 3:** Este último tramo se corresponde con los mecanismos de control de congestión de tráfico. Se trata de protocolos como *fast recovery* y *fast retransmit* que a partir de los paquetes perdidos son capaces conocer en cada momento el ancho de banda disponible en el enlace.

7.2.5.- Comportamiento del sistema con gestión en destino

De la misma manera que con gestión en origen, se realiza una prueba entre servicios. Para realizar esta prueba se mantiene el árbol de preferencias de la figura 7.2. El gráfico de la figura 7.14 muestra los resultados obtenidos a partir de monitorizar el tráfico de la conexión ADSL a la salida de la interfaz del cliente 1.



Figura 7.14.- Gráfico de pruebas entre servicios con gestión en destino

Con estos resultados se puede ver como el servicio WEB tan solo tiene asignado el ancho de banda excedente del resto de servicios. Durante toda la prueba, la suma de anchos de banda utilizados es constante y se corresponde con el límite de 512kbps. Finalmente, en el último tramo se observa la relación entre los distintos servicios mostrada en el árbol de preferencias, con lo que se comprueba que con el sistema en destino se gestiona el enlace ADSL de manera satisfactoria.

7.3.- ANÁLISIS DE COLAS FINALES

El uso de HTB permite configurar la cola final o *leaf queue* entre todas las posibles colas simples. Esta cola final es la que se encarga de desencolar el tráfico de todos los flujos de una misma clase. Sin embargo, la aplicación *WebHTB* tan solo deja escoger entre las más comunes, que son PFIFO y SFQ (también permite usar ESFQ, pero no es más que una ampliación de SFQ).

Por defecto, ya sea en la disciplina de colas HTB o en la mayoría de routers, la cola final es PFIFO. En esta disciplina de cola los paquetes son desencolados en el mismo orden que fueron encolados y apenas permite diferenciar entre tipos de tráfico. La otra posible alternativa es la disciplina SFQ o ESFQ. Este tipo de cola configura una cola FIFO para cada flujo, y basa en un algoritmo que busca equidad y justicia entre todos ellos.

7.3.1.- Según latencia

Como se ha comentado en capítulos anteriores, la latencia es el retardo introducido por la red entre el envío de un paquete y su recepción, y depende principalmente del grado de congestión de la red. La latencia en una red se puede descomponer entre:

- **Latencia de transmisión:** La velocidad de los circuitos de la red.
- **Latencia de inserción:** La velocidad a la cual los routers pueden redireccionar los paquetes.
- **Registro de latencia:** La velocidad con la que los paquetes llegan al router.
- **Latencia de propagación:** El tiempo que demora la luz en atravesar la distancia física entre el origen y el destino.

La latencia introducida por un elemento de la red o latencia de inserción en periodos de congestión, como es el caso del gestor, depende principalmente del tamaño de la cola. Cabe decir que esta latencia será diferente para cada clase, siendo menor para las más prioritarias. Por lo tanto, este valor puede variar según la cola que se utilice y los parámetros con los que estén configuradas, sabiendo que cuanto menor es el tamaño de la cola menor es también la latencia.

Para el caso de colas SFQ, el tamaño de la cola no es configurable, teniendo un valor fijo de 127 paquetes. Este valor se considera relativamente elevado, lo que implica un aumento de la latencia, pero a su vez una disminución en la pérdida de paquetes.

Con la disciplina PFIFO este parámetro si que es configurable por el administrador, pudiendo variar la capacidad máxima de la cola. Por defecto, en *WebHTB* este parámetro es de 5 paquetes, lo que implica una menor latencia en comparación con la cola SFQ.

7.3.2.- Según protocolo

El protocolo del tráfico que se quiere gestionar también influye en el tipo de cola final a utilizar. La principal diferencia es que la cola PFIFO presenta algunas desventajas cuando se trata de protocolo UDP.

El protocolo de tráfico TCP cuenta con mecanismos de control de flujo extremo a extremo que permiten ajustar el tamaño de las ventanas de transmisión de manera que todos los flujos de una misma disciplina reciban una misma capacidad de canal. Por lo tanto, en presencia de mecanismos de control de flujo y de control de congestión, los algoritmos de justicia que impone SFQ para repartir el enlace de manera equitativa no aportan ninguna ventaja respecto a las colas PFIFO. Sin embargo, para protocolo de tráfico UDP, no existen estos mecanismos de control de flujo, con lo que las colas PFIFO atenderán a los distintos flujos de manera arbitraria.

En base a estos conocimientos se han realizado pruebas reales para determinar cual es la cola que mejor se adapta a las necesidades del sistema. En la figura 7.15 se puede ver la diferencia entre el uso de colas SFQ y PFIFO para diferentes flujos UDP que congestionan la red y que son desencolados por una misma cola final.

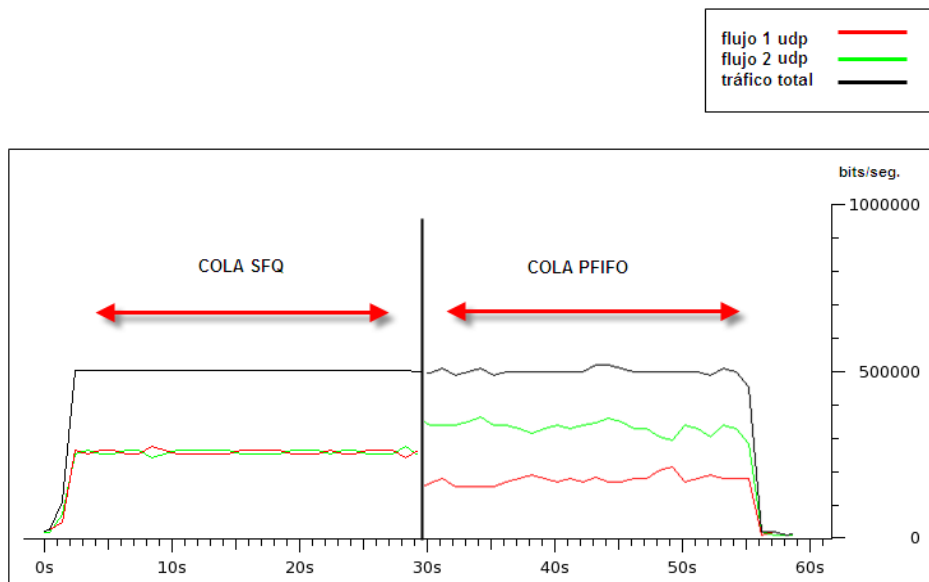


Figura 7.15.- Gráfico de comparación de colas con protocolo UDP

En este gráfico se demuestra el comportamiento descrito anteriormente. Cuando no existen mecanismo de control de flujo, la cola PFIFO desencola los paquetes de manera arbitraria, mientras que los algoritmos de justicia de la cola SFQ hace que cada flujo sea transmitido a la misma velocidad.

De la misma manera, se han realizado pruebas en entornos reales para flujos de tráfico TCP. En el gráfico de la figura 7.16 se muestran tres situaciones diferentes en función de la cola utilizada. En base a este gráfico se puede comprobar que cuando el enlace no presenta síntomas de congestión, ambas colas son capaces de repartir el ancho de banda para tráfico TCP de manera justa. Sin embargo, en periodos de congestión de red, la cola PFIFO presenta inestabilidades y fluctuaciones en su transmisión que hace que sea menos eficiente que la cola SFQ.

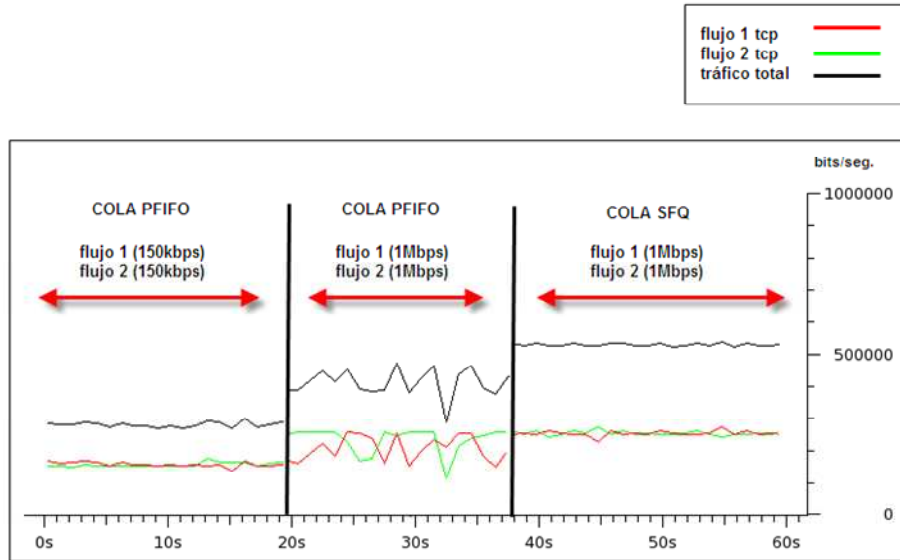


Figura 7.16.- Gráfico de comparación de colas con protocolo TCP

7.3.3.- Conclusiones

Según los resultados de este análisis, se puede concluir que la cola PFIFO presenta ventajas en términos de latencia, pero se muestra inestable en situaciones de congestión de red. Sin embargo, la cola SFQ, aunque presente unos resultados de latencia más desfavorables, se considera robusta a congestiones y hace una repartición de la capacidad de forma justa, tanto para tráfico de protocolo UDP como TCP.

Según este análisis, se utilizará siempre que sea posible la disciplina SFQ como cola final para desencolar los paquetes de una misma clase, ya que aunque presente desventajas en cuanto a latencia, desencola los paquetes de manera justa y eficiente para diferentes protocolos, además de una menor carga computacional.

7.4.- CONCLUSIONES DEL PROCESO DE PRUEBAS

A partir de los resultados prácticos, se concluye que el sistema de gestión es capaz de gestionar correctamente y de manera eficiente el tráfico de salida de sus interfaces para una configuración de red sencilla basada en tecnología Ethernet. Tanto la gestión del primer nivel del árbol de preferencias (servicios) como la del segundo y último nivel (clientes) presentan resultados favorables.

Sin embargo, cuando se configura una red más compleja y se pretende gestionar un enlace ADSL compartido, se comprueba que se deben tener en cuenta algunos requisitos para que la gestión coincida con la prevista.

El principal requisito para este tipo de configuración es que la capacidad de la interfaz de salida del gestor debe ser menor que el enlace más restrictivo de toda la red a gestionar. Para cumplirlo, es necesario controlar la capacidad máxima de salida de la interfaz de red limitando la clase *root*, con lo que se ha contribuido a la modificación aplicación *WebHTB*. Esta modificación ha dado paso a una nueva versión v2.3.

En lo que se refiere a disciplinas de colas finales, aunque la disciplina simple PFIFO presente ventajas en cuanto a latencia, siempre que sea posible se utilizará la cola final SFQ. Esta disciplina conlleva menor carga computacional y se presenta estable y justa en periodos de congestión de red para cualquier protocolo.

Debido a las diferentes etapas que debe pasar el paquete para ser clasificado en la clase correspondiente y más tarde desencholado, al crear elementos de control de tráfico en el sistema se da un aumento de la latencia. Se ha calculado de manera empírica el valor de este aumento respecto tener la disciplina de colas por defecto PFIFO_fast y tener configurado un árbol de preferencias HTB de dos niveles con SFQ como cola final. En este contexto y en situaciones sin congestión de red se ha estimado un incremento aproximado del 5%. Este aumento no es despreciable, pero es un precio a pagar asequible para conseguir una disminución de la latencia para tráfico prioritario y una gestión de ancho de banda en situaciones de congestión de red.

En conclusión, según todo el proceso de pruebas y el análisis realizado, se concluye que, si se respetan los requisitos comentados anteriormente, es posible conseguir una gestión de ancho de banda eficaz para el caso particular de la empresa.

8.- CONFIGURACIÓN PARA GENERAL LAB

8.1.- CONFIGURACIÓN DE LA RED DE LA EMPRESA

8.1.1.- Enlace a gestionar

Uno de los objetivos de los administradores de red de la empresa es que en un futuro todas las comunicaciones estén centralizadas a través de la MacroLan. Por este motivo, se decide implantar el sistema de gestión de ancho de banda para la red MacroLan, y no para la NetLan. Esto permite ayudar a que el proceso de centralización se lleve a cabo de manera sostenida y robusta debido a la incorporación de QoS.

8.1.2.- Servicios a gestionar

Con la gestión de ancho de banda se pretende diferenciar y priorizar el tráfico que generan las aplicaciones de negocio de General Lab. Estas aplicaciones son varias, pero se pueden destacar la aplicación de gestión de laboratorios WinLabNet y el sistema de gestión de recursos ERP.

Ambas aplicaciones están centralizadas, con lo que todo el tráfico que se genera viaja a través de la MacroLan hasta los servidores centrales. Las características de las distintas aplicaciones se encuentran resumidas en el cuadro de la figura 8.1, donde se listan los distintos servicios, puertos y direcciones IP que utilizan.

ENTRADA	SERVICIO	PROTOCOLO	SERVIDOR ORIGEN	PUERTO ORIGEN
1	Terminal Server	TCP	W2k3-APPSNet (128.0.0.25)	3389
2	SQL	TCP	W2k3-WinLabNetA (128.0.0.23) W2k3-WinLabNetB (128.0.0.24)	2034 3527 1433 1434
3	SQL	UDP	W2k3-WinLabNetA (128.0.0.23) W2k3-WinLabNetB (128.0.0.24)	1433 1434
4	ERP	TCP	AppSrvWeb1 (128.254.254.20) AppSrvWeb2	80

Figura 8.1.- Tabla de servicios prioritarios

En el lado opuesto se encuentra el resto de tráfico no prioritario de la empresa. Dentro de este grupo se engloban servicios como correo, acceso a Internet, Web interna, etc. Este tipo de tráfico también está centralizado y debe pasar a través de la MacroLan hacia los servidores centrales.

Según las necesidades de la empresa, el grupo de servicios prioritarios debe tener una mayor porción de ancho de banda asignado y una mayor prioridad. Según un análisis del tipo de tráfico que pasa a través de la MacroLan, la relación debe ser de 9 a 1. Es decir, que el tráfico de negocio debe tener asignado un 90% frente a un 10% del resto. Para ambas clases, se debe gestionar el ancho de banda excedente de manera que puedan disponer de todo el enlace en el caso que no esté siendo utilizado.

8.1.3.- Ubicación física del servidor de gestión

La ubicación física del servidor dentro de la red es un aspecto importante si se quiere gestionar el ancho de banda de manera óptima. Como se verá más adelante, el sistema debe realizar gestión en origen para poder controlar a través de la interfaz eth1 el tráfico en sentido descendente.

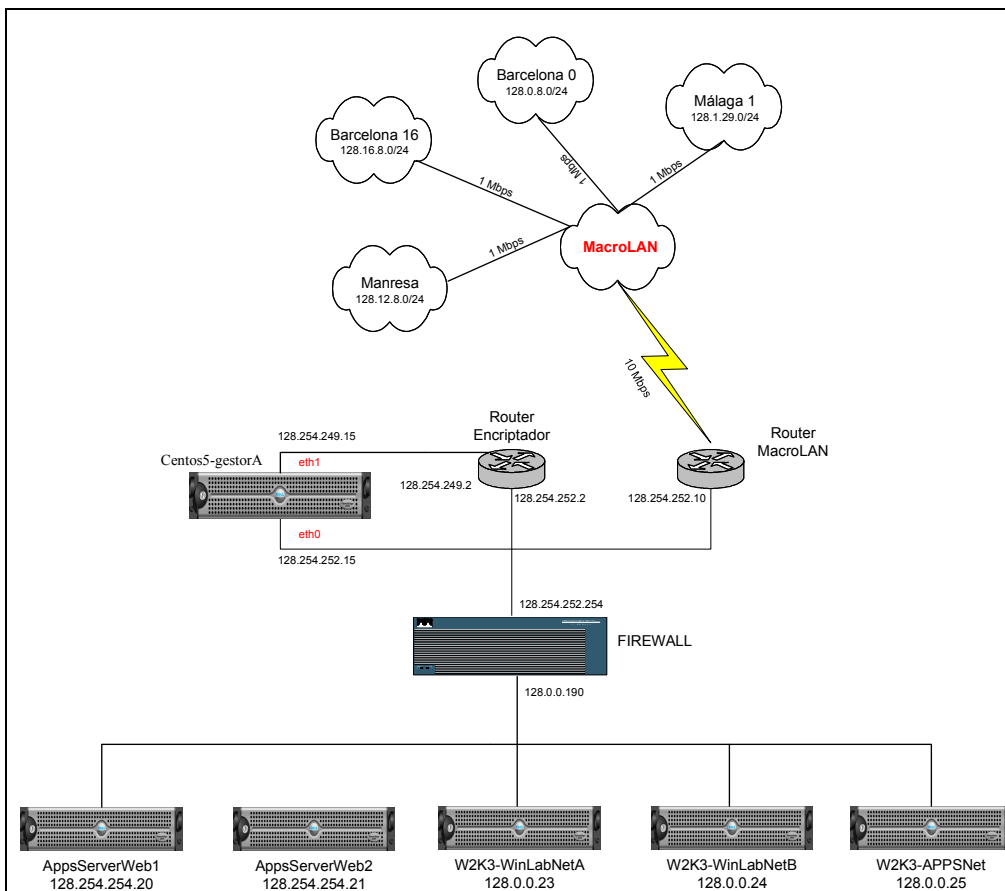


Figura 8.2.- Esquema de ubicación del gestor en producción⁶

⁶ En la figura tan solo se muestran los elementos de red que forman parte activa del tráfico a gestionar.

Teniendo esto en cuenta, la ubicación que permite gestionar el tráfico de toda la MacroLan es entre el router encriptador y el *firewall* (Figura 8.2). Por este punto de la red pasa todo el tráfico propio de la MacroLan si se modifica la tabla de rutas del *firewall* central.

8.1.4.- Aclaraciones sobre la red a gestionar

Debido a que la red MacroLan no garantiza la LOPD (Ley Orgánica de Protección de Datos), es necesario instalar un router adicional en cada punto de la red para encriptar el tráfico (router encriptador). Esto hace que existan dos routers tanto en los centros como en la sede central: el router MacroLan y el router encriptador.

El router MacroLan es el router de acceso que se encarga de conectar la sede central con la red MacroLan. Este router tan solo enruta y no hace NAT de los paquetes que procesa, ya que todos los centros saben de la existencia del rango IP del resto de centros. Por otro lado, el router encriptador tiene las funciones de encriptar y desencriptar el tráfico de esta red. Una vez se ha desencriptado el tráfico, se envía al *firewall*, que es quien hace las funciones de enrutamiento hacia su destino final, y viceversa.

El servidor de gestión tiene capacidad para gestionar el tráfico por todas las interfaces de red instaladas. Sin embargo, debido a la estructura de red de la empresa y a las características de las aplicaciones de negocio, tan solo se requiere gestionar el tráfico en una de ellas, concretamente en la que se corresponde con el sentido descendente (servidores → centros).

En un principio, el sistema de gestión tan solo se va a utilizar para controlar el tráfico generado por los centros con conexión a la MacroLan. Sin embargo, el router encriptador es compartido entre la red MacroLan y la NetLan. Si se quisiera gestionar el tráfico en sentido ascendente, el router encriptador debería diferenciar el tráfico en función de la dirección de origen, enrutando hacia centos5-gestorA los centros a gestionar y hacia el *firewall* el resto de tráfico. Pero el modelo de router utilizado no permite hacer este tipo de diferenciación, y tan solo es posible enrutar por destino, no por origen. En el caso del tráfico descendente es el *firewall* quien debe hacer las funciones de enrutar hacia el gestor los paquetes con destino los centros de la MacroLan y hacia el router el resto de paquetes.

Ya que no todos los equipos de la red permiten diferenciar por direcciones IP de origen, el sistema de gestión tan solo contempla la gestión del ancho de banda en sentido descendente.

Sin embargo, este hecho no es del todo perjudicial. La principal función de las aplicaciones de negocio de la empresa es la de consultar información y descargar datos de los servidores centrales, no la de cargar datos. Esto hace que el enlace conflictivo y crítico sea el enlace descendente, ya que en ningún caso podrá existir una situación de congestión del enlace ascendente teniendo una capacidad de 1Mbps, como es el caso de las conexiones MacroLan.

8.1.5.- Modificaciones en la red debido al sistema de gestión

Dentro de este contexto, es necesario configurar y modificar algunos de los diferentes elementos de red que forman parte del control del tráfico, como son el servidor de gestión y el *firewall* de la sede central.

- **Configuración de red del gestor:** Este servidor debe tener configurada una tabla de rutas como la de la figura 8.3 para poder enrutar el tráfico hacia los distintos centros.

Kernel IP routing table						
Destination	Gateway	Genmask	Flags	Metric	Ref	Use Iface
128.16.8.0	128.254.249.2	255.255.255.0	UG	0	0	0 eth0
128.12.8.0	128.254.249.2	255.255.255.0	UG	0	0	0 eth0
128.0.8.0	128.254.249.2	255.255.255.0	UG	0	0	0 eth0
128.1.29.0	128.254.249.2	255.255.255.0	UG	0	0	0 eth0
128.0.0.0	128.254.252.254	255.255.255.0	UG	0	0	0 eth1
128.254.254.0	128.254.252.254	255.255.255.0	UG	0	0	0 eth1
128.254.249.0	*	255.255.255.0	U	0	0	0 eth0
128.254.252.0	*	255.255.255.0	U	0	0	0 eth1

Figura 8.3.- Tabla de rutas en producción

- **Modificaciones en el *firewall*:** Este *firewall* debe contener reglas para enviar hacia el gestor (128.254.252.15) en lugar de hacia al router encriptador (128.254.252.2) los paquetes con destino los centros de la MacroLan. El modelo de este *firewall* es Cisco PIX 515E, y los comandos utilizados para enrutar el tráfico se muestran en la figura 8.4.

```
no route wan 128.16.8.0 255.255.255.0 128.254.252.2 1
route wan 128.16.8.0 255.255.255.0 128.254.252.15 1

no route wan 128.12.8.0 255.255.255.0 128.254.252.2 1
route wan 128.12.8.0 255.255.255.0 128.254.252.15 1

no route wan 128.0.8.0 255.255.255.0 128.254.252.2 1
route wan 128.0.8.0 255.255.255.0 128.254.252.15 1

no route wan 128.1.29.0 255.255.255.0 128.254.252.2 1
route wan 128.1.29.0 255.255.255.0 128.254.252.15 1
```

Figura 8.4.- Comandos para enrutar el tráfico en el *firewall*

8.2.- CREACIÓN DE LOS ELEMENTOS DE CONTROL CON *WEBHTB*

Para gestionar el ancho de banda disponible según las necesidades de la empresa se utiliza la aplicación *WebHTB*. A partir de esta herramienta, se crean las clases y los filtros necesarios para cada interfaz según el manual de usuario que se encuentra en el anexo E.- *Manual de usuario de WebHTB*.

8.2.1.- Creación de las clases

Para el caso del sistema en producción, se realiza gestión en origen de un enlace de 10Mbps, que tiene que canalizar el tráfico de todos los centros MacroLan con una capacidad máxima de 1Mbps cada uno. Por este motivo, se limita la clase *root* a una capacidad algo inferior de 10Mbps siguiendo la ecuación 7.1. Debajo de esta clase *root*, se crea una clase madre para cada uno de los centros de la MacroLan. Como la conexión de todos los centros tiene las mismas características, se configuran todas las clases madre siguiendo el mismo criterio, con un *rate* y un *ceil* igual a su ancho de banda (1Mbps). En un principio no se considera ningún tipo de prioridad entre centros, por lo que la prioridad de cada clase debe ser la misma. En lo referente a las colas finales, se utilizan colas SFQ según el análisis realizado en el capítulo anterior.

De la misma manera, se configuran dos clases hijas o clientes en cada clase madre. Para todas ellas, la primera de estas clases es la que gestiona el tráfico prioritario de negocio de la empresa (clase *NEGOCIO_centro*⁷). A esta clase se le asigna un *rate* aproximadamente del 90% del ancho de banda disponible (920kbps) y un *ceil* de la capacidad total (1Mbps).

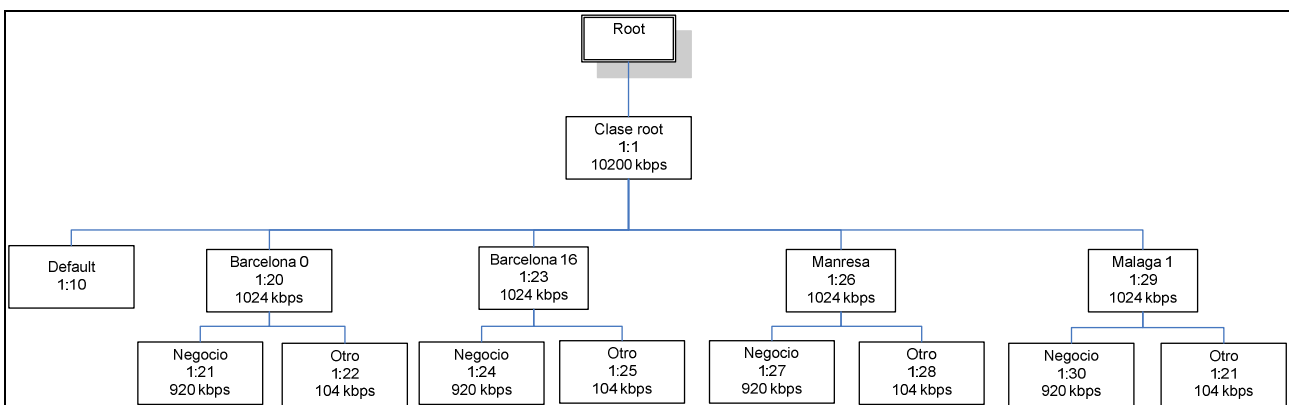


Figura 8.5.- Árbol de preferencias en producción

⁷ *Centro* equivale a la codificación utilizada para representar cada uno de los centros de la empresa, que consiste en una abreviatura de la ciudad y el número de direccionamiento IP asignado.

La otra clase hija se corresponde con el resto de tráfico no prioritario (clase OTRO_ *centro*). Esta debe tener asignada aproximadamente el 10% restante de la capacidad del enlace (104kbps) y un *ceil* de la capacidad total (1Mbps). Es importante que esta clase sea algo menos prioritaria que la anterior.

Toda esta configuración se puede ver en forma de árbol de preferencias en la figura 8.5. Además, en la figura 8.6 se muestra la ventana principal de la aplicación *WebHTB* donde se han creado los distintos elementos de control de tráfico HTB.

CLIENT	BANDWIDTH	LIMIT	BURST	PRIORITY	UPLOAD	MARK	SRC IPS	DST IPS	EDITAR	ELIMINAR
ACTUALMENTE TRABAJANDO EN LA INTERFAZ: eth1										
CLASS BARCELONA16, BANDWIDTH 1024, BURST AUTO, LIMIT 1024, PRIORITY 0, QUE SFQ [-]										
1.WinlabNet_bcn16	920	1024	0	0			128.0.0.23 128.0.0.24 128.0.0.25	128.16.8.0 128.16.8.0 128.16.8.0		
2.Otros_bcn16	104	1024	0	3				128.16.8.0		
CLASS MANRESA, BANDWIDTH 1024, BURST AUTO, LIMIT 1024, PRIORITY 0, QUE SFQ [-]										
1.WinlabNet_man	920	1024	0	0			128.0.0.23 128.0.0.24 128.0.0.25	128.12.8.0 128.12.8.0 128.12.8.0		
2.Otros_man	104	1024	0	3				128.12.8.0		
CLASS BARCELONA0, BANDWIDTH 1024, BURST AUTO, LIMIT 1024, PRIORITY 0, QUE SFQ [-]										
1.ERP_bcn0	920	1024	0	0			128.254.254.20 128.254.254.21	128.0.8.0 128.0.8.0		
2.Otros_bcn0	104	1024	0	3				128.0.8.0		
CLASS MALAGA1, BANDWIDTH 1024, BURST AUTO, LIMIT 1024, PRIORITY 0, QUE SFQ [-]										
1.WinLabNet_mal1	920	1024	0	0			128.0.0.23 128.0.0.24 128.0.0.25	128.1.29.0 128.1.29.0 128.1.29.0		
2.Otros_mal1	104	1024	0	3				128.1.29.0		

Figura 8.6.- Ventana de creación de elementos en *WebHTB*⁸

⁸ Los nombres de las clases y los clientes pueden variar de los expuestos en el proyecto debido a que se han utilizado nombres internos.

8.2.2.- Configuración de los filtros

A la hora de crear y configurar los filtros, es importante el orden en que se comprueban las reglas que lo conforman (Figura 8.7). Los paquetes deben entrar al gestor y comprobar si cumplen algunas de las condiciones registradas en los filtros de la clase `NEGOCIO_centro`.

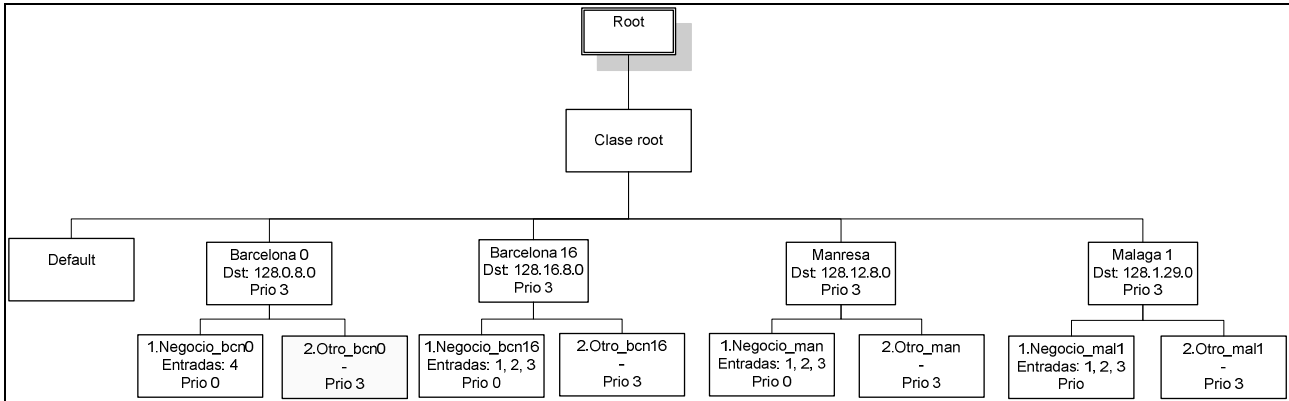


Figura 8.7.- Configuración de los filtros en producción⁹

Si se cumple una de estas, automáticamente el paquete es encolado hacia la clase correspondiente. Si no cumplen ninguna, deben comprobar si cumplen algunas de las condiciones de la clase `OTROS_centro`. Si el paquete pasa por todos los filtros sin coincidir con ninguna de las reglas, es encolado a la clase por defecto. Como se da el caso de que la clase `NEGOCIO_centros` y la clase `OTROS_centros` comparten reglas de filtrado (en concreto, ambas coinciden en la dirección IP de destino) es importante que se ordenen los filtros para evitar clasificaciones de paquetes no deseadas.

El orden de los filtros de una misma clase se hace utilizando el parámetro *prio*, que define la prioridad de cada filtro. Pero como la aplicación *WebHTB* define por defecto la misma prioridad para todos ellos, este orden se debe hacer alfanuméricamente, de manera que las reglas que se encuentren antes se comprueban primero.

8.2.3.- Comprobación de los elementos de control de tráfico creados

Una vez creados los distintos elementos de control de tráfico a partir de la aplicación *WebHTB*, es conveniente comprobar sus parámetros y su configuración en el *kernel*. Esto se consigue haciendo uso del binario *tc* con la opción *show*. Para el caso en concreto del sistema de gestión en producción, se tienen las clases y las disciplinas de colas mostradas en las figuras 8.8 y 8.9. Los valores de estos elementos de control de tráfico siguen las reglas descritas en el punto 6.2.3.- *Parámetros HTB*.

⁹ Entradas correspondientes a la tabla 8.1

```

#Clase root
class htb 1:1 root rate 10200Kbit ceil 10200Kbit burst 3060Kb cburst 2872b

#Clase default
class htb 1:10 parent 1:1 leaf 10: prio 0 rate 64000bit ceil 8000bit burst 1607b
cburst 1600b

#Clase BARCELONA16 y sus clientes
class htb 1:20 parent 1:1 rate 1024Kbit ceil 1024Kbit burst 307Kb cburst 1727b
class htb 1:21 parent 1:20 leaf 21: prio 0 rate 920000bit ceil 1024Kbit burst
307Kb cburst 1727b
class htb 1:22 parent 1:20 leaf 22: prio 3 rate 104000bit ceil 1024Kbit burst
307Kb cburst 1727b

#Clase MANRESA y sus clientes
class htb 1:23 parent 1:1 rate 1024Kbit ceil 1024Kbit burst 307Kb cburst 1727b
class htb 1:24 parent 1:23 leaf 24: prio 0 rate 920000bit ceil 1024Kbit burst
307Kb cburst 1727b
class htb 1:25 parent 1:23 leaf 25: prio 3 rate 104000bit ceil 1024Kbit burst
307Kb cburst 1727b

#Clase BARCELONA0 y sus clientes
class htb 1:26 parent 1:1 rate 1024Kbit ceil 1024Kbit burst 307Kb cburst 1727b
class htb 1:27 parent 1:26 leaf 27: prio 0 rate 920000bit ceil 1024Kbit burst
307Kb cburst 1727b
class htb 1:28 parent 1:26 leaf 28: prio 3 rate 104000bit ceil 1024Kbit burst
307Kb cburst 1727b

#Clase MALAGA1 y sus clientes
class htb 1:29 parent 1:1 rate 1024Kbit ceil 1024Kbit burst 307Kb cburst 1727b
class htb 1:2a parent 1:29 leaf 2a: prio 0 rate 920000bit ceil 1024Kbit burst
307Kb cburst 1727b
class htb 1:2b parent 1:29 leaf 2b: prio 3 rate 104000bit ceil 1024Kbit burst
307Kb cburst 1727b

```

Figura 8.8.- Parámetros de las clases en producción

```

#Disciplina de la interfaz de red
qdisc ingress ffff: parent ffff:fff1 -----

#Disciplina de la clase root
qdisc htb 1: root r2q 10 default 10 direct_packets_stat 0

#Disciplina de la clase default
qdisc pfifo 10: parent 1:10 limit 5p

#Disciplinas de las clases hijas de BARCELONA16
qdisc sfq 21: parent 1:21 limit 127p quantum 1514b
qdisc sfq 22: parent 1:22 limit 127p quantum 1514b

#Disciplinas de las clases hijas de MANRESA
qdisc sfq 24: parent 1:24 limit 127p quantum 1514b
qdisc sfq 25: parent 1:25 limit 127p quantum 1514b

#Disciplinas de las clases hijas de BARCELONA0
qdisc sfq 27: parent 1:27 limit 127p quantum 1514b
qdisc sfq 28: parent 1:28 limit 127p quantum 1514b

#Disciplinas de las clases hijas de MALAGA1
qdisc sfq 2a: parent 1:2a limit 127p quantum 1514b
qdisc sfq 2b: parent 1:2b limit 127p quantum 1514b

```

Figura 8.9.- Parámetros de las disciplinas de colas finales en producción

8.3.- RESULTADOS Y ANÁLISIS DEL SISTEMA EN PRODUCCIÓN

Una vez el sistema de gestión está en producción, es más complejo testear y comprobar su comportamiento, debido a que se puede alterar el tráfico de negocio y la actividad de la empresa. Sin embargo, se entiende que es indispensable analizar el sistema para poder identificar cualquier problema o error.

En primer lugar, se analiza de manera aproximada el volumen de tráfico que se encola en cada clase. De este análisis se extrae que el centro que requiere mayor capacidad del enlace es Barcelona 16, con una ocupación media aproximada en periodos de producción de un 30% de su ancho de banda.

Por estos motivos, se realiza una prueba entre clientes para la clase BARCELONA 16. En esta prueba se intentan comprobar dos puntos. Por un lado, que se respeten las prioridades de las diferentes clases clientes. Esto quiere decir que los flujos de tráfico de negocio no se deben mostrar alterados en caso de que un tráfico no prioritario congestione la red. Por otro lado, se debe comprobar que el sistema gestione correctamente el tráfico excedente según la configuración previa (Figura 8.10).

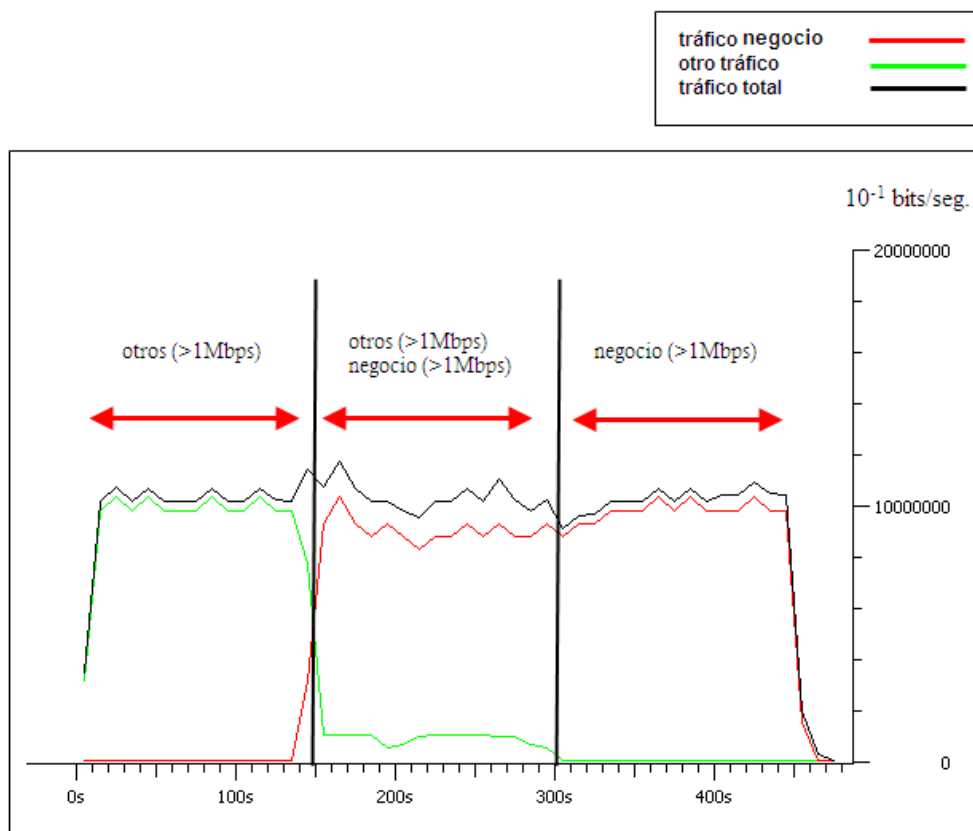
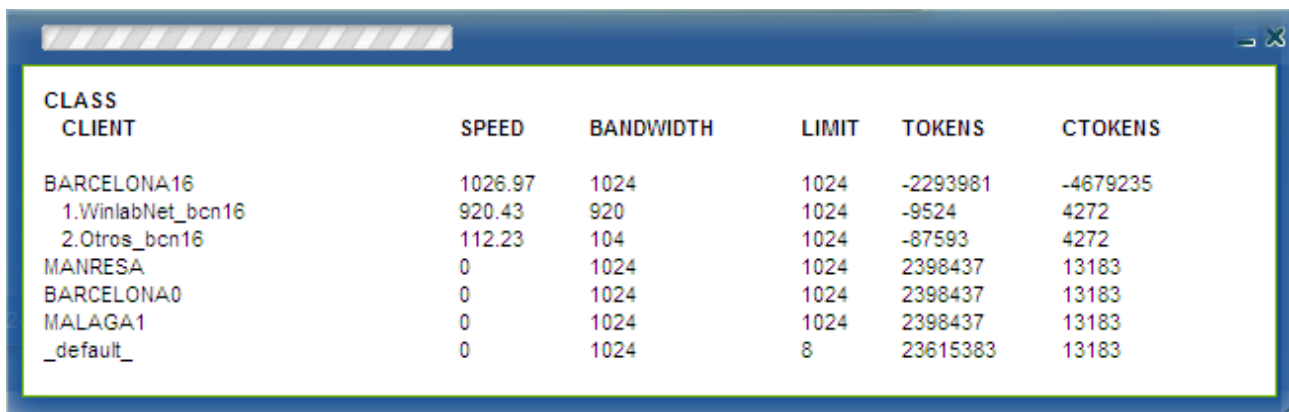


Figura 8.10.- Gráfico de resultados en producción

Durante toda la prueba, se ha intentado someter a la red a más tráfico del que puede soportar, con el fin de comprobar el comportamiento del sistema en situaciones de congestión. Hay que destacar que durante el transcurso de la prueba, además del tráfico generado para testear el sistema también se tiene tráfico propio del trabajo de la empresa. Aunque se ha intentado realizarla en periodos de baja actividad, los valores de capacidad y la precisión pueden variar significativamente.

En el primer tramo, se observa como el tráfico no prioritario correspondiente a la clase OTRO_bcn16 ocupa la mayor parte del enlace. El resto ancho de banda hasta llegar al límite de 1Mbps lo ocupa la aplicación de negocio WinLabNet, con lo que se comprueba que aunque se congestione la red con tráfico no prioritario, el sistema respeta las prioridades establecidas.

En el segundo tramo, en cambio, se tiene una superposición de tráfico no prioritario y de tráfico de negocio. En este caso, se puede observar como el sistema gestiona el ancho de banda disponible de manera que reparte un 90% de esta capacidad al tráfico de negocio y un 10% al resto, tal y como se ha definido previamente en el árbol de preferencias. Este reparto se puede ver más claramente en la ventana que muestra el tráfico en tiempo real (Figura 8.11).



CLASS	SPEED	BANDWIDTH	LIMIT	TOKENS	CTOKENS
CLIENT					
BARCELONA16	1026.97	1024	1024	-2293981	-4679235
1.WinlabNet_bcn16	920.43	920	1024	-9524	4272
2.Otros_bcn16	112.23	104	1024	-87593	4272
MANRESA	0	1024	1024	2398437	13183
BARCELONA0	0	1024	1024	2398437	13183
MALAGA1	0	1024	1024	2398437	13183
default	0	1024	8	23615383	13183

Figura 8.11.- Ventana de tráfico en tiempo real para la prueba en producción.

En base a estos resultados, se puede concluir que el sistema gestiona correctamente el ancho de banda disponible en producción, tal y como se había comprobado el proceso de pruebas y testing. Los resultados obtenidos demuestran que la precisión de la gestión es elevada, aún teniendo en cuenta que la prueba se ha realizado en condiciones reales en periodos de actividad.

9.- ANÁLISIS DEL PROYECTO

9.1.- PLANIFICACIÓN DEL PROYECTO

La planificación del proyecto se ha llevado a cabo en base al diagrama mostrado al inicio del documento en la figura 1.3. Aunque no ha sido posible cumplir estrictamente los plazos previstos, se ha conseguido finalizar el proyecto aproximadamente sobre las fechas indicadas.

Este proyecto se ha realizado mayoritariamente en las instalaciones de General Lab entre los meses de agosto de 2008 y febrero de 2009, pero también se ha realizado cierto trabajo fuera de la empresa.

En las instalaciones de General Lab: El número días dedicados al proyecto dentro de la empresa ha sido de 140 días. Dentro de estos días no se ha tenido dedicación total al proyecto, sino que también se ha colaborado en tareas puntuales propias de la empresa, como resolución de incidencias, soporte al proceso de centralización y cursos de formación. El número de horas aproximado que se han dedicado en exclusividad al proyecto es de unas 720 horas. Estas horas se han dedicado principalmente a llevar a cabo tareas que requerían del material de la empresa, como sería todo el proceso de instalación, configuración, pruebas y análisis. También se han dedicado algunas de estas horas a búsqueda de información y documentación y a dar formación sobre el sistema de gestión al personal de la empresa.

Fuera de las instalaciones de General Lab: Para finalizar el proyecto han sido necesarias unas 100 horas aproximadamente, las cuales se han dedicado básicamente a búsqueda bibliográfica, búsqueda de información para la resolución de incidencias y labores de documentación.

En total, el número de horas dedicadas a la elaboración y la documentación del proyecto es de 820 horas, repartidas entre los meses de agosto de 2008 y febrero de 2009.

9.2.- COSTE DEL PROYECTO

El coste económico del proyecto equivale principalmente al coste del servidor utilizado como gestor de ancho de banda y las horas dedicadas por convenio dentro de General Lab. El resto de material utilizado para los bancos de pruebas no ha supuesto coste adicional, ya que se trata de material reciclado o en stock, y las aplicaciones utilizadas son con licencia GNU GPL.

TIPO	CONCEPTO	COSTE
COSTE MATERIAL	SERVIDOR	2.700 €
COSTE PERSONAL	CONVENIO	5.040 €
TOTAL		7.740 €

Figura 9.1.- Tabla de coste del proyecto

En la tabla de la figura 9.1 se muestra un resumen del coste del proyecto teniendo en cuenta tan solo el coste del servidor y las horas dedicadas dentro del convenio. Dentro del coste en concepto de convenio, solo se han considerado las horas dedicadas en exclusividad al proyecto dentro de las instalaciones de General Lab.

Como se puede observar en el resultado total, el coste del proyecto según se ha llevado a cabo es significativamente superior al de las herramientas en el mercado analizadas en el capítulo 4. Sin embargo, al precio de estas herramientas se le debe añadir el del tiempo dedicado por el proveedor para poner en funcionamiento el sistema y realizar la documentación necesaria, lo que hace que el precio final sea variable.

La diferencia entre los dos presupuestos radica principalmente en el coste del personal, que equivale a una dedicación prácticamente exclusiva al desarrollo del proyecto de una persona durante un periodo de 720 horas. Aunque el coste total indicado es a priori superior al de la implementación de una herramienta ya desarrollada, el hecho de dedicar este número de horas al proyecto ha permitido implementar y configurar un sistema personalizado que cumple de manera exhaustiva con todos los requisitos iniciales de la empresa. Además, ha sido posible realizar una documentación detallada, dar formación al personal de la empresa y realizar un análisis de posibles alternativas de futuro, que permiten mejorar y ampliar el sistema.

9.3.- EVALUACIÓN DE OBJETIVOS

Objetivo 1.- Obtención de conocimientos teóricos sobre QoS y gestión de ancho de banda en entorno GNU/Linux.

Evaluación: A partir de la búsqueda bibliográfica, se ha conseguido establecer una fuerte base teórica sobre todo el proceso y las distintas partes que componen la gestión de ancho de banda en entorno GNU/Linux. Dentro de estos conocimientos entra el estudio de los modelos QoS, las distintas disciplinas de colas y su funcionamiento, y las herramientas de configuración de red que utiliza GNU/Linux.

Objetivo 2.- Análisis de distintas herramientas disponibles para proporcionar QoS.

Evaluación: En primer lugar se han definido unos requisitos que debe cumplir la herramienta que finalmente sea utilizada para gestionar el ancho de banda. En base a estos requisitos, se han analizado tanto herramientas que se encuentran actualmente en el mercado como de las denominadas libres. Finalmente, después de analizar las características de cada una de ellas, se ha elegido la herramienta *WebHTB*, por ser la que mejor se adapta a las necesidades del proyecto.

Objetivo 3.- Instalación y configuración del sistema.

Evaluación: Este objetivo ha sido el más complejo desde el punto de vista técnico y el que más tiempo ha llevado alcanzarlo. Durante el periodo de instalación, tanto de la máquina como de la aplicación, han surgido algunas incidencias que se han ido solucionando. Todas estas incidencias, así como su resolución y el proceso de instalación, han sido convenientemente documentadas con el fin de ayudar al personal de la empresa a afrontar las posibles mejoras y ampliaciones del proyecto.

Objetivo 4.- Facilitar a la empresa de un manual de instalación y de la documentación necesaria sobre el uso y el mantenimiento del sistema.

Evaluación: En base a todos los conocimientos adquiridos durante el desarrollo de los distintos objetivos, se ha documentado todo el proceso de instalación, configuración y mantenimiento de la herramienta. Además, se han dado cursos formativos para el personal de General Lab con la finalidad de exponer el proyecto y resolver todo tipo de dudas que han podido surgir. También se han propuesto algunas posibles ampliaciones y mejoras que se pueden llevar a cabo para optimizar el servicio de gestión de ancho de banda.

Objetivo 5.- Análisis y ejecución de un procedimiento para la puesta en producción del servicio.

Evaluación: Junto con el departamento de sistemas se ha analizado la mejor configuración del gestor para cumplir con los requisitos de la empresa. Seguidamente se han configurado en el servidor y se han modificado los distintos elementos de la red por tal de poder gestionar el tráfico. Con el gestor en producción, se ha analizado exhaustivamente su funcionamiento. Este análisis ha resultado ser positivo, y todas las pruebas realizadas han sido satisfactorias.

10.- CONCLUSIONES Y LINEAS DE FUTURO

10.1.-CONCLUSIONES

Un sistema de QoS extremo a extremo con gestión de ancho de banda es útil para situaciones de congestión de red o de uso de gran parte del ancho de banda disponible. En la situación actual de la empresa, no se dan estas situaciones debido a las altas prestaciones de las redes MacroLan y a los pocos centros todavía centralizados. Sin embargo, en vistas a la posible centralización de nuevos centros, se puede dar el caso que el servicio de gestión sea indispensable para conseguir un crecimiento sostenido de la empresa.

En el lado opuesto, se tiene los inconvenientes de la aparición de de *starvation* en situaciones de congestión que se prolongan en el tiempo, además de un aumento significativo de la latencia. Tras analizar las ventajas y los inconvenientes del sistema, las magnitudes de estos fenómenos se consideran despreciables frente a la mejora del uso de la red con el sistema de gestión en situaciones críticas de congestión.

A lo largo del proyecto se han tomado decisiones en cuanto a las disciplinas de colas finales, los anchos de banda asignado a cada tipo de tráfico y los valores de diferentes parámetros en base algunas necesidades del proyecto. Sin embargo, otras configuraciones y otros valores se podrían considerar también válidos.

Además de las herramientas habituales que se comercializan en el mercado para configurar sistemas de QoS con gestión de ancho de banda, también existen paquetes del núcleo de sistemas GNU/Linux que presentan buenos resultados. Para dar lugar al sistema de gestión en este proyecto se ha utilizado la aplicación *WebHTB* para crear y configurar los parámetros de la disciplina HTB sobre el binario *tc* del paquete *iproute*. La disciplina de colas HTB se considera relativamente nueva, por lo que todavía no existe una documentación amplia y desarrollada. En lo que se refiere a la aplicación *WebHTB*, todavía está en vías de desarrollo y presenta puntos susceptibles de mejora.

Debido a que este es un proyecto de futuro para la empresa, una parte importante ha sido la de documentar detalladamente el funcionamiento de todas las partes del sistema y la de formar al personal del departamento de sistemas de General Lab, además de la de proponer posibles líneas de futuro.

10.2.- LÍNEAS DE FUTURO

10.2.1.- Migración del sistema operativo

Para los sistemas operativos Linux es importante mantenerse al día las actualizaciones que van apareciendo. Para CentOS en concreto, si el servidor se encuentra conectado a Internet, es posible configurarlo para que avise cada vez que hay actualizaciones disponibles. Una vez se notifica las novedades, es posible actualizar el sistema ya sea a través del escritorio o a través de la consola ejecutando el comando *yum update* con permisos de súper-usuario.

Antes de ejecutar estas actualizaciones, es importante asegurarse e informarse de los nuevos paquetes que se instalarán, y si afectarán a los paquetes actualmente en uso en el sistema. También es aconsejable tener una copia de seguridad del sistema. Una vez actualizado el sistema, se recomienda reiniciar el equipo para que todas las actualizaciones tengan efecto inmediato.

10.2.2.- Configuración del sistema en alta disponibilidad

Por política de empresa, todo dispositivo de red o servidor debe ser redundante. Esto quiere decir que deben existir físicamente dos dispositivos idénticos, uno en producción y el otro de *backup*.

Esta regla se aplica en todos los servidores de la empresa, así como routers, *firewalls*, etc. Normalmente, el cambio de un dispositivo a otro en caso de caída se hace manualmente. Sin embargo, existen puntos críticos donde es conveniente que este cambio se haga de manera automática y transparente para el usuario.

Este proyecto tan solo considera la instalación y puesta en producción de un gestor dentro de la empresa., pero ya a que a través de este gestor pasa gran parte del tráfico de negocio, sería recomendable que se configurara en alta disponibilidad. Para hacerlo, en primer lugar serían necesarios dos servidores iguales pero con direcciones IP distintas dentro del mismo rango. Por otro lado, habría que configurar el *firewall* para que en caso de caída de uno de ellos el tráfico pasara a través del redundante de manera automática.

10.2.3.- Migración a nuevas versiones de *WebHTB*

Durante el transcurso del proyecto han ido surgiendo actualizaciones de la aplicación *WebHTB*. La mayor parte del proyecto se ha llevado a cabo con la versión *WebHTB* v2.3, y es la que ha quedado instalada al finalizar el proyecto. Sin embargo, hay constancia de la existencia de la versión v2.7 con mejoras en seguridad y autenticación, pero no se ha visto conveniente instalarla.

Para conocer las actualizaciones de *WebHTB* y descargarlas hay que visitar el enlace oficial. Como se ha comentado anteriormente, es importante que antes de actualizar se lean detenidamente las *releases notes* para ver las modificaciones respecto la versión anterior, y tener una copia de seguridad.

10.2.4.- Análisis sobre la incorporación al sistema de NAPI

NAPI (New API) es una modificación en el mecanismo de interrupciones utilizado por los núcleos de los sistemas Linux, con la intención de mejorar el procesado de paquetes a velocidades elevadas.

Cuando el sistema está sometido a situaciones de tráfico elevado, se pueden llegar a producir cientos de interrupciones por segundo. Por defecto, se genera una interrupción cada vez que se recibe un paquete. Con esta modificación, el mecanismo mezcla estas interrupciones con encuesta (*polling*), con lo que se reduce el número de interrupciones y por consiguiente la carga en el procesador, enviando el mismo número de paquetes por segundo.

La posibilidad de utilizar NAPI ha sido introducida a partir de las versiones de kernel 2.6, y es totalmente opcional. En el sistema utilizado para crear el gestor de ancho de banda se tiene esta modificación deshabilitada. Se recomienda analizar la posibilidad de habilitarla, teniendo en cuenta las ventajas (menor carga computacional) y desventajas (aumento de la latencia), y comprobar a partir de que capacidad del enlace es recomendable su uso.

10.2.5.- Integración en el sistema de copias de seguridad de la empresa

La empresa tiene un sistema centralizado de copias de seguridad. A partir de la aplicación de agente SymantecVeritas instalada en uno de los servidores centrales es posible hacer copia de seguridad periódicamente de cualquier servidor en la red de la empresa que tenga instalado el cliente Symantec Veritas. Por estos motivos, se recomienda instalar esta aplicación en el gestor para tener una copia de seguridad de forma periódica.

ANEXOS

A.- Instalación y configuración del sistema operativo

A.1.- Instalación de CentOS

La versión más reciente es CentOS 5.2. Para conseguirla se puede descargar desde su página oficial en la referencia [33]. Desde la pestaña **DOWNLOADS** se puede elegir la versión de la distribución, el tipo y desde que enlace se quiere descargar. En este caso ha descargado una imagen DVD de la versión 5.2 para i386.

La instalación se ejecuta en entorno gráfico por ser más ágil y dinámico, gracias al asistente Anaconda. Al iniciar la instalación en este entorno se observa que el sistema se queda bloqueado en la instrucción:

```
ACPI: Interpreter enabled
ACPI: Using IOACPI for interrupt routing
ACPI: PCI Root Bridge [PCI0](0000:00)
```

Este error es debido a que algún hardware o la propia BIOS no soportan alguna de las funciones de ACPI. La función de ACPI (Advanced Configuration and Power Interface) es permitir al sistema operativo configurar y controlar cada componente de hardware por separado. Entre otras cosas, proporciona diversos datos sobre la batería, interfaz de red, temperatura y ventilador e informa de acontecimientos en el sistema como el estado de la batería.

Para solucionar este problema, se puede optar por deshabilitar las funciones de ACPI para todos los dispositivos (*acpi=off*) o deshabilitarlo tan solo para los dispositivos PCI (*pci=noacpi*). Además, en este punto se puede escoger el idioma del proceso de instalación.¹⁰

```
linux acpi=off lang=es + (ENTER)
```

Como es la primera vez que se utiliza el disco donde está almacenada la distribución, es aconsejable verificar su integridad. Esta acción puede durar varios minutos, según de la velocidad del sistema.

Una vez verificado correctamente, para crear una disposición de las particiones que permita un mayor control del sistema, es aconsejable evitar la disposición por defecto y crear una personalizada. Se dota a la partición */var* de gran parte de la capacidad total del sistema, ya que se ejecutarán distintas herramientas en entorno Web (Figura A.1).

¹⁰ En este punto de la instalación todavía no se ha configurado ningún idioma para el teclado, por lo que es necesario incluir los distintos símbolos según su código ASCII.

Punto de montaje	Sistema de archivos	Tamaño
/boot	Ext3	125MB
/	Ext3	80GB
/tmp	Ext3	10GB
/usr	Ext3	8GB
-	Swap	4GB
/var	Ext3	50.5GB

Figura A.1.- Distribución de las particiones del sistema

Para definir el nombre del host se sigue la política implantada por la empresa de mencionar en este nombre el sistema operativo utilizado y una palabra que identifique a la máquina, ya sea por su función o por su ubicación.

Algunos ejemplos reales son:

- **W2K-STA**: Servidor con Windows 2000 ubicado en el hospital Santa Teresa de a Coruña.
- **W2K3-WEB**: Servidor con Windows 2003 que soporta la página Web de la empresa.

Siguiendo estos criterios, el nombre elegido es centos5-gestorA.general-lab.com

- Localhost: **centos5-gestorA**
- Localdomain: **general-lab.com**

Para finalizar el proceso de instalación, se configura la contraseña del administrador *root*, el reloj del sistema y se instala el escritorio GNOME para hacer más dinámico y visual algunas operaciones.

A.2.- Compilación del *kernel*

A.2.1.- Introducción

El uso de herramientas específicas de control de tráfico hace que sea necesario tener configurado ciertos parámetros del núcleo o *kernel* de una manera específica. Las configuraciones que vienen por defecto en las distribuciones no siempre soportan este tipo de herramientas, lo que obliga a compilar el *kernel*.

Además de esta particularidad, existen ventajas a favor de la compilación del núcleo. El hecho de que el núcleo sea software libre permite descargar su código fuente y configurarlo con opciones adecuadas a las necesidades y al hardware, lo que supone una optimización y una mejora notable del sistema. Por estos motivos, además de configurar el *kernel* para que soporte las herramientas de control de tráfico, se aprovechará para descargar la última versión y personalizarlo.

A.2.2.- Identificación de hardware

Para garantizar un buen funcionamiento y optimizar el uso del nuevo *kernel*, es necesario determinar el hardware y los controladores que utiliza el sistema. La información referente a estos dispositivos está almacenada bien en binarios o en ficheros virtuales.

A continuación se muestra como visualizar alguno de estos ficheros:

- **Módulos utilizados por el sistema:** a través del binario */sbin/lsmmod*.
- **Tipo de procesador:** a través del fichero virtual */proc/cpuinfo*.
- **Dispositivos PCI:** a través del binario */sbin/lscpi*.
- **Dispositivos USB:** a través del binario */sbin/lsub*.

A.2.3.- Instalación de paquetes necesarios

Una vez recopilada toda la información referente al hardware, es necesario instalar algunos paquetes que serán de utilidad tanto para el proceso de compilación del *kernel* como para posteriores actividades. La gestión de paquetes en sistemas operativos basados en la distribución Red Hat se hace mediante el comando *yum*.

Esta herramienta es la que se encarga de descargar e instalar los paquetes *rpm* (Red Hat Package Manager) de los repositorios publicados en Internet.¹¹

```
[root@centos5-gestorA /]# yum install yum-utils gcc glibc-devel ncurses-devel  
qt-devel gcc-c++ rpm-build
```

¹¹ Para acceder a estos repositorios publicados en Internet es necesario tener acceso a Internet.

A.2.4.- Descarga y configuración del nuevo *kernel*

Para conocer la versión actual del *kernel* en la que se está trabajando se utiliza el comando *uname* de la siguiente manera:

```
[root@centos5-gestorA /]# uname -r
2.6.18-92.1.13
```

Este es el núcleo por defecto incluido en la distribución CentOS 5.2 instalada en el sistema. Como se ha comentado anteriormente, se aprovechará la necesidad de compilar el *kernel* para actualizarlo a la última versión. Para esto, se accede al enlace [35] y se elige la versión más reciente del núcleo. En este caso se descarga la versión 2.6.26.5, se descomprime y se accede a la carpeta creada.

```
[root@centos5-gestorA etc]# wget
http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.26.5.tar.bz2
[root@centos5-gestorA etc]# tar jxvf linux-2.6.26.5
[root@centos5-gestorA etc]# cd linux-2.6.26.5
```

Una manera ágil y dinámica de configurar el *kernel* es con el comando *make* y la opción *xconfig*. Esta instrucción compila y ejecuta una interfaz hecha en *ncurses* que permite examinar la lista de posibles opciones. Otras opciones menos elaboradas gráficamente son *config* o *menuconfig*.

```
[root@centos5-gestorA linux-2.6.26.5]# make xconfig
```

Dentro de este menú se debe configurar el *kernel* habilitando o eliminando funciones, controladores y módulos según las especificaciones de controladores y hardware vistas anteriormente. Como norma general se debe mantener el núcleo lo mas pequeño posible.

Es aconsejable incluir en el núcleo:

- Controladores para dispositivos integrados en la placa basa de uso continuo.
- Controladores de dispositivos de uso continuo, como controladores de disco o de buses de transporte.
- Controladores para sistemas de ficheros.

Por otro lado, es aconsejable excluir del núcleo y compilar como módulo:

- Controladores de dispositivos periféricos.
- Controladores de dispositivos que se extraigan o intercambien con frecuencia.

Como se ha comentado al inicio, en esta compilación hay que prestar especial atención a la configuración de los parámetros referentes al control de tráfico, como son QoS y encolamiento, ya que deben tener unos valores específicos. Tanto la configuración del *kernel* actual como la del nuevo *kernel* después de compilarlo se pueden encontrar en el archivo */boot*. Los parámetros que interesan especialmente se encuentran en el menú:

Networking → Networking support → Networking Options → QoS and/or fair queueing

Las siguientes opciones de configuración del *kernel* en la sección **Networking Options** tienen que estar habilitadas, preferiblemente dentro del núcleo:

- CONFIG_NETLINK
- CONFIG_NETFILTER
- CONFIG_NET_SCHED

En la sección **Networking options, QoS and/or fair queueing** deben ser las siguientes:

- CONFIG_NET_SCH_CBQ
- CONFIG_NET_SCH_HTB
- CONFIG_NET_SCH_SFQ
- CONFIG_NET_SCH_TBF
- CONFIG_NET_SCH_PRIO
- CONFIG_NET_SCH_RED
- CONFIG_NET_SCH_GRED
- CONFIG_NET_SCH_DSMARK
- IG_NET_SCH_INGRESS
- CONFIG_NET_QOS
- CONFIG_NET_CLS
- CONFIG_NET_CLS_TCINDEX
- CONFIG_NET_CLS_F
- CONFIG_NET_CLS_U32
- CONFIG_NET_CLS_POLICE

- CONFIG_NET_CLS_ROUTE4
- CONFIG_NET_EMATCH_U32
- CONFIG_PACKET
- CONFIG_NETFILTER
- CONFIG_IP_NF_FILTER
- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_CONNTRACK
- CONFIG_IP_NF_TARGET_MASQUERADE
- CONFIG_IP_NF_MATCH_MAC

La compilación del nuevo *kernel* se ejecuta con el comando *make* dentro de la carpeta donde se ha descomprimido:

```
[root@centos5-gestorA /]# cd /etc/linux-2.6.26.5  
[root@centos5-gestorA linux-2.6.26.5]# make
```

Esta compilación tarda varios minutos, dependiendo de la capacidad del sistema y de la cantidad de parámetros que se hayan modificado.

Tras verificar en la compilación que el nuevo núcleo no contiene ningún error, se procede a la instalación de los módulos y del núcleo:

```
[root@centos5-gestorA linux-2.6.26.5]# make modules_install  
[root@centos5-gestorA linux-2.6.26.5]# make install
```

Estas instrucciones instalarán el nuevo núcleo y sus archivos correspondientes en el directorio */boot*.

Algunos de los ficheros que contienen información sobre el núcleo son:

- **system.map-2.6.26.5**: Contiene los nombres de todos los símbolos y direcciones del nuevo *kernel*. Se utiliza principalmente para *debugar*.
- **config-2.6.26.5**: Configuración del nuevo *kernel*. Contiene la información de todos los parámetros, tanto si están dentro del núcleo, como módulo o como no presente.

- **initrd-2.6.26.5.img**: Este fichero es un sistema de archivos temporal utilizado por el *kernel* durante el inicio del sistema. Se suele utilizar para hacer los cambios necesarios antes de que el sistema de archivos raíz sea montado.
- **vmlinuz-2.6.26.5**: Es un archivo ejecutable enlazado estáticamente y que contiene el núcleo del sistema en formato ejecutable.

Además, se añade una entrada al fichero de configuración de GRUB, colocándose por detrás de los núcleos anteriormente. El GRUB (Grand Unified Bootloader) es un gestor de arranque múltiple que permite iniciar diferentes *kernels* en un mismo equipo, o incluso diferentes sistemas operativos. Su principal característica es que no es necesario instalar una partición nueva para cada caso. Si se quiere configurar el nuevo núcleo como predeterminado, es necesario cambiar el *kernel* por defecto en el archivo `/boot/grub/grub.conf`.

Por otro lado, en este tipo de distribuciones existe la posibilidad de crear un paquete *rpm* a partir de los binarios recién compilados. Es necesario volver a invocar el mandato *make* desde la ruta donde hemos descomprimido el *kernel*. Una vez se compila el paquete, se instala con el comando *rpm*.

```
[root@centos5-gestorA /]# cd /etc/linux-2.6.26.5
[root@centos5-gestorA linux-2.6.26.5]# make binrpm-pkg
[root@centos5-gestorA /]# cd /usr/src/redhat/RPMS/i386/
[root@centos5-gestorA i386]# ls
kernel-2.6.26.5-2.i386.rpm
[root@centos5-gestorA i386]# rpm -ivh kernel-2.6.26.5-2.i386.rpm
```

B.- Personalización del sistema

Una vez se ha instalado el sistema operativo y se ha recompilado según sus necesidades y el hardware que tiene que soportar, es aconsejable preparar correctamente el sistema para optimizar su uso y su seguridad.

B.1.- Gestión de usuarios

Para evitar identificarse siempre con el usuario *root*, es conveniente crear otro usuario con algunos privilegios de administrador. El uso abusivo del usuario *root* puede conllevar ciertos problemas, ya que al ser súper-usuario tiene plenos poderes sobre el sistema. Esto puede causar que se eliminen archivos necesarios, o incluso que el sistema se vuelva inestable. Utilizando otro usuario, se evitan riesgos de seguridad y es posible continuar ejecutando tareas administrativas con el uso del comando *su*. A continuación se muestra como crear el nuevo usuario, configurar su contraseña y añadirlo al grupo *wheel*.

```
[root@centos5-gestorA /]# useradd informatica
[root@centos5-gestorA /]# gpasswd *****
[root@centos5-gestorA /]# gpasswd -a informatica wheel
```

El uso del grupo *wheel* es una forma sencilla y efectiva de mejorar la seguridad del sistema. Debido a que no es aconsejable identificarse como *root*, se añaden los usuarios que deben efectuarán tareas administrativas al grupo *wheel*. Después, se configura para que el comando *su* solo pueda ser utilizado por este grupo de usuarios. Este comando permite cambiar la identidad de un usuario a otro, y aunque es posible cambiar a cualquier usuario, principalmente se utiliza para identificarse como súper-usuario. De esta manera se asegura que tan solo el grupo *wheel* y el propio *root* puedan efectuar tareas administrativas.

Por defecto, el propietario de este comando es *root*, y cualquier usuario tiene permisos para ejecutarlo. Una manera de cambiarlo es configurando los permisos del binario para que ningún usuario a parte del propietario pueda hacer uso de él. Después, se configura el grupo *wheel* como propietario.

```
[root@centos5-gestorA /]# ls -l /bin/su
-rwsr-xr-x 1 root root 24120 may 24 17:19 /bin/su
[root@centos5-gestorA /]# chmod 4750 /bin/su
[root@centos5-gestorA /]# ls -l /bin/su
-rwsr-x--- 1 root root 24120 may 24 17:19 /bin/su
[root@centos5-gestorA /]# chgrp wheel /bin/su
[root@centos5-gestorA /]# ls -l /bin/su
-rwxr-x--- 1 root wheel 24120 may 24 17:19 /bin/su
```


Para dotar de más poder a este grupo, se edita el fichero *sudoers* para que pueda ejecutar cualquier tipo de comando sin necesidad de identificación. Además, si se edita con la opción *visudo*, antes de guardar los cambios se asegura de que no haya errores sintácticos ni que se estén efectuando ediciones simultáneas del mismo fichero. Por otro lado, ya a que el fichero *sudoers* tan solo tiene permisos de lectura, es necesario cambiarle los permisos antes de editarlo y volver a dejarlos en su configuración original después de hacer los cambios (Figura B.1).

```
[informatica@centos5-gestorA ~]$ su root
Contraseña: *****
[root@centos5-gestorA informatica]# chmod 777 /etc/sudoers
[root@centos5-gestorA informatica]# /usr/sbin/visudo
#editar el fichero#
[root@centos5-gestorA informatica]# chmod 0440 /etc/sudoers
```

```
## Allows people in group wheel to run all commands
##wheel ALL=(ALL) ALL
## Same thing without a password
%wheel ALL=(ALL) NOPASSWD: ALL
```

Figura B.1.- Copia parcial del fichero */etc/sudoers*

Para comprobar que el cambio se ha hecho correctamente, se puede llamar al comando *sudo* para que muestre que los ficheros a los que tiene acceso. En este caso, el usuario *informatica* tiene permisos para todos los comandos sin necesidad de introducir contraseña, de la misma manera que *root*.

```
[informatica@centos5-gestorA ~]$ sudo -l
User informatica may run the following commands on this host:
(ALL) NOPASSWD: ALL
```

También es necesario agregar algunas rutas a la variable *PATH* del nuevo usuario. Para hacerlo de forma puntual en la sesión actual, basta con modificar por consola el valor de la variable. Antes es aconsejable mirar que rutas tiene este usuario agregadas haciendo un *echo* de la variable para asegurarse de que rutas son necesarias.

Por ejemplo, en el caso del usuario informática, le agregamos al *PATH* la ruta */sbin*. Esta ruta es muy útil ya que contiene gran cantidad de binarios de configuración de red, como *ifconfig*, *route* o *iptables*.

```
[informatica@centos5-gestorA ~]$ echo $PATH /usr/lib/qt-3.3/bin
:/usr/kerberos/bin:/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/informatica
/bin
[informatica@centos5-gestorA ~]$ PATH=$PATH:/sbin
[informatica@centos5-gestorA ~]$ export PATH
```

Para que estos cambios en la variable se conserven de forma permanente, hay que editar el perfil del usuario añadiéndole las rutas necesarias como se muestra en la figura B.2.

```
[informatica@centos5-gestorA~]$ vi /home/Informatica/.bash_profile
```

```
# .bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
. ~/.bashrc
fi
# User specific environment and startup programs
PATH=$PATH:$HOME/bin:/sbin
export PATH
```

Figura B.2.- Copia del fichero *.bash_profile* del usuario informatica

B.2.- Gestión de servicios

Para poder gestionar los servicios del sistema correctamente, es importante la elección del *runlevel*. Para conocer el nivel en el que se está trabajando, se puede hacer con el comando *runlevel*.

```
[root@centos5-gestorA ~]# runlevel
N 5
```

Para este caso, se elige el nivel 5, que es el que corresponde al modo gráfico multiusuario X11. Con este nivel es posible trabajar en un entorno gráfico como GNOME, con lo que se agilizan algunas tareas. Si por cualquier motivo fuese necesario arrancar el sistema con otro *runlevel*, bastaría con editar el archivo */etc/inittab* cambiando el identificador por el del nivel requerido.

Una vez determinado el *runlevel* con el que trabajará el sistema habitualmente, se debe gestionar qué servicios son necesarios que arranquen automáticamente con la máquina y cuales no. Para conocer el estado de los distintos servicios, se utiliza el comando *chkconfig* con la opción *list*.

```
[root@centos5-gestorA ~]# chkconfig --list
```

httpd	0:desactivado	1:desactivado	2:desactivado	3:desactivado
4:desactivado	5:desactivado	6:desactivado		
mysqld	0:desactivado	1:desactivado	2:desactivado	3:desactivado
4:desactivado	5:desactivado	6:desactivado		
network	0:desactivado	1:desactivado	2:activo	3:activo
4:activo	5:activo	6:desactivado		
iptables	0:desactivado	1:desactivado	2:activo	3:activo
4:activo	5:activo	6:desactivado		
bluetooth	0:desactivado	1:desactivado	2:activo	3:activo
4:activo	5:activo	6:desactivado		

Figura B.3.- Estado de algunos servicios según el *runlevel*

Esta instrucción lista los servicios que se están ejecutando y en que *runlevels* arrancan automáticamente. En la figura B.3 se muestra un ejemplo de algunos de estos servicios. Se puede ver como en el nivel 5 (el que anteriormente se ha definido como por defecto) los servicios *httpd* y *mysqld* están desactivados. Como se verá mas adelante, estos servicios son necesarios para que funcione la herramienta de gestión de tráfico *WebHTB*, por lo que sería conveniente que se iniciaran automáticamente. Por otro lado, existen otros servicios como es el caso de *bluetooth*, que no son necesarios y que consumen recursos de la máquina.

Para activar o desactivar estos servicios en un determinado *runlevel*, se utiliza también el comando *chkconfig*, indicando el nombre del servicio y los niveles donde se quiere cambiar la configuración de arranque. Este procedimiento hay que seguirlo para todos los servicios ya instalados o para los que se instalen en un futuro.

```
[root@centos5-gestorA ~]# chkconfig --level 345 httpd on
[root@centos5-gestorA ~]# chkconfig --level 345 mysqld on
[root@centos5-gestorA ~]# chkconfig --level 2345 bluetooth off
```

Además de la gestión de estos servicios, también es necesario activar el *routing* y desactivar las funciones *SELinux* del servidor.

El *routing* es necesario para que la máquina sea capaz de enrutar paquetes entre sus interfaces según su tabla de rutas. Esta herramienta se puede activar bien de manera puntual por consola o de manera permanente editando el archivo */etc/sysctl.conf* (Figura B.4). Para ambas soluciones, lo que se hace es habilitar el registro *ip_forward*.

```
[root@centos5-gestorA ~]# echo "1" > /proc/sys/net/ipv4/ip_forward
```

```
[root@centos5-gestorA ~]# vi /etc/sysctl.conf

# Kernel sysctl configuration file for Red Hat Linux
#
# For binary values, 0 is disabled, 1 is enabled.  See sysctl(8) and
# sysctl.conf(5) for more details.

# Controls IP packet forwarding
net.ipv4.ip_forward = 1
```

Figura B.4.- Copia parcial del fichero */etc/sysctl.conf*

Para que el gestor sea capaz de acceder y modificar algunos ficheros referentes a la aplicación *WebHTB*, es necesario que el *SELinux* (Security Enhanced Linux) esté deshabilitado. *SELinux* es una herramienta de seguridad integrada en los *kernels* 2.6.x para las distribuciones del grupo Red Hat.

Esta herramienta permite establecer reglas para aceptar o denegar el acceso a distintos archivos del sistema por distintos usuarios. Si no se ha deshabilitado en el proceso de instalación del sistema operativo, es necesario hacerlo ahora. Para comprobar el estado actual, se puede hacer con el comando *getenforce*.

```
[root@centos5-gestorA ~]# getenforce
Enforcing
```

En este caso en concreto, se observa que esta función está activada. Para deshabilitarla, es necesario editar el fichero de configuración de *SELinux* y resetear el servidor (Figura B.5).

```
[root@centos5-gestorA ~]# vi /etc/sysconfig/selinux
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled  - SELinux is fully disabled.
SELINUX=disabled
```

Figura B.5.- Copia parcial del fichero */etc/sysconfig/selinux*

B.3.- Configuración de tablas de rutas

Para poder redireccionar el tráfico, en ocasiones es necesario crear una tabla de rutas en el servidor. Estas rutas se pueden configurar de manera dinámica o estática. De manera dinámica significa que tan solo tendrán vigencia hasta que se reinicie el servicio de red (servicio network).

Se tiene el ejemplo de querer añadir una ruta a un rango de direccionamiento IP 192.168.4.0/24 a través de la interfaz 192.168.2.120. Una posible manera de configurar la ruta en el gestor de manera dinámica sería la siguiente:

```
[root@centos5-gestorA /]# route add -net 192.168.4.0 netmask 255.255.255.0 gw
192.168.2.120
```

Una ruta estática es una ruta que permanece activa aún después de reiniciar el servicio. Para configurarla, hay que crear un fichero en el directorio donde se tienen las configuraciones de red con las diferentes rutas (*/etc/sysconf/network-scripts/*). El nombre de esta carpeta debe ser de la forma *route-ethx*, donde x es el número que identifica a la interfaz donde se quiere insertar la ruta.

La primera manera de construir este fichero es indicando implícitamente la puerta de enlace, la máscara y la dirección de la red a la que se quiere acceder, seguidos de un número que identifica la ruta. En la figura B.6 se muestra un ejemplo de cómo introducir la ruta número 0.

```
GATEWAY0=192.168.2.120
NETMASK0=255.255.255.0
ADDRESS0=192.168.4.0
```

Figura B.6.- Ejemplo 1 del fichero *route-eth0*

Otra manera más directa es indicando en una misma línea de código toda la información referente a la ruta, nombrando a través de que puerta de enlace hay que recurrir para acceder a la dirección requerida. En la figura B.7 se observa un ejemplo de este archivo.

```
192.168.4.0/24 via 192.168.2.120
```

Figura B.7.- Ejemplo 2 de fichero *route-eth0*

B.4.- Instalación de herramientas de utilidad

Para mejorar y facilitar el uso del sistema en momentos y acciones puntuales, existen distintas herramientas. Dentro de estas herramientas se incluyen *webmin*, *wireshark* y *cacti*.



Webmin [35] es una aplicación basada en un entorno Web y escrito en lenguaje de programación Perl que facilita la administración de sistemas Linux. Su principal característica es que permite gestionar gran variedad de programas y servicios de forma gráfica y sencilla a partir de formularios. Estos formularios se utilizan para editar los ficheros de configuración de los distintos servicios, aunque también existe la posibilidad de editarlos a partir del editor de texto. Esta herramienta está construida de forma modular, con lo que es posible descargarse módulos para diferentes funciones o programas. Entre estas funciones están las de gestionar usuarios, las interfaces de red, el servidor Web *Apache* o *MySQL*.

Este servicio se utilizará en casos puntuales para determinados servicios o programas y cuando no sea posible hacerlo de otro modo, ya que es altamente aconsejable ejecutar las tareas administrativas a través de la consola de comandos. De esta manera se tiene un conocimiento más amplio del funcionamiento de la aplicación en particular y del sistema en general.

Es posible descargar la última versión del archivo en distintos formatos desde el enlace oficial. Una vez descomprimido, se inicia el proceso de instalación.

```
[root@centos5-gestorA]# cd /etc
[root@centos5-gestorA etc]# tar xvf webmin-1.430.tar.gz
[root@centos5-gestorA etc]# cd webmin-1.430
[root@centos5-gestorA webmin-1.430]# ./setup.sh
```

Esta es una instalación atendida, en la que se le pregunta al administrador información sobre la configuración y ubicación de directorios, como se muestra en la figura B.8.

```
Installing Webmin in /etc/webmin-1.430 ...
Config file directory [/etc/webmin]:
Log file directory [/var/webmin]:
Full path to perl (default /usr/bin/perl):
Operating system name: CentOS Linux
Operating system version: 5.2
Web server port (default 10000):
Login name (default admin):
Login password: *****
Password again: *****
Start Webmin at boot time (y/n): y
Webmin has been installed and started successfully. Use your web
browser to go to http://localhost:10000/
```

Figura B.8.- Copia parcial del asistente de instalación de *webmin*.



Wireshark [36] es un analizador de protocolos (*sniffer*) utilizado para identificar y analizar el tráfico de una interfaz de red en un determinado momento. Permite capturar paquetes y mostrarlos gráficamente en tiempo real, aunque también tiene la opción de almacenar la información para su posterior tratamiento. Para ambas acciones, se requieren permisos de ejecución especiales, y por esto que se ejecuta con permisos de súper-usuario. Se diferencia del resto de analizadores principalmente por su amigable interfaz gráfica, su gran variedad de filtros y su detallada información de cada paquete.

Esta aplicación será utilizada para comprobar el comportamiento del sistema durante el proceso de pruebas y testing. Como se ha comentado, debe ser utilizada de manera puntual, ya que mantener analizando paquetes un periodo de tiempo demasiado amplio requiere mucho espacio en disco, con lo que se puede llegar a bloquear el sistema

Las últimas versiones de este analizador se pueden conseguir en el enlace oficial, tanto para sistemas operativos basados en Windows como para Linux. Sin embargo, desde el repositorio *rpbone* es posible descargarse la última versión en formato *rpm* para un escritorio GNOME. Al ser un archivo *rpm*, tan solo basta con descargarlo e instalarlo utilizando el comando *yum* como se ha hecho en otras ocasiones.

```
[root@centos5-gestorA ~]# cd /etc/  
[root@centos5-gestorA etc]#yum install wireshark-gnome-1.0.3-4.el5_2.i386.rpm
```



Cacti [37] es una herramienta para la representación gráfica de datos de monitorización de red, basada en RRDTool. Permite representar información sobre carga del procesador, temperatura o procesos activos, pero para este proyecto su principal función será la de monitorizar el tráfico en la red. Para obtener información sobre los distintos dispositivos de red utiliza el protocolo SNMP, lo que supone tener el servicio activo y correctamente configurado. Por otro lado, para almacenar la información utiliza bases de datos MySQL.

Es posible instalar *cacti* a través del comando *yum*, pero para ello es necesario descargar e instalar el repositorio DAG. También se requieren los paquetes RRDTool y net-snmp.

```
[root@centos5-gestorA etc]# wget http://dag.wieers.com/rpm/packages/rpmforge-  
release/rpmforge-release-0.3.6-1.el5.rf.i386.rpm  
[root@centos5-gestorA etc]# rpm -Uvh rpmforge-release-0.3.6-1.el5.rf.i386.rpm  
[root@centos5-gestorA etc]# yum install net-snmp rrdtool cacti
```

Para configurar el protocolo SNMP, hay que editar el fichero */etc/snmp/snmpd.conf* en función de la interfaces de red que se quieran utilizar.

```
[root@centos5-gestorA etc]# vi /etc/snmp/snmpd.conf  
#editar archivo#  
[root@centos5-gestorA etc]# /etc/init.d/snmpd start  
[root@centos5-gestorA etc]# chkconfig snmpd on
```

En lo referente a MySQL, hay que crear una base de datos y un usuario específicos para la herramienta. En este caso, se crea la base de datos *cacti* y el usuario *cactiuser*. Las instrucciones de cómo hacerlo se pueden encontrar en el anexo *D.- Manual de instalación de WebHTB*.

Esta configuración hay que utilizarla para editar el fichero de *cacti config.php*. También es necesario dar permisos al usuario *Apache* en la carpeta donde se encuentra la herramienta y reiniciar el servicio para que los cambios tengan efecto.

```
[root@centos5-gestorA etc]# vi /var/www/cacti/incluye/config.php
#editar archivo#
[root@centos5-gestorA etc]# chown -R apache. /var/www/cacti
[root@centos5-gestorA etc]# service httpd restart
```

Una vez instalado y configurado, basta con acceder al servicio a través del navegador Web entrando en el recurso *cacti*. Si se quiere poder acceder por red a este servicio, hay que editar la configuración *httpd* para *cacti* en el fichero */etc/httpd/conf.d/cacti.conf*.

En la figura B.9 se muestra un gráfico que monitoriza el tráfico en tiempo real de una de las interfaces del servidor. Los datos se obtienen haciendo *polling* con un periodo de 5 minutos.

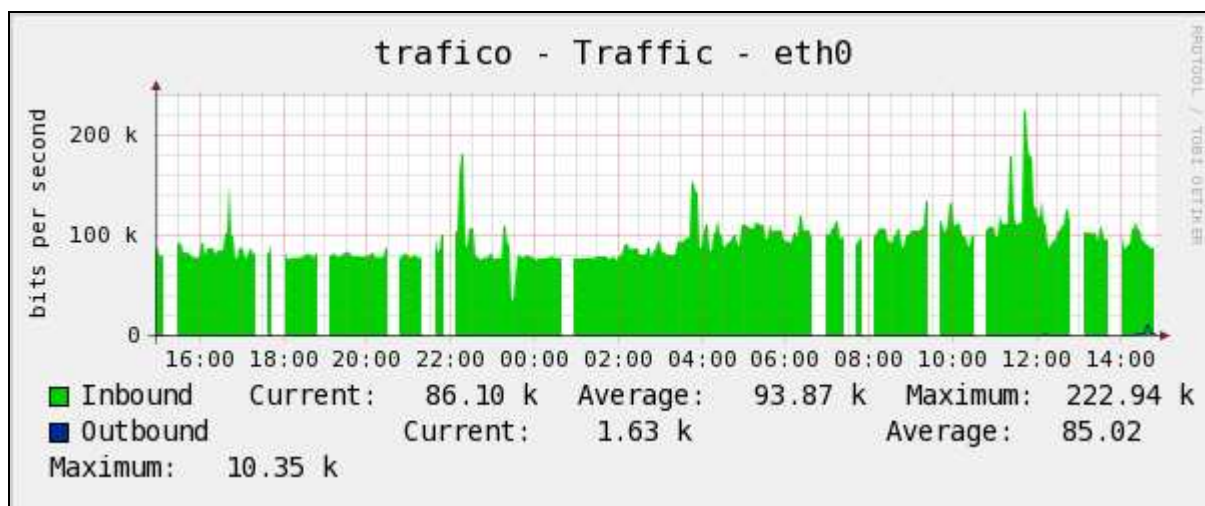


Figura B.9.- Captura de monitorización del tráfico con *cacti*

C.- Mantenimiento del sistema

C.1.- Copia de seguridad

Durante el transcurso del proyecto, el servidor no ha estado conectado a la red de la empresa de una manera definitiva. Como no era posible tener una copia periódica del sistema con el cliente Veritas, se han ido realizando esas copias con la herramienta Norton Ghost v8, que ha permitido hacer una copia completa del disco utilizado a otro disco idéntico de *backup*. De esta manera se ha tenido copia de seguridad en todo momento, y especialmente antes de hacer cambios y actualizaciones importantes.

C.2.- Control del sistema

Es importante mantener periódicamente un control de los diferentes dispositivos hardware del sistema, con el fin de evitar fallos, sobrecargas o llenados de memoria. Para ver el estado del sistema existen diversos comandos:

- **Comando *w***: Muestra por pantalla la carga media del procesador en tiempo real.
- **Comando *ps aux***: Esta instrucción muestra información sobre el estado de los distintos procesos que corren en la máquina.
- **Comando *top***: Muestra un resumen de la información anterior. La salida de este comando se muestra en la figura C.1.

```
[root@centos5-gestorA /]# top
```

```
top - 19:15:25 up 9 days, 3:07, 2 users, load average: 0.06, 0.07, 0.05
Tasks: 135 total, 1 running, 134 sleeping, 0 stopped, 0 zombie
Cpu(s): 3.8%us, 0.2%sy, 0.0%ni, 95.8%id, 0.0%wa, 0.2%hi, 0.0%si, 0.0%st
Mem: 2050968k total, 1889316k used, 161652k free, 383720k buffers
Swap: 4096532k total, 0k used, 4096532k free, 829144k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
12927 informat  20   0 61628  13m 8872  S   6   0.7   0:02.03  gnome-terminal
 5711 root       20   0 337m  16m 7812  S   1   0.8  13:42.31  Xorg
    1 root       20   0 2060   632  544  S   0   0.0   0:01.11  init
    2 root       15  -5     0     0     0  S   0   0.0   0:00.00  kthreadd
    3 root       RT  -5     0     0     0  S   0   0.0   0:00.47  migration/0
    4 root       15  -5     0     0     0  S   0   0.0   0:07.31  ksoftirqd/0
```

Figura C.1.- Salida del comando *top*

- **Comando *df***: Este comando, con la opción *h*, muestra el espacio libre en las distintas unidades de ficheros (Figura C.2).

```
[root@centos5-gestorA /]# df -h
```

S.ficheros	Tamaño	Usado	Disp	Uso%	Montado en
/dev/hda2	76G	2,5G	70G	4%	/
/dev/hda7	48G	706M	45G	2%	/var
/dev/hda5	7,6G	2,5G	4,7G	35%	/usr
/dev/hda3	9,5G	384M	8,6G	5%	/tmp
/dev/hda1	122M	24M	92M	21%	/boot
tmpfs	1002M	0	1002M	0%	/dev/shm

Figura C.2.- Salida del comando *df*

C.3.- Logs del sistema

A la hora de resolver problemas o incidencias, los sistemas GNU/Linux ponen a disposición del usuario los distintos logs del sistema. Estos logs o registros arrancan con el *script /etc/init.d/syslog* y normalmente se guardan en el directorio */var/log*. En la figura C.3 se listan los archivos que contiene este directorio.

```
[root@centos5-gestorA /]# cd /var/log/
[root@centos5-gestorA log]# ls
```

anaconda.log	boot.log.2	cron	dmesg	maillog	messages.1	ppp
rpm_pkgs.4	secure.3	spooler.4		yum.log		
anaconda.syslog	boot.log.3	cron.1	faillog	maillog.1	messages.2	prelink
samba	secure.4	tallylog				
anaconda.xlog	boot.log.4	cron.2	gdm	maillog.2	messages.3	rpm_pkgs
scrollkeeper.log	spooler	vbox				
audit	btmp	cron.3	httpd	maillog.3	messages.4	rpm_pkgs.1
secure	spooler.1	wtmp				
boot.log	conman	cron.4	lastlog	maillog.4	mysqld.log	rpm_pkgs.2
secure.1	spooler.2	Xorg.0.log				
boot.log.1	conman.old	cups	mail	messages	pm	rpm_pkgs.3
secure.2	spooler.3	Xorg.0.log.old				

Figura C.3.- Logs del sistema

Entre estos ficheros se pueden destacar:

- **messages**: Logs que llegan con prioridad información, notificación o aviso.
- **dmesg**: Este registro almacena información que genera el *kernel* durante el arranque del sistema.
- **boot.log**: Información sobre el proceso de arranque de la máquina en general.
- **mysqld.log**: Información sobre el servicio MySQL (accesos a bases de datos, usuarios, etc.)

- **yum.log:** Este log registra todos los paquetes instalados mediante el comando *yum*.
- **httpd:** En esta carpeta se almacena información sobre los accesos al servidor *Apache* y errores.

C.4.- Reinicio del sistema

Existen situaciones en las que, ya sea por fallos de la máquina o para tareas de mantenimiento, es necesario reiniciar el sistema. En ocasiones basta con reiniciar el servicio en concreto y en otras es necesario reiniciar la máquina al completo.

Para el caso del servicio de gestión de ancho de banda, el servicio principal es *webhtb*, ubicado en */etc/init.d/webhtb*. Sin embargo, existen otros servicios necesarios para que este funcione, como son el servicio *Apache* o el MySQL. Para reiniciar alguno de estos servicios:

```
[root@centos5-gestorA /]# /etc/init.d/webhtb restart
[root@centos5-gestorA /]# service webhtb restart
[root@centos5-gestorA /]# service httpd restart
[root@centos5-gestorA /]# service mysqld restart
```

Si es necesario reiniciar la máquina, en principio todos los servicios necesarios deberían arrancar automáticamente si se han seguido las especificaciones del anexo *B.- Personalización del sistema*. Para ver los servicios activos basta con ejecutar el comando *ps* como se ha comentado anteriormente.

D.- Manual de instalación de *WebHTB*

Durante el proceso de instalación de la aplicación *WebHTB* han ido surgiendo incidencias que se han ido solucionando. Por este motivo, este apartado se va a diferenciar entre el proceso de instalación y la identificación y resolución de estas incidencias.

D.1.- Instalación de la aplicación *WebHTB*

Antes de empezar este proceso, es necesario que el sistema cumpla unos prerequisites y que tenga instalados algunos paquetes necesarios.

- Es necesario tener configuradas las opciones de QoS y encolamiento del *kernel* tal y como se ha descrito en el anexo A.- *Instalación y configuración del sistema operativo*.
- Es recomendable copiar los binarios de la carpeta *tc* del paquete *iproute2* en el directorio */sbin*.
- Es recomendable que la versión del *kernel* sea como mínimo la 2.6.25

WebHTB utiliza lenguaje PHP, bases de datos con MySQL y *Apache* para su entorno Web, por lo que se necesitan algunos paquetes adicionales.

```
[root@centos5-gestorA ~]# yum install flex mysql mysql-server php  
php-mysqld httpd
```

El primer paso es arrancar el servicio MySQL y crear una base de datos para *WebHTB*. Como más adelante se instalarán nuevas versiones, se crea una base de datos para cada versión. La primera versión a instalar es la v2.1. Al acabar, se comprueba que se ha creado correctamente.

```
[root@centos5-gestorA ~]# service mysqld start  
[root@centos5-gestorA ~]# mysql  
mysql> create database webhtb21;  
mysql> show databases;
```

```
+-----+  
| Database          |  
+-----+  
| information_schema|  
| mysql             |  
| test              |  
| webhtb21         |  
+-----+
```

Anexos

Para evitar utilizar el súper-usuario, se crea un usuario específico para esta aplicación, dotándole de una contraseña de seguridad. Una vez creado se comprueban sus permisos.

```
mysql> use webhtb21;
mysql> create user webhtb_user identified by '*****';
mysql> show grants for webhtb_user;
```

```
+-----+
| Grants for webhtb_user@% |
+-----+
| GRANT USAGE ON *.* TO 'webhtb_user'@'%' IDENTIFIED BY PASSWORD '59a04797707760ac' |
+-----+
```

Seguidamente, se establece el usuario *root* como el usuario administrador global de todas las bases de datos, incluso de las que todavía no se han creado. Este usuario tendrá todos los privilegios sobre estas, así como la capacidad de otorgar privilegios a otros.

```
mysql> grant all privileges on *.* to root@localhost' identified by '*****'
with grant option;
```

Con la base de datos y el usuario MySQL creados, se pasa a descargar e instalar *WebHTB*. Se descarga el archivo comprimido desde el enlace oficial [31] y se guarda en el directorio que *Apache* tiene configurado para aplicaciones Web. Una vez descargado, se descomprime para que los distintos archivos puedan ser ejecutados.

```
[root@centos5-gestorA ~]# cd /var/www/html/
[root@centos5-gestorA html]# tar -jxvf WebHTB_v2.1.bz2
[root@centos5-gestorA html]# cd webhtb/
```

Dentro de la carpeta descomprimida se encuentran todos los archivos y ficheros de la aplicación, como se observa en la figura D.1.

add_classes.php	del_classes.php	que_show.php
addIface.php	delClient.php	release_notes.txt
anc.php	delIface.php	save_order_cl.php
apply_changes.php	delSrcDst.php	setup
a_r_ip_all.php	docs	show_autor.php
AUTOR	edit_classes.php	showClients.php
bin	images	show_howto.php
ch_a_class.php	include	show_lic.php
change_client_iface_class.php	index.php	shutdown.php
changeDstSrt.php	interfaces.php	startup.php
ch_cl_if_cl.php	ips_allowed.php	stopQ_parser.php
ch_limits.php	langs	xml
choose_lang.php	licence.txt	xml_parser.php
config	que_parser.php	

Figura D.1.- Contenido de la carpeta *WebHTB* v2.1

Aunque para solucionar alguno de los problemas de la aplicación se ha revisado el código de alguno de estos ficheros, este no es uno de los objetivos del proyecto. Por lo tanto, no se va a entrar en detalle a comentar el código fuente de la herramienta. Solo comentar que en el directorio */var/www/html/webhtb/docs/* hay documentos de utilidad para la instalación de la aplicación.

Finalmente, para que el servidor Web reconozca los nuevos ficheros y directorios, es necesario reiniciar el servicio *Apache*.

```
[root@centos5-gestorA /]# service httpd restart
```

Para dar paso a la configuración de *WebHTB* hay que acceder mediante un explorador Web al fichero de configuración (*127.0.0.1/setup/index.php*). En esta ruta aparece un cuadro de configuración, donde hay que introducir los valores MySQL configurados anteriormente. Si la instalación es satisfactoria, aparece por pantalla un mensaje como el de la figura D.2.



Figura D.2.- Pantalla de configuración de *WebHTB*

En este punto de la instalación es necesario volver a editar el archivo *sudoers*. Esta nueva modificación es para que el usuario del servidor Web pueda tener acceso a los directorios donde se almacenan los archivos de configuración. En concreto, hay que agregarle permisos al usuario *Apache* para que tenga permisos sobre el archivo *qos.sh* ubicado en la carpeta temporal sin necesidad de autenticarse mediante contraseña (Figura D.3). No hay que olvidar dejar el archivo con los permisos adecuados al acabar la edición.

```
[root@centos5-gestorA /]# visudo
```

```
"apache          ALL = NOPASSWD: /tmp/qos.sh"
```

Figura D.3.- Copia parcial del fichero *sudoers*

Por otro lado, también es necesario otorgarle permisos de escritura y ejecución para cualquier usuario a la carpeta *xml*.

```
[root@centos5-gestorA /]# chmod 777 /var/www/html/webhtb/xml
```

D.2.- Identificación y resolución de incidencias en el proceso de instalación

D.2.1.- Error al acceder al archivo de instalación

El primer error aparece al intentar configurar los parámetros MySQL en el fichero de configuración de *WebHTB*. Al acceder a este fichero a través del navegador aparece un mensaje de error que indica que el usuario no tiene permisos para ejecutar el binario *ifconfig*.

Después de comprobar el código fuente y buscar información sobre el error y sobre las posibles causas de este, se descubrió que el problema era que el usuario *Apache* no puede ejecutar determinados binarios ni acceder a determinadas carpetas mediante un fichero con extensión PHP, aunque se le otorguen los permisos apropiados o se le haga propietario. En cambio, este mismo usuario si que es capaz de ejecutar los mismos binarios y acceder a las mismas carpetas si lo hace a través de la consola de comandos.

Para comprobar la causa del problema, se creó un sencillo fichero con extensión PHP (Figura D.4) en el que se comprueba si los binarios son ejecutables en este entorno.

```
<?
if(is_executable("/sbin/ifconfig")){
    echo "El comando ifconfig es ejecutable para este usuario ";
}else{
    echo "El comando ifconfig no es ejecutable para este usuario";
}
?>
```

Figura D.4.- Fichero de prueba *test.php*

Al ejecutar este fichero desde el servidor Web aparece por pantalla el mismo mensaje de que el binario *ifconfig* no es ejecutable para este usuario. El usuario en cuestión es sobre el que corre el servidor Web, es decir, el usuario *Apache*.

Además de esta prueba, se hicieron otras para intentar identificar el error. Por ejemplo, se creó otro fichero en el que se accedía a la carpeta */tmp* para copiar o crear un fichero (Figura D.5).

```
<?  
shell_exec (touch /tmp/test)  
>
```

Figura D.5.- *Script de prueba test2.php*

Si se ejecuta este otro fichero en el servidor Web se puede ver en el log de *Apache* como el usuario *Apache* no tiene permisos en la carpeta */tmp*. En cambio, es posible hacer la misma acción si se efectúa directamente desde la consola como usuario *Apache*, sin hacerlo mediante un fichero PHP.

Finalmente se descubrió que el problema estaba en que el servicio SELinux no permite que ciertos usuarios ejecuten algunos comandos de configuración del sistema mediante archivos con extensión PHP o que accedan de la misma manera a algunas carpetas. Una posible solución es deshabilitar el servicio SELinux, como se muestra en el anexo *B.- Personalización del sistema*.

D.2.2.- Error con permisos de *sudoers*

Si una vez se modifica el archivo *sudoers* no se vuelven a reestablecer los permisos necesarios, no es posible acceder a la aplicación. Este error, así como todos los que tengan relación con el servidor Web, se pueden encontrar en el fichero *log* del servicio *httpd* (Figura D.6).

```
[root@centos5-gestorA /]# cat /etc/httpd/logs/error_log
```

```
[Wed Oct 01 17:10:40 2008] [error] [client 127.0.0.1] PHP Notice: Undefined  
variable: content in /var/www/html/webhtb/setup/save.php on line 239, referer:  
http://127.0.0.1/webhtb/setup/index.php  
sudo: /etc/sudoers is mode 0777, should be 0440
```

Figura D.6.- Error de permisos de *sudoers*

Tal y como indica el log, para solucionarlo tan solo hay que volver a configurar los permisos por defecto.

D.2.3.- Error al no definir *tty*

Una vez definidos todos estos permisos, si se vuelve a intentar entrar por Web a la aplicación vuelven a surgir problemas. Si se comprueba el fichero de logs, se observa un nuevo error que corresponde con el de la figura D.7.


```
sudo: sorry, you must have a tty to run sudo
```

Figura D.7.- Error del terminal TTY

Este error quiere decir que es necesario definir una consola o terminal TTY donde lanzar el comando *sudo*. El TTY (teletypewriter) es un dispositivo de entrada que permite teclear y enviar caracteres alfanuméricos.

Para solucionar este problema hay que volver a editar el fichero *sudoers* y comentar o eliminar la línea en la que se pide el terminal tty por defecto (Figura D.8).

```
[root@centos5-gestorA /]# visudo
```

```
#Defaults requiretty in /etc/sudoers
```

Figura D.8.- Copia parcial del fichero */etc/sudoers*

E.- Manual de usuario de *WebHTB*

Este manual pretende explicar el uso de la herramienta *WebHTB* a través de su interfaz Web. Esta herramienta es sencilla de utilizar gracias a su cuidada interfaz Web, a su menú de pestañas desplegadas y a los comentarios que aparecen al desplegar cada una de las ventanas. A continuación se muestra como utilizar estos menús con un sencillo ejemplo.

E.1.- Añadir interfaces

La aplicación detecta automáticamente las interfaces existentes en el momento de arrancar el sistema. Como se comentó en el capítulo 4.- *Control de tráfico en Linux*, toda disciplina de cola con clases debe estar asociada a una interfaz de red, por lo que es necesario seleccionar las que se quieran gestionar. En la pestaña **Interfaces** → **Añadir interfaz** se puede ver un listado de las interfaces disponibles. Si la interfaz se ha añadido correctamente, aparecerá en el panel general, desaparecerá de la lista de interfaces disponibles y se mostrará un mensaje conforme la interfaz se ha añadido correctamente. En el ejemplo se tienen las interfaces *eth0*, *eth1* y *lo*, y se quiere añadir la *eth0* (Figura E.1).

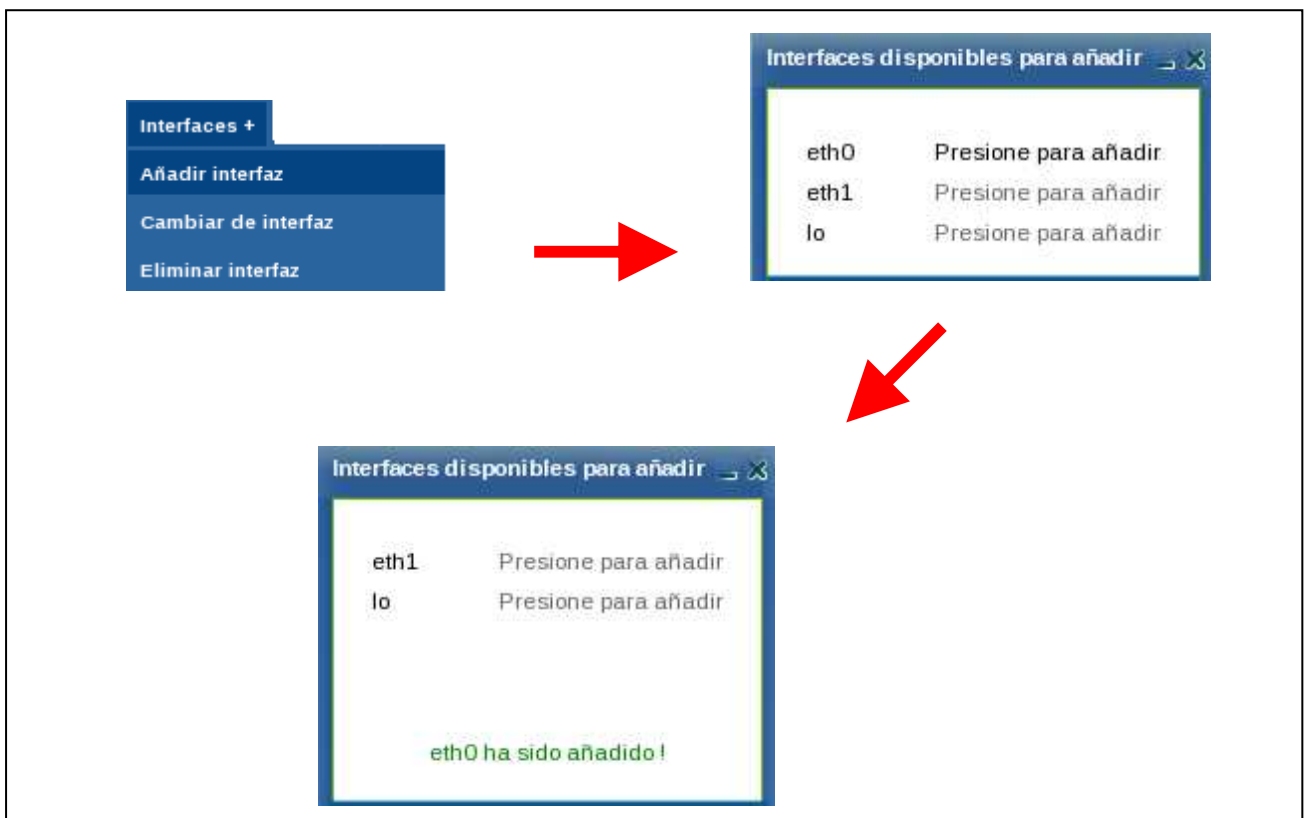


Figura E.1.- Secuencia para añadir una interfaz

E.2.- Añadir clase

En *WebHTB*, existen principalmente cuatro tipos de clases con su propia nomenclatura:

- Clase *root*
- Clase *default*
- Clase madre o clase
- Clase hija o cliente

Tanto la clase *root* como la clase *default* están configuradas por defecto en cada una de las interfaces que se añadan. Sin embargo, las clases y los clientes es necesario añadirlos y configurarlos según las necesidades.

Para añadir una clase hay que acceder a la pestaña Clases → Añadir clases y rellenar los parámetros indicados. Los parámetros referentes a anchos de banda como son *rate* y *ceil* se especifican en kbps y deben ser múltiplos de 8 (Figura E.2).

- **Nombre:** Nombre de la clase
- **Bandwidth (rate):** Ancho de banda mínimo requerido.
- **Limit (ceil):** Ancho de banda máximo que puede utilizar esta clase.
- **Burst:** El valor 0 calcula el *burst* por defecto de la clase.
- **Priority (prio):** Prioridad de la clase. En caso de congestión, las clases con más prioridad desencolarán primero sus paquetes, siendo 0 el valor más prioritario.
- **Que:** Disciplina de cola simple asignada a la *leaf class*. Es la cola que se encarga de desencolar los paquetes de una misma clase. Se puede elegir entre PFIFO, SFQ y ESFQ.

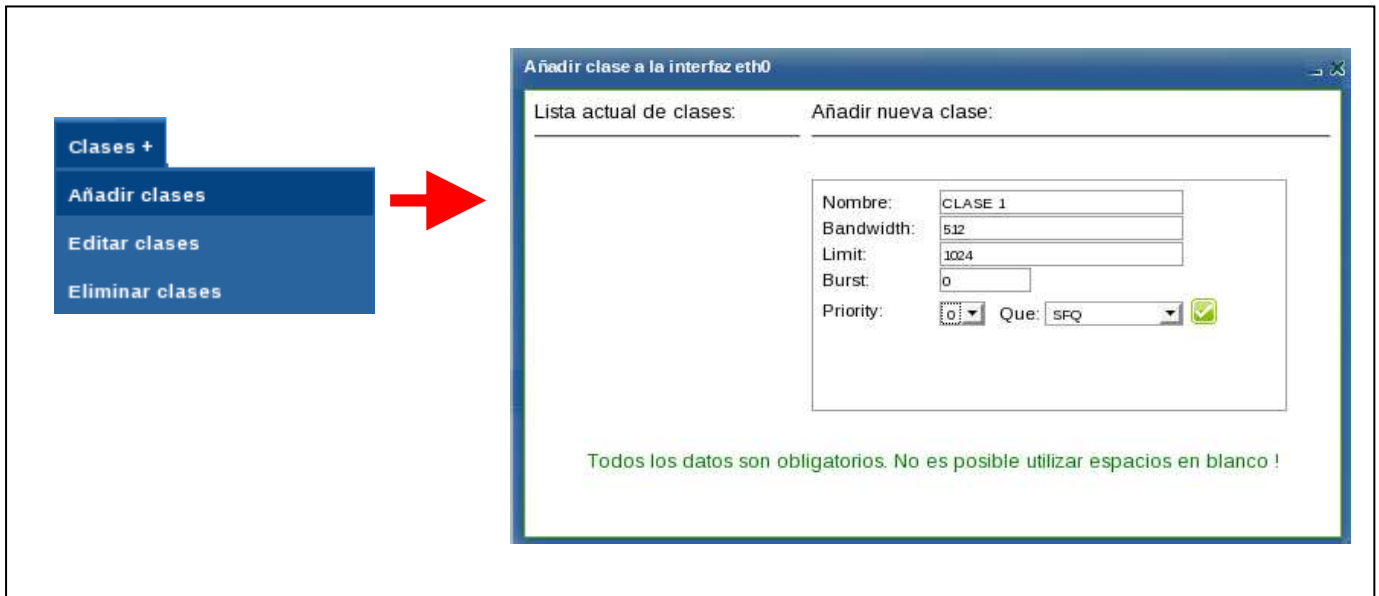


Figura E.2.- Secuencia para añadir una clase

E.3.- Añadir cliente

Una vez creada una clase, es posible asignarle clases hijas o clientes. Para crear un nuevo cliente hay que acceder a la pestaña **Añadir cliente** y elegir a que clase debe pertenecer. De la misma manera que para una clase madre, es necesario configurar unos parámetros obligatorios marcados en rojo (Figura E.3).

A cada cliente se le asigna un filtro con una serie de reglas. Es posible filtrar el tráfico a esta clase hija por todas las reglas o solo por algunas de ellas. Se puede filtrar según los siguientes parámetros:

- **SRC IPS:** Direcciones IP origen.
- **SRC PORTS:** Puertos origen. Si se especifican más de un puerto deben estar separados por comas.
- **DST IPS:** Direcciones IP destino
- **DST PORTS:** Puertos destino.
- **UPLOAD:** Filtrado del tráfico que sobrepase una determinada velocidad.
- **MARK:** Filtrado según la marca *iptables*.

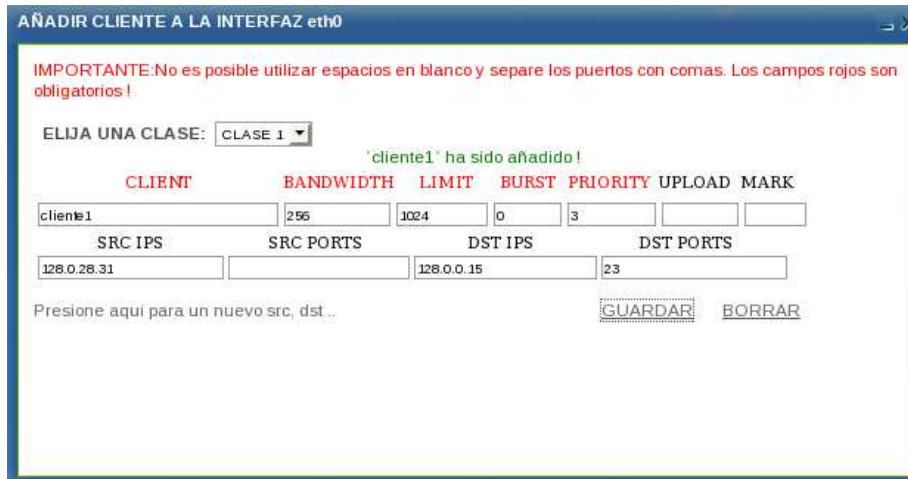


Figura E.3.- Ventana para añadir un cliente

E.4.- Aplicar cambios

Una vez se han creado todas las clases y todos los clientes necesarios, hay que activar el servicio *WebHTB* en la interfaz seleccionada o resetearlo si lo que se quiere es aplicar modificaciones. Para hacerlo hay que acceder a [Cambiar el estado QoS → Aplicar cambios](#).

Este comando crea el archivo `/tmp/qos.sh` en el temporal del servidor y lo ejecuta. En este *script* se encuentran convenientemente comentados todos los comandos para crear los diferentes elementos de tráfico HTB (Figura E.4).

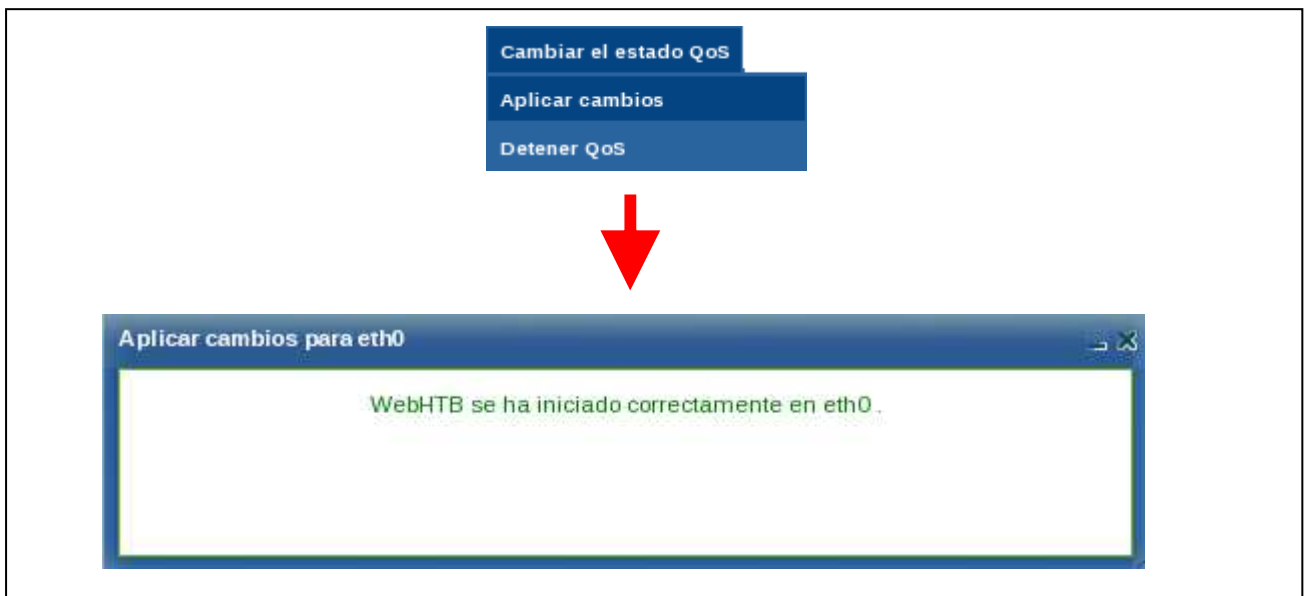


Figura E.4.- Secuencia para aplicar cambios

E.5.- Editar clases

Una vez se han creado las distintas clases, es posible editarlas para modificar alguno de sus parámetros. Para editar una clase hay que entrar en **Clases → Editar clases**. En esta pestaña aparece una ventana donde hay que seleccionar la clase a editar entre todas las comentadas anteriormente (Figura E.5). Para que los cambios en cualquiera de estas clases surjan efecto, hay que resetear el servicio como se ha comentado anteriormente.

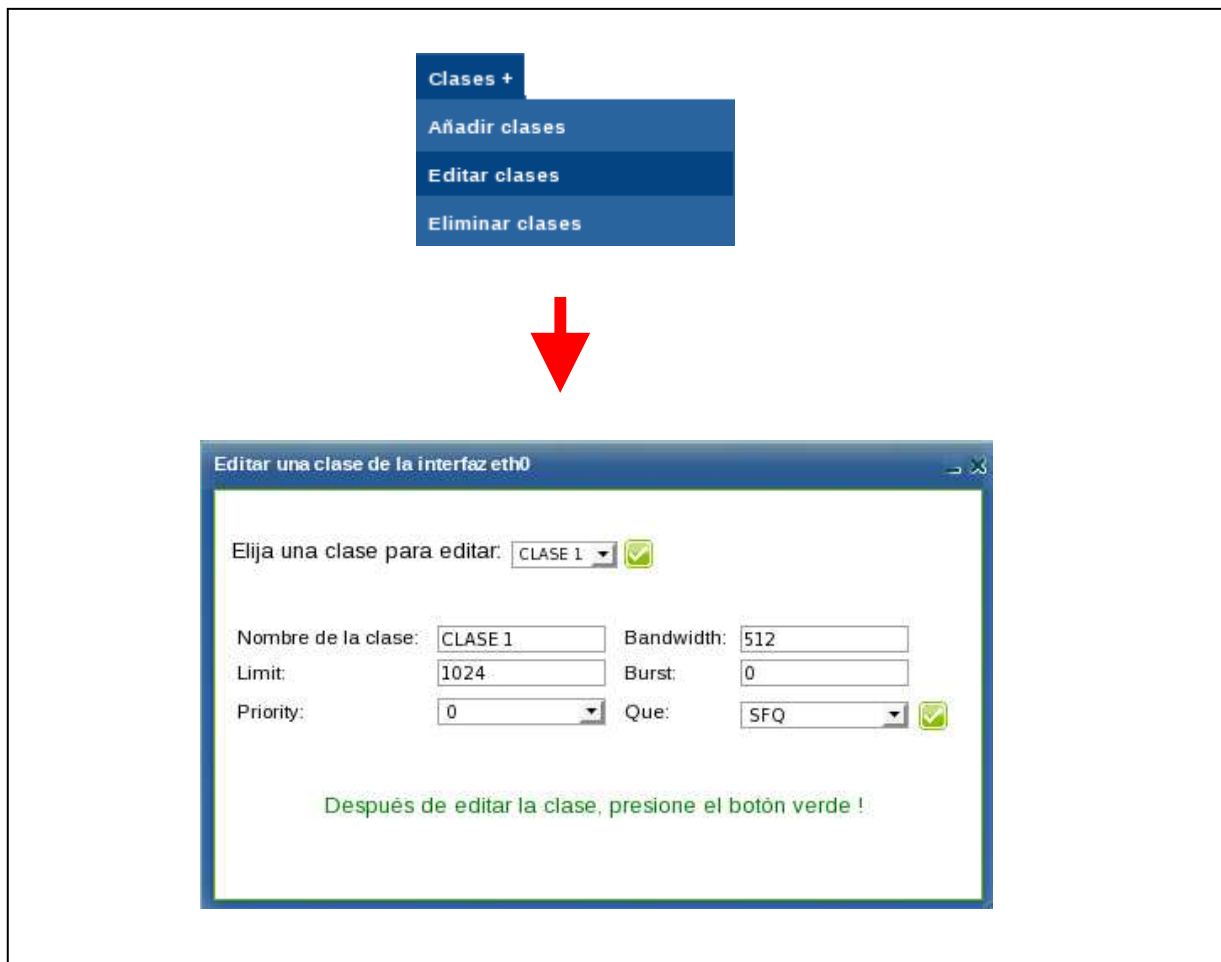


Figura E.5.- Secuencia para editar clases

E.6.- Configuración remota

WebHTB permite ser configurado remotamente. Para ello, hay que incluir la direcciones IP de los equipos remotos en la ventana **IP's permitidas**. No se requiere ningún tipo de autenticación para administrar el gestor de manera remota, tan solo tiene que estar incluida la dirección IP entre este grupo de administradores (Figura E.6).

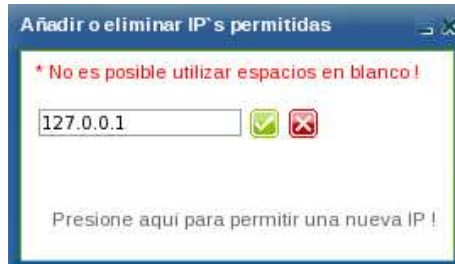


Figura E.6.- Ventana para habilitar direcciones IP

E.7.- Mostrar tráfico

Una de las ventajas de *WebHTB* respecto otras herramientas de control de ancho de banda es que permite ver el tráfico en tiempo real de las diferentes clases y clientes (Figura E.7). Este tráfico se muestra con un retraso de unos 5 segundos y se calcula en una ventana de 1 segundo. A partir de la versión v2.1, en esta ventana tan solo se muestran los clientes activos. Otra característica es que a parte del ancho de banda que ocupa cada clase (SPEED), también se puede ver el número de *tokens* y de *ctokens* para cada una de ellas. Esta opción se encuentra en [Mostrar → Mostrar tráfico](#).

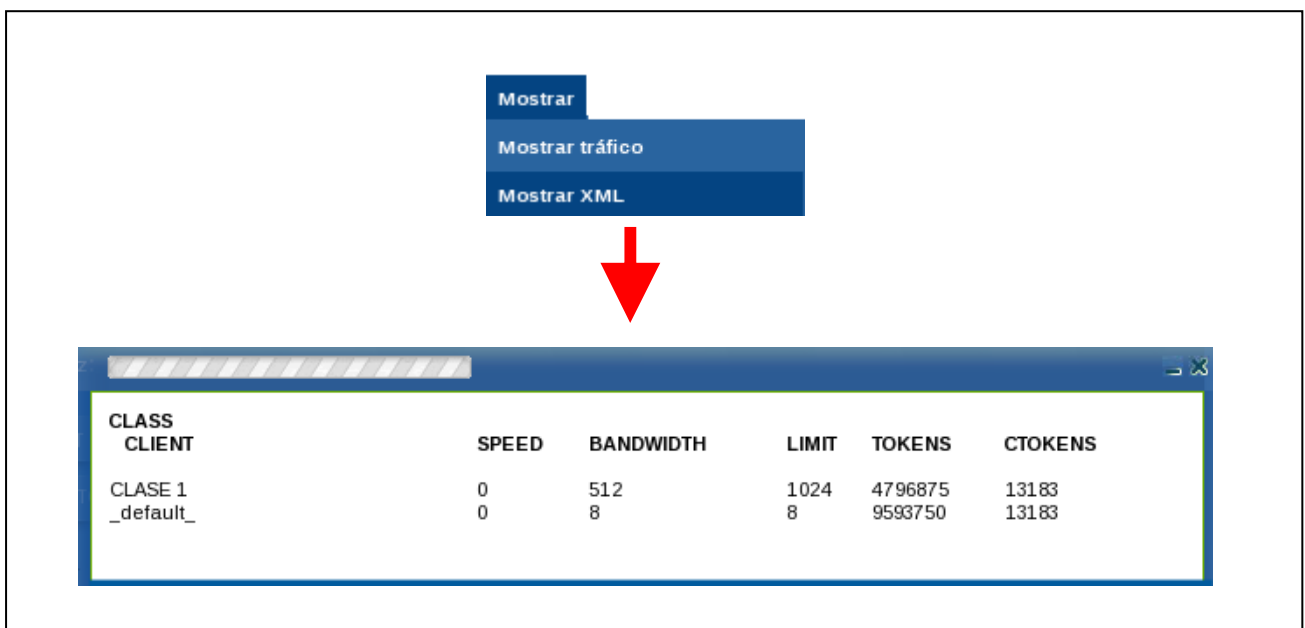


Figura E.7.- Secuencia para mostrar tráfico

E.8.- Detener el servicio

De la misma manera que se puede activar el servicio *WebHTB*, también es posible detenerlo. Esto se puede hacer desde la pestaña [Cambiar el estado QoS → Detener QoS](#).

Si el servicio se ha detenido correctamente en la interfaz seleccionada, aparecerá el mensaje de la figura E.8 por pantalla. Cabe decir que tanto para iniciar el servicio como para detenerlo hay que hacerlo para cada una de las interfaces activas.

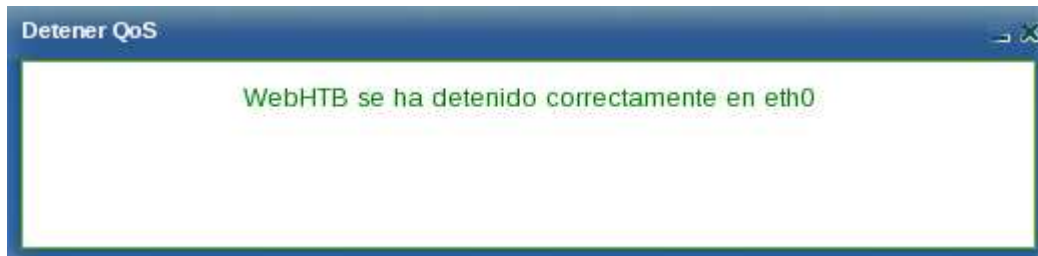


Figura E.8.- Ventana de parada del servicio

E.9.- Consejos para el uso de la aplicación

- Las unidades de los valores referentes a capacidad están expresadas en bits por segundo.
- Las unidades de los valores referentes a volumen de datos están expresadas en Bytes.
- Todos los valores de parámetros referentes a capacidades deben ser múltiplos de 8 bits/seg.
- Es obligatorio rellenar los campos marcados en rojo.
- No es posible asignar un mismo nombre a clientes diferentes, aunque sean de distintas clases madre o incluso de distintas interfaces.
- Los puertos utilizados como reglas de los filtros son tanto TCP como UDP.
- Para incluir más de un puerto en una regla hay que separarlos por comas.
- Para definir uno o varios puertos para cualquier rango IP, hay que utilizar la dirección IP genérica 0.0.0.0.
- Para ordenar los filtros, hay que hacerlo alfanuméricamente a partir del nombre del cliente.
- Para ordenar las clases, es posible hacerlo arrastrando las ventanas correspondientes con el ratón.

F.- Elementos de control de tráfico del gestor en producción

```
#!/bin/sh
DEV=eth1
TC=`which tc`
U32="filter add dev $DEV protocol ip parent 1:0 prio 1 u32"
U32u="filter add dev $DEV protocol ip parent ffff: prio 1 u32"

#Delete previous root qdisc

$TC qdisc del dev $DEV root >/dev/null 2>&1
$TC qdisc del dev $DEV ingress >/dev/null 2>&1

#Add root qdisc

$TC qdisc add dev $DEV handle ffff: ingress; >/dev/null 2>&1
$TC qdisc add dev $DEV root handle 1: htb default 10

#Add root class

$TC class add dev $DEV parent 1: classid 1:1 htb rate 1275kbps ceil 1275kbps
burst 3060k prio 0

#Add default class

$TC class add dev $DEV parent 1:1 classid 1:0x10 htb rate 8kbps ceil 1kbps
quantum 6000
$TC qdisc add dev $DEV parent 1:0x10 handle 0x10: pfifo limit 5

#Add class BARCELONA16

$TC class add dev $DEV parent 1:1 classid 1:0x20 htb rate 128kbps ceil 128kbps
burst 307k prio 0

#Add client 1.WinlabNet_bcn16

$TC class add dev $DEV parent 1:0x20 classid 1:0x21 htb rate 115kbps ceil
128kbps burst 307k prio 0
$TC ${U32} match ip src 128.0.0.23/32 match ip sport 1433 0xffff match ip dst
128.16.8.0/24 flowid 1:0x21
$TC ${U32} match ip src 128.0.0.23/32 match ip sport 1434 0xffff match ip dst
128.16.8.0/24 flowid 1:0x21
$TC ${U32} match ip src 128.0.0.23/32 match ip sport 2034 0xffff match ip dst
128.16.8.0/24 flowid 1:0x21
$TC ${U32} match ip src 128.0.0.23/32 match ip sport 3527 0xffff match ip dst
128.16.8.0/24 flowid 1:0x21
$TC ${U32} match ip src 128.0.0.24/32 match ip sport 1433 0xffff match ip dst
128.16.8.0/24 flowid 1:0x21
$TC ${U32} match ip src 128.0.0.24/32 match ip sport 1434 0xffff match ip dst
128.16.8.0/24 flowid 1:0x21
$TC ${U32} match ip src 128.0.0.24/32 match ip sport 2034 0xffff match ip dst
128.16.8.0/24 flowid 1:0x21
$TC ${U32} match ip src 128.0.0.24/32 match ip sport 3527 0xffff match ip dst
128.16.8.0/24 flowid 1:0x21
$TC ${U32} match ip src 128.0.0.25/32 match ip sport 3389 0xffff match ip dst
128.16.8.0/24 flowid 1:0x21
$TC qdisc add dev $DEV parent 1:0x21 handle 0x21: sfq
```

#Add client 2.Otros_bcn16

```
$TC class add dev $DEV parent 1:0x20 classid 1:0x22 htb rate 13kbps ceil 128kbps
burst 307k prio 3
$TC ${U32} match ip dst 128.16.8.0/24 flowid 1:0x22
$TC qdisc add dev $DEV parent 1:0x22 handle 0x22: sfq
```

#Add class MANRESA

```
$TC class add dev $DEV parent 1:1 classid 1:0x23 htb rate 128kbps ceil 128kbps
burst 307k prio 0
```

#Add client 1.WinlabNet_man

```
$TC class add dev $DEV parent 1:0x23 classid 1:0x24 htb rate 115kbps ceil
128kbps burst 307k prio 0
$TC ${U32} match ip src 128.0.0.23/32 match ip sport 1433 0xffff match ip dst
128.12.8.0/24 flowid 1:0x24
$TC ${U32} match ip src 128.0.0.23/32 match ip sport 1434 0xffff match ip dst
128.12.8.0/24 flowid 1:0x24
$TC ${U32} match ip src 128.0.0.23/32 match ip sport 2034 0xffff match ip dst
128.12.8.0/24 flowid 1:0x24
$TC ${U32} match ip src 128.0.0.23/32 match ip sport 3527 0xffff match ip dst
128.12.8.0/24 flowid 1:0x24
$TC ${U32} match ip src 128.0.0.24/32 match ip dst 128.12.8.0/24 flowid 1:0x24
$TC ${U32} match ip src 128.0.0.25/32 match ip dst 128.12.8.0/24 flowid 1:0x24
$TC qdisc add dev $DEV parent 1:0x24 handle 0x24: sfq
```

#Add client 2.Otros_man

```
$TC class add dev $DEV parent 1:0x23 classid 1:0x25 htb rate 13kbps ceil 128kbps
burst 307k prio 3
$TC ${U32} match ip dst 128.12.8.0/24 flowid 1:0x25
$TC qdisc add dev $DEV parent 1:0x25 handle 0x25: sfq
```

#Add class BARCELONA0

```
$TC class add dev $DEV parent 1:1 classid 1:0x26 htb rate 128kbps ceil 128kbps
burst 307k prio 0
```

#Add client 1.ERP_bcn0

```
$TC class add dev $DEV parent 1:0x26 classid 1:0x27 htb rate 115kbps ceil
128kbps burst 307k prio 0
$TC ${U32} match ip src 128.254.254.20/32 match ip dst 128.0.8.0/24 flowid
1:0x27
$TC ${U32} match ip src 128.254.254.21/32 match ip dst 128.0.8.0/24 flowid
1:0x27
$TC qdisc add dev $DEV parent 1:0x27 handle 0x27: sfq
```

#Add client 2.Otros_bcn0

```
$TC class add dev $DEV parent 1:0x26 classid 1:0x28 htb rate 13kbps ceil 128kbps
burst 307k prio 3
$TC ${U32} match ip dst 128.0.8.0/24 flowid 1:0x28
$TC qdisc add dev $DEV parent 1:0x28 handle 0x28: sfq
```

#Add class MALAGA1

```
$TC class add dev $DEV parent 1:1 classid 1:0x29 htb rate 128kbps ceil 128kbps
burst 307k prio 0
```

#Add client 1.WinLabNet_mall

```
$TC class add dev $DEV parent 1:0x29 classid 1:0x2a htb rate 115kbps ceil
128kbps burst 307k prio 0
$TC ${U32} match ip src 128.0.0.23/32 match ip sport 1433 0xffff match ip dst
128.1.29.0/24 flowid 1:0x2a
$TC ${U32} match ip src 128.0.0.23/32 match ip sport 1434 0xffff match ip dst
128.1.29.0/24 flowid 1:0x2a
$TC ${U32} match ip src 128.0.0.23/32 match ip sport 2034 0xffff match ip dst
128.1.29.0/24 flowid 1:0x2a
$TC ${U32} match ip src 128.0.0.23/32 match ip sport 3527 0xffff match ip dst
128.1.29.0/24 flowid 1:0x2a
$TC ${U32} match ip src 128.0.0.24/32 match ip sport 1433 0xffff match ip dst
128.1.29.0/24 flowid 1:0x2a
$TC ${U32} match ip src 128.0.0.24/32 match ip sport 1434 0xffff match ip dst
128.1.29.0/24 flowid 1:0x2a
$TC ${U32} match ip src 128.0.0.24/32 match ip sport 2034 0xffff match ip dst
128.1.29.0/24 flowid 1:0x2a
$TC ${U32} match ip src 128.0.0.24/32 match ip sport 3527 0xffff match ip dst
128.1.29.0/24 flowid 1:0x2a
$TC ${U32} match ip src 128.0.0.25/32 match ip sport 3389 0xffff match ip dst
128.1.29.0/24 flowid 1:0x2a
$TC qdisc add dev $DEV parent 1:0x2a handle 0x2a: sfq
```

#Add client 2.Otros_mall

```
$TC class add dev $DEV parent 1:0x29 classid 1:0x2b htb rate 13kbps ceil 128kbps
burst 307k prio 3
$TC ${U32} match ip dst 128.1.29.0/24 flowid 1:0x2b
$TC qdisc add dev $DEV parent 1:0x2b handle 0x2b: sfq
```

```
echo '9'
```

G.- Formación para el personal de General Lab

1.- INTRODUCCIÓN

1.1.- Presentación del proyecto

2.- INFORMACIÓN SOBRE LA RED

- 2.1.- Vista del mapa general de General Lab
- 2.2.- Ubicación física del gestor
- 2.4.- Configuración de red del gestor
- 2.5.- Modificaciones en los dispositivos de res
- 2.6.- Deshacer modificaciones

3.- INFORMACIÓN SOBRE LA APLICACIÓN

- 3.1.- Información sobre la máquina
- 3.2.- Información sobre *WebHTB*
- 3.3.- Funcionamiento del gestor
- 3.4.- Ejemplo
- 3.5.- Anotaciones
- 3.6.- Configuración en producción

4.- MANTENIMIENTO

- 4.1.- Servicios implicados
- 4.2.- Logs del sistema
- 4.3.- Herramienta monitorización *cacti*
- 4.4.- Herramienta de administración *webmin*

5.- PREGUNTAS Y COMENTARIOS

H.- Bibliografía y referencias

- [1] **Linux network traffic control - implementation overview**,
Werner Almesberger, Abril 1999,
<http://www.almesberger.net/cv/papers/tcio8.pdf>
- [2] **Linux Advanced Routing & Traffic Control**,
Bert Hubert, Octubre 2003,
<http://lartc.org/>
- [3] **HTB Linux queuing discipline manual - user guide**,
Martin Devera, Diciembre 2003,
<http://luxik.cdi.cz/~devik/qos/htb/>
- [4] **Traffic control HOWTO**,
Martin A. Brown, 2006,
<http://linux-ip.net/articles/Traffic-Control-HOWTO/>
- [5] **IP Command reference**,
Alexey N. Kuznetsov, Abril 1999,
<http://linux-ip.net/gl/ip-cref/>
- [6] **Differentiated Service on Linux HOWTO**,
Leonardo Balliache, Agosto 2003,
<http://www.opalsoft.net/qos/DS.htm>
- [7] **SELinux Quick Start Guide**,
<http://www.engardelinux.org/doc/guides/selinux-quick-start-guide/selinux-quick-start-guide/>
- [8] **Información sobre SELinux para Centos**,
<http://wiki.centos.org/HowTos/SELinux>
- [9] **Información sobre el mecanismo New API**,
<http://www.linuxfoundation.org/en/Net:NAPI>
- [10] **Quality of Service Networking**,
<http://www.cisco.com/en/US/docs/internetworking/technology/handbook/QoS.pdf>
- [11] **Linux - Advanced Networking Overview**,
Saravanan Radhakrishnan, Setiembre 1999,
<http://qos.itc.ku.edu/howto/>
- [12] **Linux 2.2 Packet Shaping HOWTO**,
Martijn van Oosterhout, Marzo 2000,
<http://svana.org/kleptog/Packet-Shaping-HOWTO.txt>
- [13] **Specification of the Controlled-Load Network Element Service**,
RFC 2211, Septiembre 1997,
<http://www.ietf.org/rfc/rfc2211.txt>

- [14] **Specification of Guaranteed Quality of Service**,
RFC 2212, Septiembre 1997,
<http://www.ietf.org/rfc/rfc2212.txt>
- [15] **Resource ReSerVation Protocol (RSVP)**,
RFC 2205, Septiembre 1997,
<http://www.ietf.org/rfc/rfc2205.txt>
- [16] **An Architecture for Differentiated Services**,
RFC 2475, Diciembre 1998,
<http://www.ietf.org/rfc/rfc2475.txt>
- [17] **Assured Forwarding PHB Group**,
RFC 2597, Junio 1999,
<http://www.ietf.org/rfc/rfc2597.txt>
- [18] **An Expedited Forwarding PHB**,
RFC 2598, Junio 1999,
<http://www.ietf.org/rfc/rfc2598.txt>
- [19] **An Architecture for Differentiated Services**,
RFC 2475, Diciembre 1998,
<http://www.ietf.org/rfc/rfc2475.txt>
- [20] **Multiprotocol Label Switching Architecture**,
RFC 3031, Enero 2001,
<http://www.ietf.org/rfc/rfc3031.txt>
- [21] **TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms**,
RFC 2001, Junio 1997,
<http://www.ietf.org/rfc/rfc2001.txt>
- [22] **Integrated Services in the Internet Architecture: an Overview**,
RFC 1633, Junio 1994,
<http://www.ietf.org/rfc/rfc1633.txt>
- [23] **Type of Service in the Internet Protocol Suite**,
RFC 1349, Julio 1992,
<http://www.ietf.org/rfc/rfc1349.txt>
- [24] **Enlace del proyecto Netfilter**,
<http://www.netfilter.org/>
- [25] **Enlace oficial de la compañía Eduna**,
<http://www.e-duna.com/>
- [26] **Enlace oficial de Softperfect Bandwidth Manager de la compañía Softperfect**,
<http://www.softperfect.com/products/bandwidth/>
- [27] **Enlace oficial de la solución Packetshaper de la compañía Packeteer**,
<http://www.bluecoat.com/products/packetshaper/>

- [28] **Información y descarga del proyecto HTB.init,**
<http://sourceforge.net/projects/htbinit>
- [29] **Enlace oficial del proyecto HTB-tools,**
<http://htb-tools.skydevel.ro/>
- [30] **Información y descarga del módulo HTB para webmin,**
<http://www.sehier.fr/webmin-htb/>
- [31] **Enlace oficial del proyecto WebHTB,**
<http://webhtb.sourceforge.net/>
- [32] **Ethloop - local simulator for testing Linux QoS disciplines,**
<http://luxik.cdi.cz/~devik/qos/ethloop/>
- [33] **Enlace oficial de la distribución Centos,**
<http://www.centos.org/>
- [34] **Repositorio para la descarga de kernels,**
<http://www.kernel.org/>
- [35] **Enlace oficial del proyecto webmin,**
<http://www.webmin.com/>
- [36] **Enlace oficial del proyecto wireshark,**
<http://www.wireshark.org/>
- [37] **Enlace oficial del proyecto cacti,**
<http://www.cacti.net/>
- [38] **Enlace oficial del gestor de bases de datos MySQL,**
<http://www.mysql.com/>
- [39] **Enlace oficial del proyecto Apache httpd Server,**
<http://httpd.apache.org/>
- [40] **Enlace oficial del proyecto Net-SNMP,**
<http://www.net-snmp.org/>
- [41] **Información sobre las distintas distribuciones y proyectos GNU/Linux,**
<http://distrowatch.com/>
- [42] **Enlace de la empresa General Lab,**
<http://www.general-lab.com/>
- [43] **Enlace del grupo Labco,**
<http://www.labco.com/>

I.- Índice de figuras

Figura 1.1.- Evolución de la actividad de la empresa en peticiones [42]	1
Figura 1.2.- Evolución de la facturación del grupo Labco [43]	2
Figura 1.3.- Esquema general de comunicaciones de General Lab.	6
Figura 1.4.- Diagrama de Gantt de planificación del proyecto	9
Figura 2.1.- Cabecera MPLS	15
Figura 3.1.- Procesamiento de paquetes en el <i>kernel</i> .	17
Figura 3.2.- Disposición de los elementos de control de tráfico en Linux [6]	19
Figura 3.3.- Relación entre elementos de control de tráfico	21
Figura 3.4.- Esquema FIFO [4]	22
Figura 3.5.- Esquema PFIFO_fast [4]	22
Figura 3.6.- Significado de algunos valores del campo ToS [23]	22
Figura 3.7.- Esquema SFQ [4]	23
Figura 3.8.- Esquema TBF [4]	23
Figura 3.9.- Esquema PRIO [6]	26
Figura 3.10.- Esquema DSMARK [6]	27
Figura 3.11.- Árbol de preferencias para HTB [6]	28
Figura 4.1.- Ejemplo de configuración de HTB.init	41
Figura 4.2.- Ejemplo del <i>script eth1-qos.sh</i>	42
Figura 4.3.- Tabla comparativa de herramientas para gestión de tráfico	45
Figura 6.1.- Copia parcial del fichero <i>/etc/httpd/conf/httpd.conf</i>	51
Figura 6.2.- Árbol de preferencias simple de ejemplo	51
Figura 6.3.- Ejemplo del fichero <i>qos.sh</i>	52
Figura 6.4.- Configuración de los parámetros de las clases de ejemplo	54
Figura 6.5.- Configuración de los parámetros de las disciplinas de colas de ejemplo	54
Figura 6.6.- Ventana para visualizar el tráfico en <i>WebHTB</i>	55
Figura 6.7.- Ejemplo de filtrado utilizando <i>mark</i>	57
Figura 7.1.- Red de pruebas básica sobre Ethernet	58
Figura 7.2.- Esquema del árbol de preferencias de pruebas sobre Ethernet	59
Figura 7.3.- Configuración de los filtros de ejemplo	60
Figura 7.4.- Gráfico de prueba entre clientes sobre Ethernet	62
Figura 7.5.- Gráfico de prueba entre servicios sobre Ethernet	63
Figura 7.6.- Red de prueba con gestión en origen sobre ADSL	64
Figura 7.7.- Red de pruebas con diversos enlaces	65

Figura 7.8.- Contenido del <i>script</i> para hacer un NAT <i>nat.sh</i>	66
Figura 7.9.- Contenido del <i>script qos_start.sh</i>	66
Figura 7.10.- Tabla de relación entre enlaces.....	67
Figura 7.11.- Gráfico de relación entre enlaces	68
Figura 7.12.- Red de prueba en destino sobre ADSL.....	69
Figura 7.13.- Gráfico de mecanismos de control de congestión.....	70
Figura 7.14.- Gráfico de pruebas entre servicios con gestión en destino.....	71
Figura 7.15.- Gráfico de comparación de colas con protocolo UDP	73
Figura 7.16.- Gráfico de comparación de colas con protocolo TCP.....	74
Figura 8.1.- Tabla de servicios prioritarios	76
Figura 8.2.- Esquema de ubicación del gestor en producción	77
Figura 8.3.- Tabla de rutas en producción.....	79
Figura 8.4.- Comandos para enrutar el tráfico en el <i>firewall</i>	79
Figura 8.5.- Árbol de preferencias en producción	80
Figura 8.6.- Ventana de creación de elementos en <i>WebHTB</i>	81
Figura 8.7.- Configuración de los filtros en producción.....	82
Figura 8.8.- Parámetros de las clases en producción	83
Figura 8.9.- Parámetros de las disciplinas de colas finales en producción	83
Figura 8.10.- Gráfico de resultados en producción.....	84
Figura 8.11.- Ventana de tráfico en tiempo real para la prueba en producción.....	85
Figura 9.1.- Tabla de coste del proyecto	87
Figura A.1.- Distribución de las particiones del sistema.....	96
Figura B.1.- Copia parcial del fichero <i>/etc/sudoers</i>	103
Figura B.2.- Copia del fichero <i>.bash_profile</i> del usuario <i>informatica</i>	104
Figura B.3.- Estado de algunos servicios según el <i>runlevel</i>	104
Figura B.4.- Copia parcial del fichero <i>/etc/sysctl.conf</i>	105
Figura B.5.- Copia parcial del fichero <i>/etc/sysconfig/selinux</i>	106
Figura B.6.- Ejemplo 1 del fichero <i>route-eth0</i>	107
Figura B.7.- Ejemplo 2 de fichero <i>route-eth0</i>	107
Figura B.8.- Copia parcial del asistente de instalación de <i>webmin</i>	108
Figura B.9.- Captura de monitorización del tráfico con <i>cacti</i>	110
Figura C.1.- Salida del comando <i>top</i>	111
Figura C.2.- Salida del comando <i>df</i>	112
Figura C.3.- Logs del sistema	112
Figura D.1.- Contenido de la carpeta <i>WebHTB v2.1</i>	115

Índice de figuras

Figura D.2.- Pantalla de configuración de <i>WebHTB</i>	116
Figura D.3.- Copia parcial del fichero <i>sudoers</i>	117
Figura D.4.- Fichero de prueba <i>test.php</i>	117
Figura D.5.- <i>Script</i> de prueba <i>test2.php</i>	118
Figura D.6.- Error de permisos de <i>sudoers</i>	118
Figura D.7.- Error del terminal TTY	119
Figura D.8.- Copia parcial del fichero <i>/etc/sudoers</i>	119
Figura E.1.- Secuencia para añadir una interfaz	120
Figura E.2.- Secuencia para añadir una clase.....	122
Figura E.3.- Ventana para añadir un cliente	123
Figura E.4.- Secuencia para aplicar cambios	123
Figura E.5.- Secuencia para editar clases.....	124
Figura E.6.- Ventana para habilitar direcciones IP.....	125
Figura E.7.- Secuencia para mostrar tráfico.....	125
Figura E.8.- Ventana de parada del servicio.....	126

