Nacido en Madrid en junio de 1978, realizó sus estudios elementales en el colegio ‘Fray Luis de León’ de Madrid. Posteriormente cursó Ingeniería de Telecomunicaciones en la universidad Carlos III de Madrid acabando dicha titulación en 2002. Paralelamente a la realización del proyecto fin de carrera, consiguió una beca de colaboración con el departamento de ingeniería telemática en donde trabajó dos años en el proyecto europeo “MobyDick” sobre redes móviles de 4ª generación. Dentro de este proyecto se centró en la parte de gestión e implementación de la calidad de servicio en la red, desplegando un entorno completo y publicando varios artículos. Así mismo realizó un curso de administración de sistemas operativos en redes. En la actualidad se encuentra trabajando en una empresa de electrónica desarrollando y comerciando aplicaciones firmware.

Gestión dinámica de políticas bajo entorno DiffServ

Pedro Antonio Vico Solano

RESUMEN

1. INTRODUCCIÓN

1.1 QoS en redes IP

Existen dos métodos principales para la provisión de calidad de servicio en redes IP. El primero de ellos es el modelo de servicios integrados: **IntServ**.

IntServ asegura QoS extremo a extremo a nivel de flujo de datos. Esto implica tener que realizar reservas a lo largo de todo el camino que sigan los paquetes para cada flujo de comunicación con un protocolo específico: el RSVP (ReSerVation Protocol). Es necesario también que los routers implicados en estas comunicaciones deban guardar estado de todos los flujos. Además, esta reserva es soft-state, lo que significa que las reservas se deben actualizar cada cierto tiempo por lo que la carga de señalización de este modelo es muy significativa. En este sentido, el modelo IntServ presenta importantes problemas de escalabilidad.

Por otro lado está el modelo **DiffServ**.

Se dice que el modelo DiffServ (servicios diferenciados) proporciona CoS (clases de servicio) a agregados de tráficos unidireccionales. En un *dominio* DiffServ diferentes tráficos de salida de usuarios se *asocian* hacia una misma *clase* DiffServ. Estas constituyen, por lo tanto, un agregado de tráfico que se identifica con un DiffServ Code Point (DSCP) determinado. En el caso de paquetes IPv6 este campo de 6 bits se encuentra dentro del campo TClass (Traffic Class) de su cabecera.

Para evitar un uso indebido de la red así como administrar los recursos utilizados y los usuarios, estas redes necesitan la incorporación de un servidor de recursos (o Bandwidth Broker) por cada dominio DiffServ.

1.2 DiffServ y control de tráfico en Linux

Una vez elegido DiffServ como modelo apropiado de QoS, pasamos a ver el soporte que ofrece para éste el sistema operativo Linux.

Para manejar mecanismos de Calidad de Servicio los nuevos kernels de Linux ofrecen el llamado control de tráfico o TC. Este soporte permite básicamente dos tipos de operaciones:

- Hacer peticiones al kernel para que instale diferentes algoritmos de gestión y planificación de colas que nos permitan clasificar, conformar y planificar los paquetes de salida en una interfaz dada.
- Obtener del kernel información sobre la correcta creación de los anteriores métodos y pedir a éste las estadísticas de uso de ellos. Esta última función nos va a permitir cerrar el lazo de administración en los métodos de calidad.

Cabe destacar que la comunicación entre el espacio de usuario y el kernel se realiza mediante unos *sockets* y mensajes *netlink*, parecidos a los *inet.*

Si nos centramos en los métodos de calidad de servicio del TC, veremos que debemos crear un árbol de calidad de servicio compuesto por:

- Disciplinas de cola (en adelante *qdiscs*): Determinan la manera en la que los paquetes serán reenviados por un máquina Linux. La política por defecto es FIFO, pero existen otras basadas en prioridades, Round Robin, RED,…
- Las clases (hijas de las disciplinas) permiten realizar separaciones del tráfico para, por ejemplo, aplicar una política diferente a cada clase.
- Filtros o clasificadores: cuya función es asignar el tráfico a las distintas clases en función del criterio de selección que se especifique.

Finalmente, existen dos herramientas que nos van a ahorrar el uso de los complicados mensajes *netlink*:

- La primera es un programa llamado '`tc`'. Se utiliza por línea de comandos en modo script.
- La segunda, y más adecuada para interactuar con programas, es la librería TC API de IBM. Son funciones en C que nos permiten hacer todo lo que permite el *tc*

1.3 El protocolo COPS

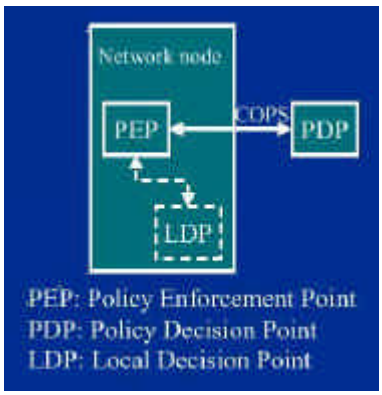
El protocolo COPS (Common Open Policy Service) descrito en la RFC 2748, define un modelo cliente/servidor sencillo para proporcionar control de políticas a protocolos de señalización de calidad de servicio.

El modelo no hace ninguna suposición acerca de los métodos utilizados en el servidor de políticas, sino que se basa en un **servidor (**llamado **PDP)** que devuelve decisiones a las peticiones realizadas por los **clientes (**llamados **PEP)**. Es el protocolo más apropiado para un intercambio de información entre un servidor de políticas y diversos clientes que instalan perfiles de usuario o aplican métodos de calidad de servicio bajo orden del servidor. El PEP también tiene la capacidad de informar al PDP si ha podido instalar localmente con éxito la decisión recibida, es también el PEP el encargado de borrar cualquier estado que ya no sea válido debido a eventos en el propio cliente o a decisiones enviadas por el servidor. Utiliza TCP como protocolo de transporte (puerto 3288) para asegurar así fiabilidad en el intercambio de mensajes entre los clientes y el servidor.

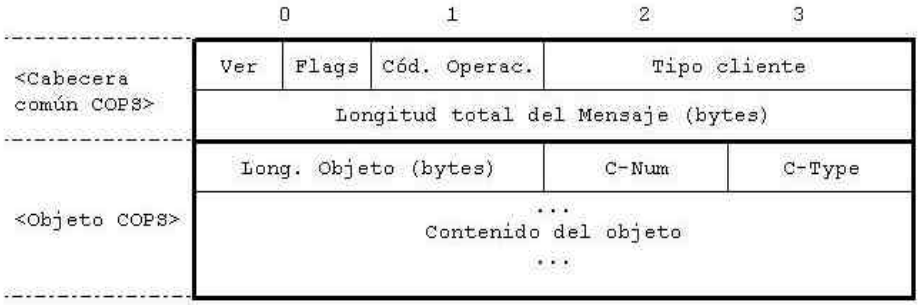
El protocolo necesita información de estados en las máquinas ya que: (1º) El estado de una petición/decisión es compartido entre el cliente y el servidor y (2º) el estado de varios eventos (pares petición/decisión) puede estar interrelacionado.

La definición del protocolo es bastante abierta para que sea extensible y pueda soportar los distintos tipos de clientes que pudieran aparecer en un futuro.

En la figura de la derecha vemos el modelo del protocolo COPS. Se utiliza para comunicar información sobre políticas de red entre los puntos de aplicación de políticas (PEPs) y un servidor de políticas remoto (PDP). Dentro del nodo de red puede existir un PDP local (LPD o LPDP) que puede ser utilizado para tomar decisiones locales en ausencia de un PDP.



Cada mensaje COPS consta de una cabecera COPS y un conjunto de objetos COPS ya definidos. La cabecera son los 8 primeros bytes y en ella podemos ver, entre otras cosas, el código de operación y el tipo de cliente que manda el mensaje. Aparte, un mensaje se compone de N objetos COPS con el mismo formato y alineados a 32 bits.



2. OBJETIVOS DEL PROYECTO

El objetivo principal de este proyecto consiste en implantar diferentes métodos de calidad de servicio en una red DiffServ así como desarrollar mecanismos que permitan su gestión y configuración. La función que debe realizar el conjunto es la de proporcionar determinada calidad de servicio a un tráfico que previamente la haya reservado. La configuración de las colas y clases de calidad de servicio, la gestión de los recursos de red así como la gestión de usuarios, estarán completamente integradas en nuestro servidor de políticas llamado Bandwidth Broker. Los demás elementos del entorno se limitarán a hacerle peticiones y aplicar sus respuestas.

Para la consecución de este objetivo se siguieron los siguientes pasos: se migró un entorno de marcado para VoIP existente en el departamento entorno al protocolo IPv6, después se le agregaron capacidades de control de tráfico a los routers de acceso mediante la

librería de control de tráfico TC API y seguidamente se modificó el entorno para adaptarlo a los requerimientos de calidad de servicio del proyecto MobyDick.

El proyecto MobyDick (**Mobility and Differentiated Services in a Future IP Network)** tiene como propósito el diseño, implementación y pruebas de una arquitectura de red móvil de cuarta generación en la que existan aspectos de AAAC (autenticación, autorización, contabilidad y facturación), calidad de servicio (QoS) y movilidad (MIP) todo ello sobre IPv6 y a través de diferentes medios de acceso (Ethernet, WLAN, WCDMA).

3. ENTORNO DE QoS

El entorno de aplicación y gestión de calidad de servicio que pasaremos a describir a continuación consta de los siguientes elementos:

Bandwidth Broker (BB)

- Controla el acceso a los servicios de la red por parte de los usuarios. Él es el encargado de hacer cumplir el SLA (Sevice Level Agreement) contratado. Partiendo de los requerimientos que tenga el tráfico de un usuario (BW, prioridad, retardo,...) escoge el DSCP más adecuado y se lo comunica (pasando por el AAAC) al usuario final para que éste marque el tráfico con ese identificador.
- Configura los dispositivos de aplicación de mecanismos de QoS => Routers de acceso. Es el servidor de políticas (PDP).
- Gestiona el uso de la red. Mediante los informes de uso del ancho de banda que le envían periódicamente los Routers y conociendo los recurso de la red y los usuarios registrados, su algoritmo de control adaptativo permite reestructurar todos los flujos de usuarios mediante el control de las colas en los routers.

Router de acceso (RA)

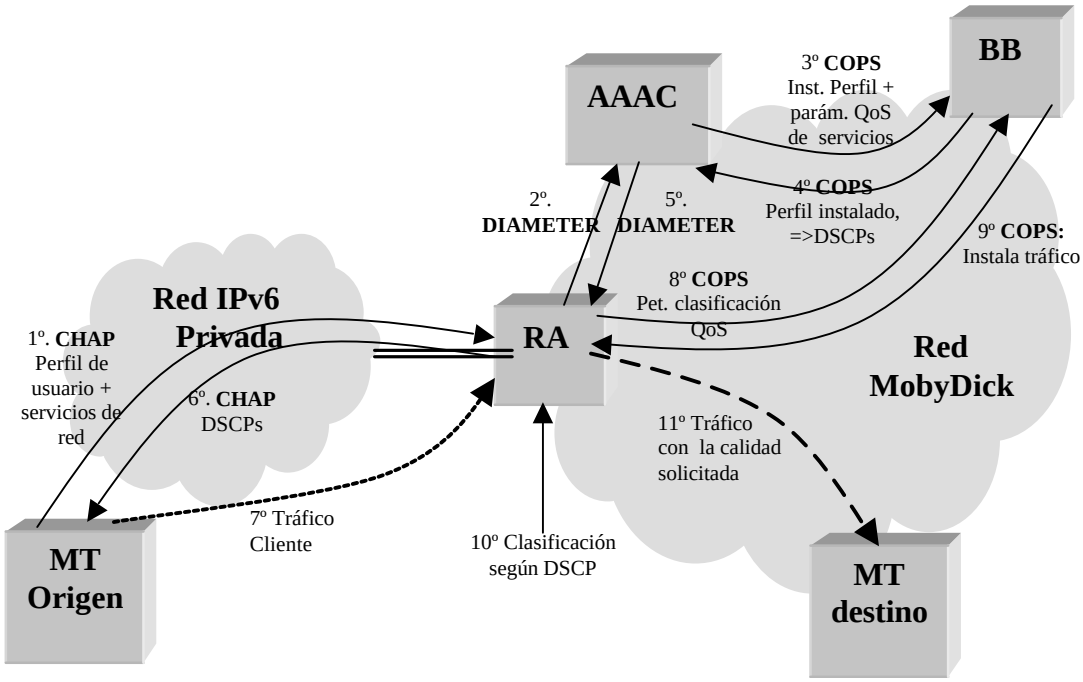
- Gestiona y planifica las colas de QoS configuradas por el BB. Es el cliente de políticas (PEP).
- Captura información de los tráficos que lo atraviesan para luego preguntar por ellos al BB.
- Monitoriza las colas de QoS para informar al BB de su uso.
- Clasifica los tráficos salientes hacia las colas según el DSCP del tráfico y bajo orden del BB.

AAAC QoS (Autenticación, Autorización, Contabilidad y Facturación)

- Obtiene el perfil contratado de un usuario de la base de datos de usuarios autorizados, trata de instalarlo en el BB y obtiene los DSCPs de sus servicios de red asociados. Es otro tipo de cliente de políticas (PEP).
- Actualiza o borra dichos perfiles según los permisos de éste.

4. FUNCIONAMIENTO

En la siguiente figura podemos ver el proceso completo que se produce al registrase un cliente en la red y mandar tráfico.



- **1º**: Un cliente (terminal móvil) que desee registrarse en la red MobyDick, debe comunicarse con el AAAC para obtener los DSCPs con los que debe marcar su(s) tráfico(s) de salida según su perfil. El AAAC se encuentra dentro de la red DiffServ – al otro lado del router de acceso– por lo que debe establecer una serie de conexiones de señalización seguras para poder acceder a éste. Primeramente el usuario debe comunicarse con su Router de acceso mediante una conexión autenticada mediante el protocolo CHAP (CHAlenge Protocol) para registrarse en el dominio DiffServ MobyDick.
- **2º**: Una vez que la autentificación CHAP se ha realizado con éxito, el Router genera un nuevo mensaje de registro hacia el AAAC mediante el protocolo seguro DIAMETER. Este tráfico, al ser de señalización y tener la dirección origen del router, será clasificado a la clase EF (expedited forwarding), la más alta dentro de las calidades de servicio.
- **3º:** el AAAC consulta en sus tablas los permisos y tráficos que al usuario se le permite cursar y avisa al Bandwidth Broker a través del protocolo COPS pasándole toda la información relacionada con el usuario y sus tráficos: dirección origen, dirección de destino, tiempo de vida, ancho de banda, prioridad y máximo retardo permitido.
- **4º:** El Bandwidth Broker busca entonces en su tabla el DSCP más adecuado para cada servicio solicitado, añade estos a una tabla interna con todos los servicios activos y devuelve los DSCPs al AAAC para que éste se los comunique al cliente.
- **5º y 6º:** la información de los DSCPs con los que el cliente debe marcar sus diferentes tráficos viaja hasta él por el camino inverso que siguió el registro. Usando los protocolos de señalización anteriormente expuestos.

- **7º:** El cliente comienza entonces a enviar un tráfico hacia otro terminal. El router captura ese tráfico con su programa capturador y clasifica temporalmente los paquetes como 'por defecto' (clase best-effort), de esta forma se evita tener que retener los paquetes que causaría muchos problemas.
- **8º**: Mediante el protocolo COPS el Router de acceso de nuestra red pide al Bandwidth Broker si tiene o no que encaminar ese tráfico a la cola de QoS asociada a su DSCP. El Router le pasa al Bandwidth Broker la dirección origen y el DSCP del flujo de datos detectado.
- **9º:** El Bandwidth Broker comprueba entonces si ese tráfico está instalado en su tabla. Si el servicio estaba instalado, el Bandwidth Broker le envía al Router una confirmación positiva mediante la conexión COPS. Si no encuentra el tráfico en su tabla la respuesta es negativa y el tráfico seguirá clasificado como 'por defecto'.

La clasificación de QoS en el Router tiene un tiempo de vida. Cuando ese tiempo se agota, el router hace una nueva petición para saber si ese flujo de datos debe seguir siendo clasificado hacia la misma cola o hacia la 'por defecto'. De esta forma la asignación de calidad de servicio a los flujos de datos puede actualizarse cada cierto tiempo. También el AAAC actualiza periódicamente el perfil de usuario en el BB o lo borra según los permisos del MT (equipo móvil).

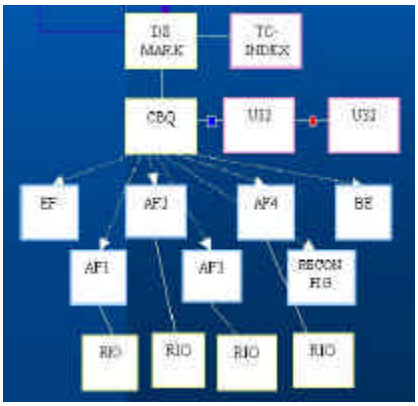
5. CONFIGURACIÓN Y ESTADÍSTICAS

Continuamos ahora describiendo los mecanismos de configuración y transferencia de estadísticas entre el Router y el BB.

Antes de que el entorno opere en su modo normal el router debe haber solicitado al BB una configuración de sus colas. Además durante todo el tiempo que dura su ejecución el router debe encargarse de realizar dos operaciones: obtener estadísticas del uso de las disciplinas de cola instaladas en su interfaz de salida y comprobar si en los mensajes que le manda el BB se le solicita una reconfiguración de sus colas. El intervalo de tiempo entre dos informes de estadísticas está también configurado por el Bandwidth Broker ya que éste se lo indica en el primer mensaje COPS al arrancar.

Para solicitar una (re)configuración se deben seguir los siguientes pasos:

- Mediante un mensaje COPS de "Configuration Request" el router solicita al BB que le mande la configuración de las colas (nuevas o primeras).
- A través de otro mensaje COPS de "Configuration Decision" el BB le manda al router dicha configuración. El mensaje lleva N objetos COPS, tantos como DSCPs se puedan filtrar por el router. Cada objeto lleva los siguientes parámetros: DSCP (con el que se clasifican los tráficos hacia una cola), nº de agregado (permiten unir varios DSCPs en una clase), ancho de banda, flag de uso de ancho de banda desocupado y parámetros de la cola RIO (Random Input-Output) si procede.
- Con la tabla recibida, el Router procede a construir



un árbol de calidad de servicio mediante las funciones de control de tráfico del TC API. Ese árbol tiene una forma como el de la figura de arriba a la derecha.

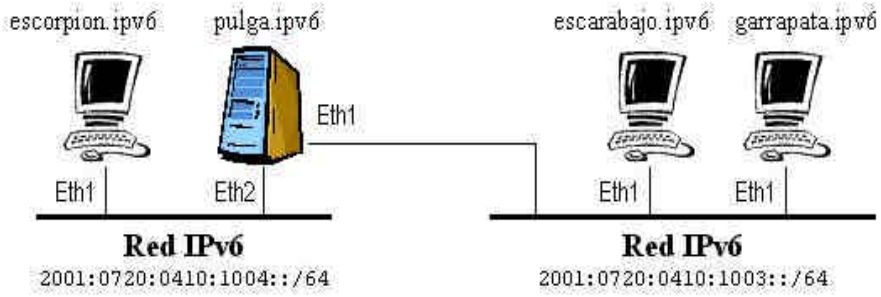
- Tanto si todo el proceso de construcción de las colas ha resultado satisfactorio como si no el RA debe informar al BB mediante un mensaje COPS de "Configuration Report". Si la respuesta es negativa se reintentará un número predefinido de veces antes de cerrarse la conexión COPS y el Router.

Además de este proceso, el Router periódicamente debe enviar un informe del estado de sus colas al BB. De este modo se cierra el lazo de control de políticas. Esta información va contenida en un mensaje COPS de "Accountig Report" con una línea por cada disciplina de colas creada. Los parámetros que se envían son: identificador de la disciplina, régimen binario, paquetes por segundo, descartes por segundo y sobrecargas por segundo promedio.

Las estadísticas que se envían al BB sirven para que éste conozca remotamente la ocupación del Router. Si el Bandwidth Broker necesitase cambiar la configuración de calidad de servicio del Router se lo comunicaría y éste procedería a repetir una secuencia de configuración de nuevo. Durante el corto tiempo que dura una reconfiguración el implementador puede especificar el ancho de banda que se le proporciona a los tráficos mediante un parámetro en el Router.

6. PRUEBAS

Para las pruebas utilizamos el siguiente escenario que es una porción del escenario completo de pruebas de la red MobyDick de la universidad. Se emplearon dos subredes Ethernet privadas de 100Mbps como muestra la siguiente figura:



En todas las máquinas se usó el sistema operativo Red Hat Linux con un kernel 2.4.16. La funcionalidad de las máquinas es la siguiente:

- Escorpion.ipv6: generador de diferentes tráficos IPv6 hacia 'garrapata.ipv6'. En las pruebas se ha utilizado tráfico unidireccional UDP sobre IPv6 (programa: mgen6).
- Pulga.ipv6: Router de acceso que encamina los paquetes de la red 2001:720:410:1004::/64 a la red 2001:720:410:1003::/64 y aplica las colas de QoS en su interfaz 'eth1'.
- Escarabajo.ipv6: en esta máquina residirá el AAAC.
- Garrapata.ipv6: máquina destino de los paquetes procedentes de 'escorpion.ipv6', en ella se obtendrán las estadísticas de los tráficos. Además aquí también se ejecutará el Bandwidth Broker.

Definición de las pruebas

Las pruebas que vamos a realizar consistirán en: verificar la correcta secuencia de los mensaje COPS y comprobar los parámetros de los tráficos cursados estáticamente y dinámicamente.

Generador de tráfico UDP: MGEN6

En el programa emisor es donde se especifican todos los datos necesarios para detallar el flujo tales como: el tamaño de los paquetes, el número de paquetes por segundo y el tiempo que dura la prueba. Los resultados obtenidos son, entre otras cosas, el régimen binario de los datos, los descartes, las colisiones, los paquetes recibidos, el retardo máximo, el jitter, etc.

Un inconveniente que tiene este generador de tráfico es que no nos permite marcar el DSCP del tráfico. Para remediar esto se desarrolló un sencillo programa de marcado (usando la disciplina DSMARK del TC API) que nos permitió marcar todo el tráfico de salida por una determinada interfaz con un DSCP que se nos solicitaba por pantalla.

Recordemos antes de comenzar las pruebas que, para que el router aplique un filtro a un tráfico, éste debe estar previamente registrado en el Bandwidth Broker. Para ello se debe instalar previamente el tráfico a probar mediante el programa que simula al AAAC. Debemos comprobar que los dos clientes (PEPs) se registran correctamente en el Bandwidth Broker y que, el Router de acceso recibe la tabla de QoS a aplicar. Si todo ha funcionado correctamente el router instala las clases de calidad de servicio en su interfaz de salida. Recordemos que DiffServ se especifica sólo para tráficos unidireccionales y, en este caso, de entrada a la red MobyDick.

6.1 Prueba de QoS estática

En este test se cursará por cada clase su régimen binario límite (un poco mayor que el máximo configurado en la cola correspondiente para que se produzcan algunos descartes). Al ser la prueba estática los archivos de configuración que nos transfiere el BB serán los mismos que éste carga en su inicialización.

Para esta prueba se utilizó el programa ‘mgen6’ con varias tasas binarias, una para cada clase AF (Assured Forwarding), EF y best-effort. En todas las clases se utilizaron paquetes de 1000 bytes de datos y las pruebas duraron 20 segundos; lo único que fue ajustado es el parámetro de ‘paquetes por segundo’ para que se cumpliesen las tasas binarias en los datos.

En las tablas siguientes presentamos los resultados y pasamos a continuación a explicarlos:

	BB cursado en datos	BB cursado en enlace
Best-effort	960 kbps	1019 kbps

	BB cursado en datos	BB cursado en enlace
EF	17333 kbps	18407 kbps

Tablas de resultados para Best-effort de 1000kbps y Expetited Forwarding de 20000 kbps

AF	Clase 1 (1,2Mbps)		Clase 2 (2,1Mbps)		Clase 3 (4,4Mbps)		Clase 4 (12Mbps)	
	BB cursado en datos	Descar-tes	BB cursado en datos	Descar-tes	BB cursado en datos	Descar-tes	BB cursado en datos	Descar-tes
Bajo porcentaje de descarte	1189 kbps	15 (0,5%)	2058 kbps	0 (0%)	4229 kbps	103 (0,9%)	10903 kbps	2161 (7,2%)
Porcentaje medio de descarte	1189 kbps	20 (0,6%)	2057 kbps	10 (0,2%)	4225 kbps	223 (2%)	10904 kbps	2303 (7,6%)
Alto porcentaje de descarte	1189 kbps	23 (0,7%)	2058 kbps	63 (1,1%)	4229 kbps	269 (2,4%)	10901 kbps	2421 (8%)

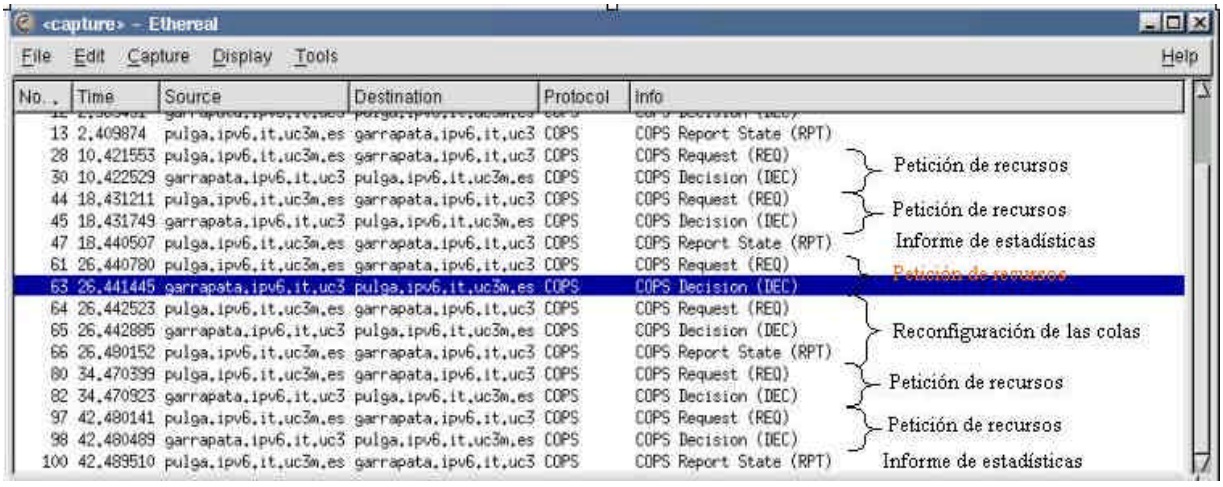
El árbol de QoS que el Bandwidth Broker ha configurado en el router consta de 4 clases AF, una EF y otra best-effort. Cada una de las cuatro clases AF tiene a su vez tres disciplinas de descarte distintas para tratar las ráfagas de paquetes.

En la primera fila de la tabla superior (clases AF) podemos ver el ancho de banda configurado a cada clase. Una vez autorizado e insertado tráfico por la interfaz podemos ver los anchos de banda que se consiguen cursar por cada clase. Cabe recordar que para que el router clasifique los tráficos a una cola determinada se basa en el DSCP de los paquetes, por lo tanto sólo para AF hay 12 DSCPs distintos. Se aprecia que el ancho de banda cursado se ajusta a lo configurado (mejor en valores bajos) y como los descartes aumentan en función de la sub-disciplina elegida debido a las ráfagas que el generador de tráfico provoca.

Para la clase Best-Effort se configuraron 1000 kbps y para la EF 20000 kbps. Podemos ver los resultados en las tablas pequeñas superiores.

6.2 Prueba de secuencia de mensajes COPS

A continuación mostraremos una secuencia típica de intercambio de información de políticas a través del protocolo COPS sobre IPv6. Para una explicación detalla de los mensajes y la secuencia consultar el proyecto adjunto.

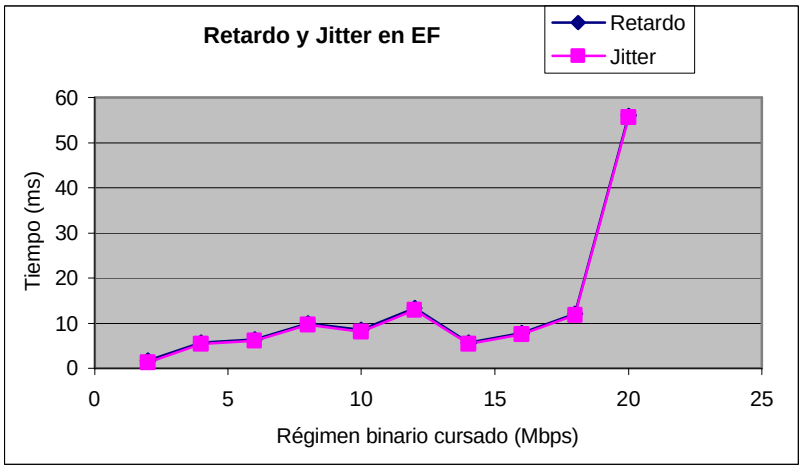


En esta secuencia entre el Router y el servidor podemos ver como, habiéndose capturado dos tráficos, el Router pide al BB si debe instalar los filtros que les proporcionen QoS (parejas de mensajes 'Resource Allocation REQuest' y 'DECision'). Después de esto y debido al valor del 'Accounting Timer' se manda un informe al Bandwidth Broker con las estadísticas de uso de la interfaz. El Bandwidth Broker realiza cálculos con esos datos y determina que es necesario una reconfiguración de las colas de QoS. Por esto, en la siguiente petición de asignación de recursos (comentada en naranja) el flag de reconfiguración estará activado (ver detalle de la captura en el proyecto).

Acto seguido, el router realiza una secuencia de reconfiguración intercambiándose los mensajes: Configuration Request, Decision y Report. Una vez reconfiguradas las colas se debe volver a pedir los recursos para los tráficos activos. Finalmente el Router vuelve a mandar estadísticas del uso de la interfaz. La conexión se cerraría con un 'Client-Close'.

6.3 Prueba retardo máximo y jitter

Para esta prueba reservamos 20Mbps para la clase EF y fuimos incrementando el tráfico cursado por ella de 2Mbps en 2Mbps hasta su valor máximo. Los datos obtenidos y su gráfica correspondiente se detallan a continuación:



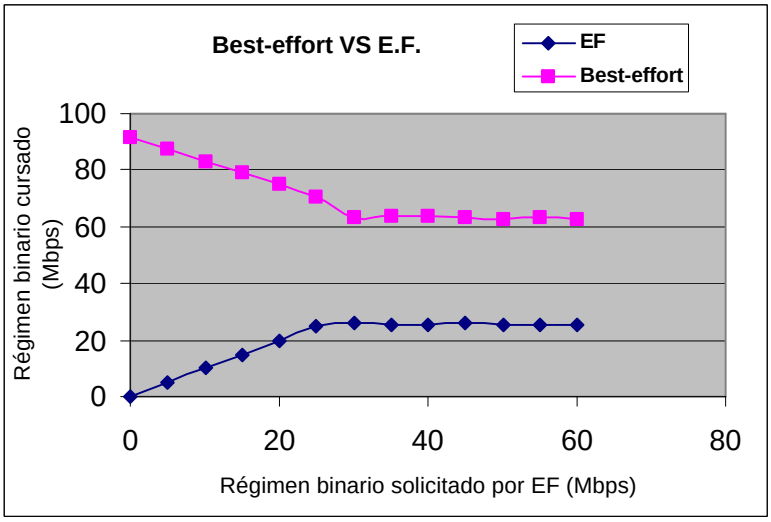
En la gráfica superior podemos ver como el retardo se mantiene por debajo de los 15 ms hasta que saturamos la cola y comienzan a descartarse paquetes a 20Mbps. Podemos asimismo ver como el retardo aumenta ligeramente a medida que aumentamos el régimen binario, esto es debido a la planificación del algoritmo WRR de la disciplina CBQ: *"delay if overlimit"*.

El Jitter (variación del retardo) sigue encontrándose alrededor de 0.4 ms por debajo del retardo máximo. Ese valor es el retardo mínimo.

6.4 Prueba de requisamiento de ancho de banda

En esta prueba se pretende comprobar como afecta a una cola –que tiene activado el flag de petición de ancho de banda sobrante– la ocupación de otra(s).

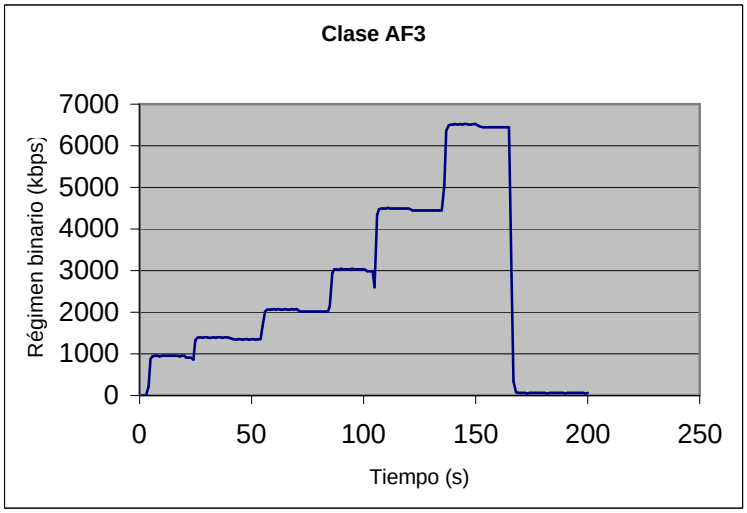
Para realizar esta prueba se midió el ancho de banda que conseguía cursar la cola Best-effort (creada con el flag de petición) frente a diversos valores de regímenes binarios en la cola EF. Para el test se generó un tráfico best-effort que intentaba ocupar todo el ancho de banda de la interfaz (100Mbps). Al mismo tiempo, la cola EF que tenía reservados 30Mbps, comenzaba a cursar tráfico incrementalmente hasta 60Mbps.



Como podemos observar, cuando por la clase EF no se cursa ningún tráfico, la clase Best-effort intenta apropiarse de todo el ancho de banda de la interfaz para ella. A medida que la clase EF comienza a requerir ancho de banda lo consigue ya que lo tiene reservado. No obstante cuando la clase EF requiere más ancho de banda del que tiene aprovisionado (30Mbps) no se la deja continuar y lo sobrante se le deja a la best-effort.

6.5 Prueba de reconfiguración dinámica

Finalmente vamos a probar el funcionamiento de un servicio de reconfiguración en el BB. Esta vez se generó un tráfico de 10Mbps sobre la clase AF3 y se programó un sencillo algoritmo de gestión de recursos en el BB: si se producen más de 20 descartes por segundo en la clase se aumenta el valor de los anchos de banda inferiores en un factor de 1,5. El funcionamiento es el siguiente: el router informa del uso de sus colas periódicamente al BB, éste hace las comprobaciones pertinentes y, si el algoritmo cambia algún dato de la tabla de configuración, activa un flag. Ese flag se le transmite al RA la próxima vez que quiera instalar o actualizar un tráfico. A la vista de ese flag el router desinstala suavemente todos sus tráficos y pide una nueva configuración.



La clase comienza con un ancho de banda de 900kbps (400+300+200kbps). Los valores teóricos de los escalones según el factor que aplica el BB serán: 900, 1350, 2025, 3037, 4556, 6834kbps y finalmente, antes de que llegue al máximo (10000 kbps), se desinstaló la conexión del BB por lo tanto el Router la clasificó como Best-effort de 50kbps. El resultado es el que tenemos en la gráfica y es bastante fiel a los cálculos teóricos.

7. CONCLUSIONES

Se ha comprobado la gran adaptabilidad a diferentes entornos. Simplemente cambiando las estructuras de datos que se necesiten transferir y guardándolas en determinados objetos se puede administrar la gran mayoría de entornos de políticas. Sin ir más lejos, en este proyecto se ha probado el protocolo COPS en dos entornos: marcado y MobyDick. Cabe destacar, no obstante, que hay que conocer profundamente su RFC para desarrollar un entorno COPS conforme al estándar. Una importante parte de este proyecto ha consistido en verificar tal cosa.

En lo referente a los programas clientes y servidor, aparte de lo realizado para su adaptación al nuevo entorno, se han reforzado sus procesos, se han corregido numerosos fallos y se ha incrementado su protección frente a errores. Cabe resaltar la gran importancia que tiene el Bandwidth Broker en nuestro escenario, él controla tanto las políticas como las capacidades de la red. Cualquier fallo en este proceso provocaría que la calidad de servicio de nuestra red no estuviese operativa.

El Router de acceso a la red es también una pieza muy importante de nuestra arquitectura, es él quien aplica los mecanismos necesarios para proporcionar calidad de servicio a las diferentes clases DiffServ y también es él el encargado de proporcionarle al Bandwidth Broker las estadísticas de uso de sus disciplinas e interfaz. En este proyecto, la interfaz BB-Router ha sido la pieza clave para la gestión dinámica de políticas.

Un apartado muy importante en este proyecto ha sido la librería TC API de IBM. De ella podemos destacar su enorme potencia y capacidad para controlar los mecanismos de calidad de servicio del kernel sin necesidad de ningún otro programa. También destacamos su facilidad de instalación (ejecutar un makefile) y de inclusión en los programas (incluir su librería de C). No obstante esta librería ha sido la parte que más trabajo ha provocado en el

proyecto debido a su pobre documentación, falta de ejemplos o ejemplos con errores y dificultad de uso. Además de esto, esta librería, aunque se encuentra ya en su versión 1.0, tiene todavía bastantes fallos sobre todo a la hora de tomar estadísticas. La herramienta '*tc*' es similar en cuanto a funcionalidad al TC API y puede ser un buen referente en cuanto a ayuda y ejemplos.

Finalmente comentar que, en el entorno implementado, se ha demostrado la correcta funcionalidad de la gestión dinámica de políticas y la conforme aplicación de los requerimientos de calidad de servicio a los tráficos en el Router.

A. Datos Personales

Nombre:	Pedro Antonio
Apellidos:	Vico Solano
Número de colegiado al COIT:	11390
Número de asociado al COIT:	13254

C. Datos del proyecto fin de carrera

Tutor de proyecto:	José Ignacio Moreno Novella
Departamento:	Ingeniería telemática
Universidad:	Carlos III de Madrid
Título:	*Gestión dinámica de políticas bajo entorno DiffServ*
Fecha de lectura:	15 de octubre de 2002
Calificación:	Matricula de honor

D. Publicaciones del autor

- "Development and implementation report on the QoS components for Moby Dick. D0202"
  Miembros del grupo WP2 (QoS). Proyecto europeo Moby Dick IST-2000-25394. Julio 2003. http://www.ist-mobydick.org/deliverables/D0202.pdf

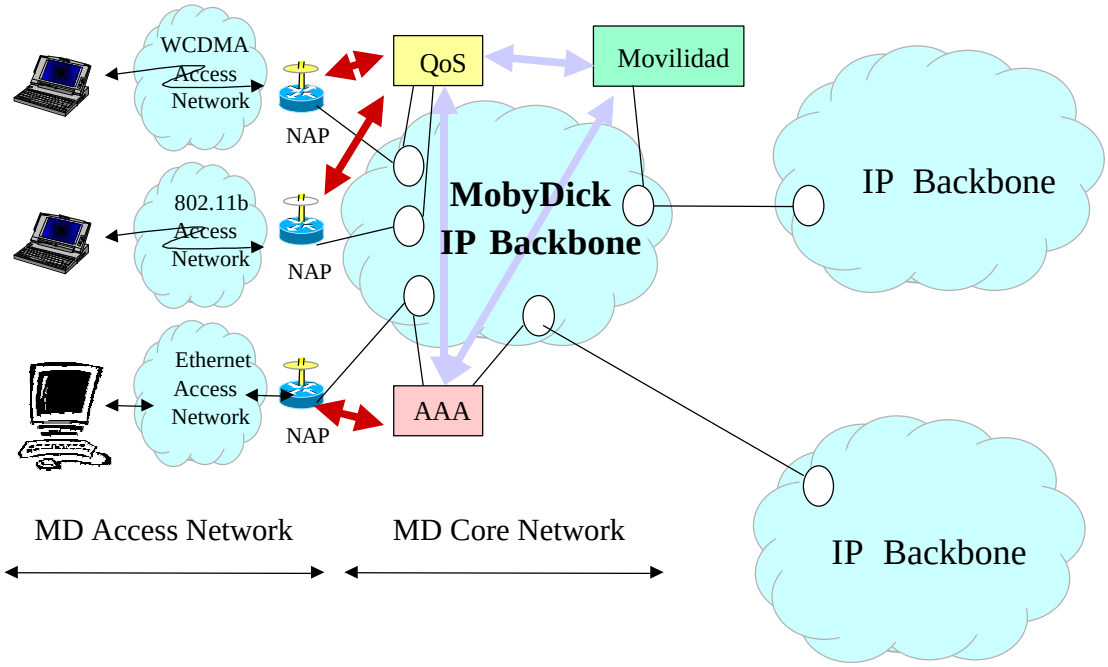
- "Provisión de QoS en redes móviles de cuarta generación"
  C. García, P.A. Vico, A. Cuevas, J.I. Moreno, I. Soto. Articulo finalista de las 4as jornadas de ingeniería telemática: JITEL. Septiembre 2003. ISBN-84-96131-38-6

E. Méritos

La meta final del presente proyecto consistió en desarrollar una porción del proyecto europeo sobre redes móviles de cuarta generación llamado MobyDick. http://www.ist-mobydick.org/ *Proyecto europeo número: IST-2000-25394.*

El proyecto MobyDick (**Mobility and Differentiated Services in a Future IP Network)** tuvo como propósito el diseño, implementación y prueba de una arquitectura de red móvil de cuarta generación en la que existan aspectos de AAAC (autenticación, autorización, contabilidad y cobro), calidad de servicio (QoS) y movilidad (MIP) todo ello sobre IPv6 y a través de diferentes medios de acceso (Ethernet, WLAN, WCDMA).

En la página web arriba citada y en el proyecto adjunto se puede encontrar más información sobre este proyecto, no obstante en la siguiente figura se presenta un resumen de su arquitectura y componentes básicos:



En este proyecto intervinieron importantes empresas de telecomunicaciones internacionales tales como NEC, Motorola y Deutsche Telekom y, varias universidades europeas como la Carlos III de Madrid y la de Stuttgart.

Dentro del MobyDick mi estudio se centró en el desarrollo y pruebas de un entorno de calidad de servicio según el método DiffServ sobre el protocolo de red IPv6.

Esto incluye el desarrollo de un programa gestor de la calidad de servicio (QoSManager) en los routers de acceso a la red MobyDick que maneje el ancho de banda y la prioridad de los tráficos, su protocolo de políticas asociado (COPS) ya que la decisión del ancho de banda asignado se toma en una entidad interna centralizada, la interacción de éstos con los demás elementos de la arquitectura (movilidad y AAAC), su documentación asociada y las pruebas oportunas para demostrar su correcto funcionamiento.