ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

ESTUDIO DE “DIFFERENTIATED SERVICES (DIFFSERV)” USANDO EL SISTEMA OPERATIVO ABIERTO (LINUX) PARA LA PROVISIÓN DE CALIDAD DE SERVICIO EN REDES CON VoIP

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

RICHARD ALEXANDER CASTILLO OCHOA
CARLOS XAVIER MENDOZA BARBERÁN

DIRECTOR: ING. CARLOS HERRERA

Quito, Marzo 2007

DECLARACIÓN

Nosotros, Richard Alexander Castillo Ochoa y Carlos Xavier Mendoza Barberán, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Prioridad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Richard Alexander Castillo Ochoa

Carlos Xavier Mendoza Barberán

AGRADECIMIENTO

A Dios por darme la oportunidad de concluir mis estudios en compañía de mi familia y de mis amigos, a nuestro director Ingeniero Carlos Herrera por todo el tiempo y la ayuda brindada, y todas las demás personas que de una u otra manera ayudaron a conseguir la finalización de este proyecto.

Carlos Mendoza.

Agradezco a Dios por haberme dado la fuerza necesaria para concluir está etapa de mi vida, a mi familia por el apoyo y la comprensión dada durante mi vida estudiantil, a nuestro director Ingeniero Carlos Herrera, por el tiempo dedicado y los consejos dados, a todos mis amigos que siempre en las buenas y las malas supieron indicarme el camino indicado, y en especial a mi ángel de la guarda, mi hermano David.

Richard Castillo O.

DEDICATORIA

A Dios, por toda la ayuda y bendiciones recibidas en mi vida.

A mis padres, Ángela Orlando por todo el apoyo, por todo su amor incondicional, por todo el esfuerzo que han hecho para poder darme una vida maravillosa.

A mi hermana Karen, a mi tía Sergia, a mi tía Rosario, por todo el amor y la paciencia que me han tenido.

A mis familiares que siempre confiaron en mí y me dieron su orientación y cariño.

A Gaby, por estar a mi lado y apoyarme cada vez que lo necesite.

A mis amigos, por sus consejos, su ayuda y sobre todo por su amistad.

Muchas gracias a todos!

Carlos Mendoza

A Dios, por darme la dicha de tener una vida llena de bendiciones.

A mis padres, José y Judith, por su gran amor y paciencia, por el esfuerzo y apoyo incondicional durante toda mi vida.

A mis hermanos, David, Andrés y Lourdes, por siempre estar conmigo, darme su apoyo y sobre todo ser mis amigos.

A todos mis amigos por haber estado conmigo en las buenas y las malas, para reír y llorar, y por sobre todo por haber estado juntos y ayudándonos a salir adelante durante nuestra vida universitaria.

A todos, muchas gracias.

Richard Castillo O.

INDICE

INDICE DE GRAFICOS

INDICE DE TABLAS

PRESENTACIÓN

El presente trabajo esta orientado a brindar Calidad de Servicio a la Voz sobre IP (VoIP), mediante la implementación de los Servicios Diferenciados, que nos permite optimizar, la utilización del ancho de banda disponible en una Red de Datos, pues en la actualidad es necesario brindar ciertos tratamientos especiales a determinados servicios, como la voz, que no pueden ser tratados como los paquetes de datos.

Se explica como funciona la Voz sobre el Protocolo IP (VoIP), y las dificultades que se tiene en la implementación de la misma en la actualidad, al tener que desaprovechar el ancho de banda disponible, realizando segmentaciones de canal, que solo incrementan costos y hacen difícil la masificación del servicio.

También se muestran las ventajas que nos brinda la utilización del Sistema Operativo Linux en la consecución de los objetivos trazados, pues nos facilita la implantación del protocolo de Servicios Diferenciados en una red ya establecida, así como un ahorro económico, lo que resulta de gran ayuda, a la hora de la aplicación.

CAPÍTULO 1

REDES DE VoIP

1.1 GENERALIDADES DE VoIP. [1]

Voz sobre IP es, a grandes rasgos, un sistema de enrutamiento de conversaciones de voz mediante paquetes de datos, a través de redes como por ejemplo una red LAN Ethernet, mediante el protocolo IP (Internet Protocol).

Los algoritmos de compresión de voz, permiten enviar la información minimizando el ancho de banda, siendo posible utilizar el protocolo IP, con una respuesta aceptable.

VoIP aprovecha las ventajas desarrolladas en las redes de datos para transmitir voz en forma eficiente, aumentando las facilidades del usuario final. Las aplicaciones pueden ser, conexiones PC a PC; PC a teléfono; teléfono a teléfono; tráfico de fax y mensajería unificada, Todas estas aplicaciones atraviesan backbones IP, Internet e Intranets, etc.

La clasificación de VoIP de acuerdo al tipo de tráfico que entrega la Central Telefónica es de dos tipos, que corresponden a:

- ***Transporte IP*** es aquella conexión donde la Central Telefónica tiene una tarjeta, generalmente ISDN (PRI o BRI), seguida de un Gateway IP externo, el cual convierte la señal digital en paquetes IP permitiendo el transporte de voz en paquetes a través de la red IP hasta el Gateway IP remoto, tal como se indica en el GRÁFICO 1.1

[1] Gabriel Fernández Crocco, Tesinas de Belgrado, Pág. 5., http://www.ub.edu.ar/investigaciones/tesinas/33_crocco.PDF.

- ***Telefonía IP Pura*** se refiere a la Central Telefónica que tiene la capacidad de entregar por una de sus tarjetas, los paquetes de voz en IP estándar. Esto permite la interconexión de dispositivos IP, como teléfonos IP o Fax IP, los cuales reciben las mismas facilidades básicas que cualquier abonado directamente conectado a la Central, como se indica en el GRÁFICO 1.2.

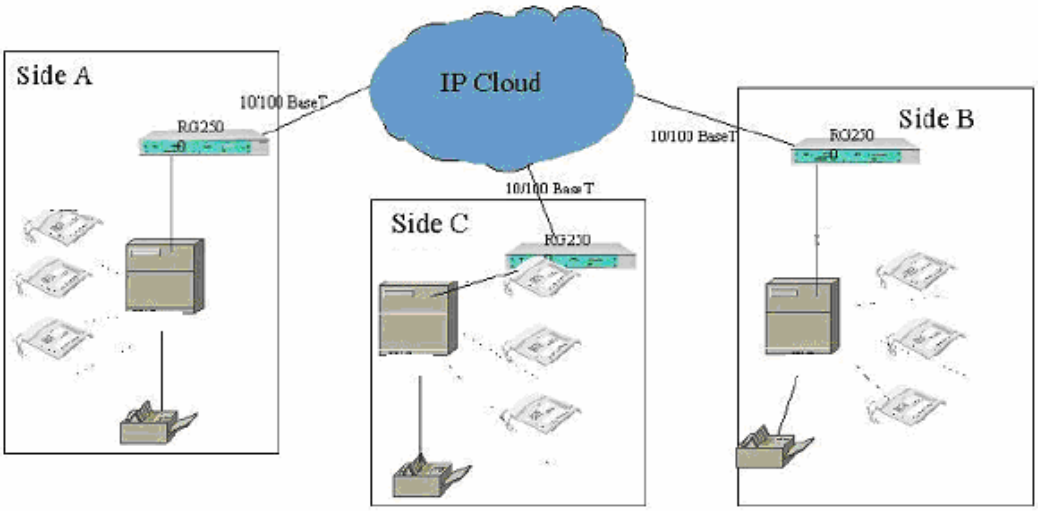


GRÁFICO 1.1. Transporte IP (Cortesía SIEMENS)

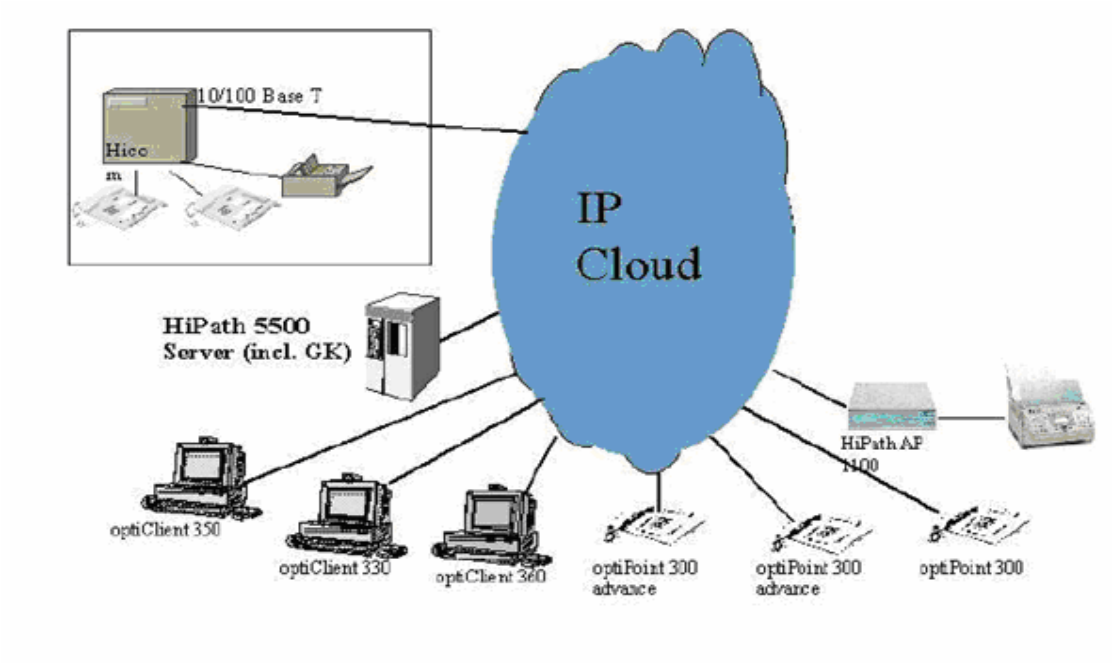


GRÁFICO 1.2. Telefonía IP (Cortesía SIEMENS)

Actualmente, VoIP se presenta como una alternativa del servicio actual de telefonía, pero no como un reemplazante inmediato.

Las ventajas de VoIP sobre la telefonía tradicional, comprenden los siguientes puntos:

- Fusión de infraestructuras de voz y datos, con la consiguiente reducción de costos que la agrupación de los equipos de soporte conlleva.

- Evitar los costos elevados de las llamadas que atraviesan las redes públicas de telefonía (PSTN).

- Minimizar el ancho de banda, bajando los costos para aquellos casos donde ya existen enlaces de datos y voz separados.

- Agregar nuevos servicios como mensajería unificada, interacción con aplicaciones de agendas y calendarios, multimedia, educación a distancia y trabajo basado en PC desde el hogar.

El éxito de incorporar esta nueva tecnología satisfactoriamente, radica en impactar lo menos posible al usuario final y no quitar la telefonía tradicional, hasta que sea estrictamente necesario realizar el cambio. El método de la tarifación debe ser compatible para asegurar la convivencia de los dos sistemas.

1.2 CONCEPTOS DE VoIP

1.2.1 ¿QUÉ ES VoIP? [2]

VoIP es la *tecnología* que permite la transmisión de voz sobre redes IP y los servicios asociados que son brindados por la misma. VoIP es el acrónimo de Voice over Internet Protocol, tal como dice el término intenta permitir que la voz viaje en paquetes IP.

La Telefonía IP es el *servicio* que utiliza la tecnología de VoIP para brindar comunicaciones telefónicas a los usuarios.

La telefonía IP conjuga dos mundos históricamente separados: la transmisión de voz y la de datos. Se trata de transportar la voz, previamente convertida a datos, entre dos puntos distantes. Esto posibilita utilizar las redes de datos para efectuar las llamadas telefónicas, y desarrollar una única red convergente que se encargue de cursar todo tipo de comunicación, ya sea voz, datos, o cualquier tipo de información.

VoIP, por lo tanto, no es en sí mismo un servicio, sino una tecnología que permite encapsular la voz en paquetes para poder ser transportados sobre

[2] www.monografias.com: Voz sobre IP (VoIP)

redes de datos, sin necesidad de disponer de los circuitos conmutados, o sea, la realización de una comunicación que requiere el establecimiento de un circuito físico durante el tiempo que dura ésta, lo que significa que los recursos que intervienen en la realización de una llamada no pueden ser utilizados en otra hasta que la primera no finalice, incluso durante los silencios que se suceden dentro de una conversación típica.

En cambio, la telefonía IP no utiliza circuitos para la conversación, sino que envía múltiples conversaciones a través del mismo canal codificadas en paquetes y flujos independientes. Cuando se produce un silencio en una conversación, los paquetes de datos de otras conversaciones pueden ser transmitidos por la red, lo que implica un uso más eficiente de la misma.

1.2.2 ¿CÓMO FUNCIONA VoIP? [3]

VoIP funciona digitalizando la voz en paquetes de datos, enviándola a través de la red y reconvirtiéndola a voz en el destino, para lo cual se van a utilizar a más de los componentes ya conocidos como ruteadores, PBX y teléfonos, los teléfonos IP, soft – phones, los servidores de telefonía y las pasarelas.

Básicamente el proceso comienza con la señal análoga del teléfono que es digitalizada por medio del codificador/decodificador de voz (CODEC). Las muestras digitalizadas son pasadas al algoritmo de compresión, el cual, comprime la voz y la fracciona en paquetes que pueden ser transmitidos para este caso a través de una red privada WAN, como se indica en el GRÁFICO 1.3. En el otro extremo de la nube se realizan exactamente las mismas funciones en un orden inverso.

[3] http://www.tecnovida.com.ve/category/telecomunicaciones/ Voz sobre IP.htm
José M. Huidrobo y David Roldan, “Integración de Voz y Datos”, Pág. 131.

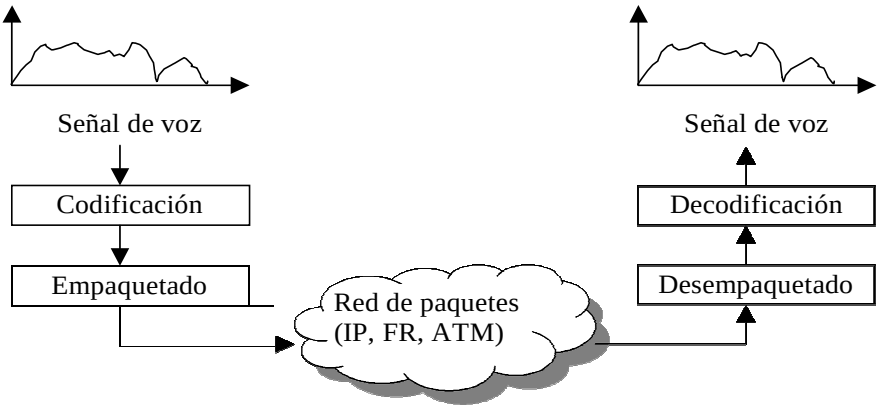


GRÁFICO 1.3. Como funciona VoIP

Una vez que hemos indicado que la voz requiere ser digitalizada para su transmisión por la red de paquetes y que dicha digitalización se la lleva a cabo por medio de un CODEC, lo que nos falta por definir es en donde tiene lugar dicha conversión, o lo que es lo mismo la ubicación del CODEC.

La solución depende del tipo de terminales disponibles para usuarios, por ejemplo si contamos con teléfonos analógicos convencionales, el CODEC estará ubicado en la PBX (o IP-PBX); otra solución sería incorporar el CODEC dentro del propio teléfono, dando lugar a un nuevo tipo de teléfonos digitales denominados teléfonos IP.

Sin embargo aún tenemos otra opción, que es emplear un soft – phone, que no es más que la aplicación de un software que se ejecuta en el PC del usuario y que se conecta con el servidor de telefonía para efectuar las funcionalidades telefónicas.

1.2.3 ANCHO DE BANDA PARA VoIP[4]

Hasta hace muy poco tiempo el ancho de banda necesario para la transmisión de voz y vídeo en tiempo real era considerablemente elevado, lo que hacia imposible este tipo de comunicaciones sobre redes de datos, que no garantizan una Calidad de Servicio, como por ejemplo Internet o redes basadas en protocolo IP.

Pero en la actualidad, el mejoramiento de las tecnologías para el uso de medios de transmisión guiados, los mecanismos de compresión de datos y la implementación de CODECs para la transmisión de voz y video, han hecho posible la transmisión de estas aplicaciones.

Actualmente la voz es digitalizada y comprimida según distintos algoritmos (GSM, G.723.1, G.711, G.729), los cuales se caracterizan por conseguir mayores relaciones de compresión en detrimento del tiempo de latencia (tiempo necesario para descomprimir la voz para que pueda ser entendida de nuevo).

La tasa de bits a la que se comprime la voz varía según la implementación y no hay una clase predominante, depende del CODEC que se implemente.

El protocolo IP añade al paquete de voz digitalizado y comprimido una serie de cabeceras para su correcto transporte a través de la red, lo que hace que el ancho de banda necesario se incremente hasta unos 16 Kbps.

---

[4] Marcos Valiño García, VoIP: una puerta hacia la convergencia.pdf, Pág. 5, http://www.cesga.es/component/option,com_docman/task,doc_download/gid,54/Itemid,13/lang,gl/.

1.2.3.1 Codificadores de voz: [5]

Los codificadores de voz (COder/DECoder) también llamados Voice CODECS, que lo llamaremos CODECS, son algoritmos de codificación que convierten la señal de voz analógica en un flujo digital de datos mediante muestreo, que es la discretización de la señal en el tiempo y cuantificación de la amplitud.

Los algoritmos, agregan facilidades como: detección de inactividad, cancelación de eco y reducción de ruido. Las características de calidad y retardo varían según cada implementación y no hay una clase predominante.

Las siguientes recomendaciones de la ITU-T establecen las características de los CODECS cuyos requerimientos de tasa de transmisión de la voz comprimida se indican en el TABLA 1.1.

TABLA 1.1 Características de CODECS

CODEC	**Método de Compresión**		**Bit rate R (Kbps)**
G.711	PCM	Pulse Code Modulation	64
G.723.1	CELP	Code Excited Linear Prediction	5.3
G.723.1	MP-MPLG	Low bit rate vocoder for Multimedia	6.4
G.726	ADPCM	Adaptive Differential PCM	32
G.728	LD-CELP	Low Delay CELP	16
G.729	CS-ACELP	Conjugate Structure Algebraic CELP	8

En la mayoría de las implementaciones los paquetes de VoIP están precedidos de un encabezamiento por cada nivel, como se indica en el GRÁFICO 1.4:

- IP (Internet Protocol) – 160 bits
- UDP (User Datagram Protocol) – 64 bits
- RTP (Real-time Transport Protocol) – 96 bits

[5] Gabriel Fernández Crocco, Tesinas de Belgrado, Pág. 8, http://www.ub.edu.ar/investigaciones/tesinas/33_crocco.PDF.

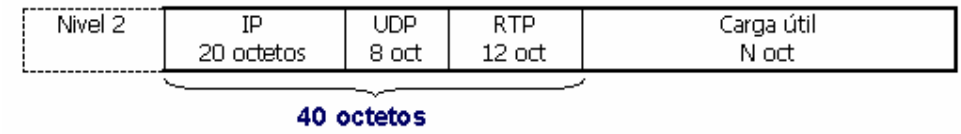


GRÁFICO 1.4 Cabecera del Paquete VoIP

Sumando los bits de cada nivel resulta un encabezado total por paquete (PS) de 40 bytes:

$$PS_{OH_N} = PS_{OH_{IP}} + PS_{OH_{UDP}} + PS_{OH_{RTP}}$$

$$PS_{OH_N} = 160bits + 64bits + 96bits = 320bits$$

$$PS_{OH_N} = 320bits \times \frac{1byte}{8bits} = 40bytes$$

$$donde:$$

$$P_{OH_N} : encabezado\ total\ por\ paquete$$

$$P_{OH_{IP}} : encabezado\ del\ encapsulamiento\ IP$$

$$P_{OH_{UDP}} : encabezado\ del\ encapsulamiento\ UDP$$

$$P_{OH_{RTP}} : encabezado\ del\ encapsulamiento\ RTP$$

ECUACIÓN 1.1 Suma total de los bits de la Cabecera del paquete VoIP

En donde PS_{OHN} corresponde a la suma total de las cabeceras IP, UDP y RTP.

El ancho de banda de encabezamiento (BW_{OHVoIP}) podemos obtenerlo al multiplicar el número de bits de la cabecera (PS_{OHN}), obtenida de la ecuación 1.1, por la tasa de transferencia o envío de paquetes (PR).

Supongamos una tasa de transferencia de paquetes de 50 paquetes por segundo (pps), es decir una frecuencia (R=50*1/s) de 50 paquetes enviados en un segundo, con lo cual el ancho de banda del encabezado, BW_{OHVoIP} está dado por la ecuación 1.2:

$$BW_{OH_{VoIP}} = PS_{OH_N} \times R$$

$$BW_{OH_{VoIP}} = 320bits \times \frac{50}{s} = 16Kbps$$

$donde:$

$BW_{OH_{VoIP}}$: $Ancho\,de\,Banda\,del\,encabezado\,del\ paquete\,de\,VoIP$

PS_{OH_N} : $encabezado\ total\ por\ paquete$

ECUACIÓN 1.2 Ancho de banda de encabezamiento (BW_{OHVoIP})

El encabezado total se antepone a cada paquete de VoIP siendo una constante dependiente de los protocolos que se utilicen para la comunicación de los datos.

El tamaño del paquete de VoIP es la carga útil (payload), medida en bytes que equivale al tiempo de duración del paquete en el canal expresado en milisegundos, calculado a una determinada tasa de envío de paquetes. El tiempo de duración de cada paquete es la inversa de la frecuencia.

El tiempo de duración del paquete de voz está referida a la cantidad de muestras, esto implica una situación de compromiso entre ancho de banda y calidad de la voz.

Entonces, aumentar el tiempo de duración del paquete, equivale a incrementar la carga útil, podremos tomar más muestras mejorando la calidad de la voz, pero a su vez se eleva el retardo total en la transmisión, siendo de esta forma más probable la pérdida de paquetes.

La pérdida de paquetes ocasiona un decaimiento de la calidad de voz. La tasa de pérdida de paquetes (Paquet Loss Rate, PLR) para G.711 que tiene una tasa de CODEC (Rc) de 64 Kbps, la podemos calcular por ejemplo para una carga útil (PS) de 32; 48 y 64 bytes, como se indica en la ecuación 1.3:

$$PLR = \frac{PS}{R_C}$$

$$PLR_{32} = \frac{32bytes \times \frac{8bits}{1byte}}{64000\frac{bits}{s}} = 0.004 \Rightarrow PLR_{32} = 0.4\%$$

$$PLR_{48} = \frac{48bytes \times \frac{8bits}{1byte}}{64000\frac{bits}{s}} = 0.006 \Rightarrow PLR_{48} = 0.6\%$$

$$PLR_{64} = \frac{64bytes \times \frac{8bits}{1byte}}{64000\frac{bits}{s}} = 0.008 \Rightarrow PLR_{64} = 0.8\%$$

$donde:$

$PLR: Paquet\ Loss\ Rate$

$PS: Carga\ útil$

$Rc: Tasa\ de\ transmisión\ de\ CODEC$

ECUACIÓN 1.3 Tasa de pérdida de paquetes (PLR)

Una alternativa para reducir la tasa de pérdida de paquetes es implementar buffers de variación de retardo para paquetes de gran tamaño, pero esto resulta en retardos de encolamiento muy elevados, que a su vez implican niveles de calidad de voz inaceptables por encima de los 30 ms.

El buffer de variación acomoda los paquetes y los reproduce con la cadencia original.

Por otra parte, al reducir excesivamente la duración del paquete, aparecen dos limitaciones:

- La primera es la necesidad de tomar al menos una muestra.

- La segunda se refiere a la eficiencia del paquete, es decir que la relación carga útil versus el encabezado se mantenga en proporciones razonables.

El tiempo de duración del paquete no debe ser inferior a los 10 ms. El tiempo de duración óptimo del paquete es de 20 ms quedando dentro de la banda de 10 ms < t < 30 ms.

La duración del paquete se expresa mediante la relación de tasa de paquetes por segundo (pps), que equivale a la cantidad de paquetes que se transmiten en el período de 1 segundo.

En este ejemplo la tasa de paquete (PR), se calcula como la inversa del tiempo de duración (Rt) de 20 ms, obteniendo los 50 paquetes por segundo, como se indica en la ecuación 1.4:

$$PR = \frac{1}{R_t} = \frac{1}{20 \times 10^{-3}(s)}$$

$$PR = 50\,pps$$

$$\begin{aligned} &donde: \\ &PR: Tasa\ de\ paquete \\ &R_t: duración\ del\ paquete \end{aligned}$$

ECUACIÓN 1.4 Tasa de paquetes (PR)

En el caso de transmitir un paquete de VoIP a través de una WAN se considera, no solo el tamaño del encabezamiento de los niveles superiores (PSohn), sino también el encabezado de los protocolos de nivel inferior (PSohl) como PPP, éstos ocupan entre 6 y 20 bytes.

El tamaño del encabezado total (PSoht), se indica en la ecuación 1.5:

$$PS_{OH_T} = PS_{OH_N} + PS_{OH_L}$$

$$PS_{OH_T} = 40bytes + 20bytes = 60bytes$$

$donde:$

PS_{OH_T} : $Tamaño\ total\ del\ encabezado\ al\ enviar\ el\ paquete\ en\ una\ WAN$

PS_{OH_N} : $Encabezado\ total\ del\ paquete\ de\ los\ niveles\ \sup eriores$

PS_{OH_L}: $Encabezado\ total\ del\ paquete\ de\ los\ niveles\ \inf eriores$

ECUACIÓN 1.5 Encabezado total del paquete VoIP (PS_T)

La codificación G.711 tiene una tasa de CODEC (Rc) de 64 Kbps, para un payload de paquete IP (PS_{PL}), de 80 bytes, con una duración de paquete (Rt) de 10 ms, podemos calcular el ancho de banda total (BW) para cada sesión utilizando la ecuación 1.6:

$$PS_{PL} = R_C \times R_t$$

$$PS_{PL} = 64000\frac{bits}{s} \times 10 \times 10^{-3} s = 640bits$$

$$PS_{PL} = 640bits \times \frac{1byte}{8bits} = 80bytes$$

$$PR = \frac{1}{R_t} = \frac{1}{10 \times 10^{-3} s} = 100\,pps$$

$$BW_{OH} = PS_{OH_T} \times PR$$

$$BW_{OH} = 60bytes \times \frac{8bits}{1byte} \times 100\,pps$$

$$BW_{OH} = 48Kbps$$

$$BW_{PL} = PS_{PL} \times PR$$

$$BW_{PL} = 80bytes \times \frac{8bits}{1byte} \times 100\,pps$$

$$BW_{PL} = 64Kbps$$

$$BW = BW_{OH} + BW_{PL} = 48Kbps + 64Kbps$$

$$BW = 112Kbps$$

$$
\begin{aligned}
&donde: \\
&PS_{PL} : Carga\ útil\ del\ paquete\ IP \\
&Rc : Tasa\ de\ CODEC \\
&R_t : Duración\ del\ paquete\ en\ segundos \\
&BW_{OH} : Ancho\ de\ Banda\ del\ encabezado \\
&BW_{PL} : Ancho\ de\ Banda\ de\ la\ carga\ útil \\
&PR : Tasa\ de\ paquete \\
&BW : Ancho\ de\ Banda\ total\ del\ paquete
\end{aligned}
$$

ECUACIÓN 1.6 Ancho de Banda para un paquete VoIP utilizando la codificación G.711, con una carga útil de 80 bytes.

La tabla 1.2 muestra los mismos cálculos para las distintas tasas de CODECs (Rc) y con tres valores de duración de paquete (Rt) diferentes 10 ms; 20 ms; 30 ms. con el tamaño de encabezado (Psoh) constante en 60 bytes.

Tasa de CODEC $R_C\,[Kbps]$		Duración del Paquete $R_t\,[ms]$	Tamaño del Encabezado PS_{OH}	Tamaño de Carga útil $PS_{PL}\,[bytes]$	Tasa de Envío $PR\,[pps]$	Ancho de Banda de Encabezado $BW_{OH}\,[Kbps]$	Ancho de Banda de Carga útil $BW_{PL}\,[Kbps]$	Ancho de Banda de VoIP en un sentido $BW\,[Kbps]$
				$PS_{PL} = R_C \times R_t$	$PR = \frac{1}{R_t}$	$BW_{OH} = PS_{OH} \times PR$	$BW_{PL} = PS_{PL} \times PR$	$BW = BW_{OH} \times BW_{PL}$
G.711	64	10	60	80	100	48	64	112
G.723.1	5.3	10	60	7	100	48	5.3	53.3
G.723.1	6.4	10	60	8	100	48	6.4	54.4
G.726	32	10	60	40	100	48	32	80
G.728	16	10	60	20	100	48	16	64
G.729A	8	10	60	10	100	48	8	56
G.711	64	20	60	160	50	24	64	88
G.723.1	5.3	20	60	13	50	24	5.3	29.3
G.723.1	6.4	20	60	16	50	24	6.4	30.4
G.726	32	20	60	80	50	24	32	56
G.728	16	20	60	40	50	24	16	40
G.729A	8	20	60	20	50	24	8	32
G.711	64	30	60	240	34	16.48	65.92	82.4
G.723.1	5.3	30	60	20	34	16.48	5.459	21.94
G.723.1	6.4	30	60	24	34	16.48	6.592	23.07
G.726	32	30	60	120	34	16.48	32.96	49.44
G.728	16	30	60	60	34	16.48	16.48	32.95
G.729A	8	30	60	30	34	16.48	8.24	24.72

TABLA 1.2 Ancho de Banda para diferentes CODECS

El GRÁFICO 1.5 muestra la diferencia en los anchos de banda requeridos para la transmisión de paquetes de VoIP, según el algoritmo del CODEC utilizado y la duración del paquete elegido:

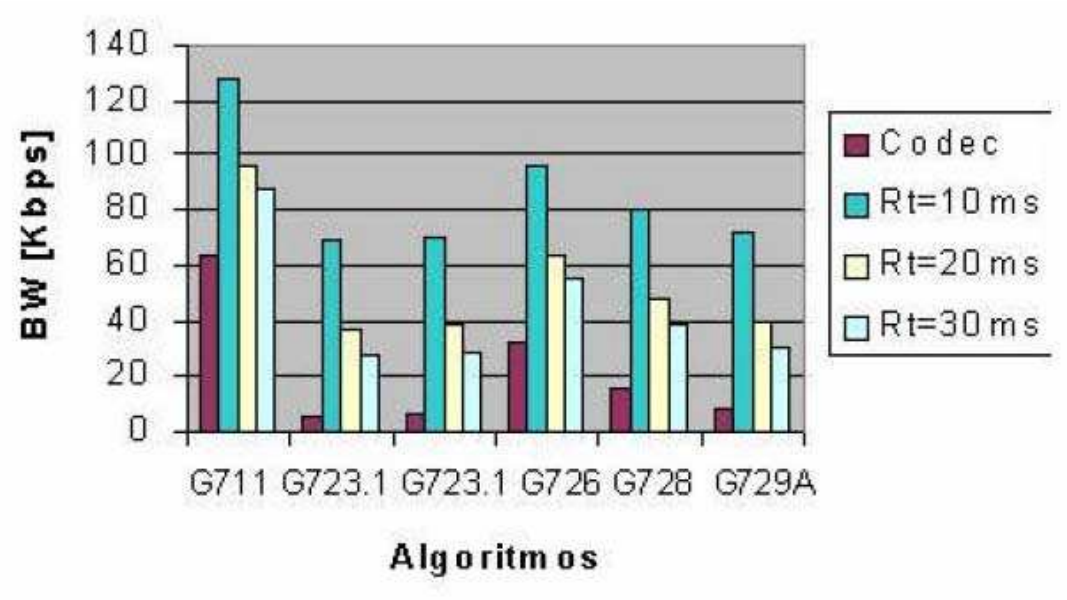


GRÁFICO 1.5 Ancho de Banda (BW) para duración de paquete (Rt) variable

Del GRÁFICO anterior se puede apreciar que el ancho de banda requerido depende de la elección del CODEC.

De la tabla 1.2, podemos obtener el resultado de Ancho de Banda requerido, para aplicar la norma H.323, con una duración del paquete (Rt = 20 ms) y un encabezado total del paquete (PS_T = 60 bytes), para los siguientes CODECS:

- G.711 => BW > 88 Kbps
- G.723.1 => BW > 30 Kbps
- G.726 => BW > 56 Kbps
- G.728 => BW > 40 Kbps
- G.729A => BW > 32 Kbps

Este análisis no es suficiente para realizar la elección del CODEC, pues se debe considerar también el retardo que introduce cada algoritmo en el esquema global y la calidad de la voz que se mide mediante el parámetro denominado MOS (resultado de opinión media), tal como se muestra en la tabla 1.3.

TABLA 1.3 Retardo introducido por cada CODEC

CODEC	Método de Compresión	Bit rate R (Kbps)	Retardo D (ms)	MOS
G.711	PCM Pulse Code Modulation	64	0.75	4.1
G.723.1	CELP Code Excited Linear Prediction	5.3	30	3.65
G.723.1	MP-MPLG Low bit rate vocoder for Multimedia	6.4	30	3.9
G.726	ADPCM Adaptive Differential PCM	32	1	3.85
G.728	LD-CELP Low Delay CELP	16	5	3.61
G.729	CS-ACELP Conjugate Structure Algebraic CELP	8	10	3.92

El MOS se obtiene de una prueba denominada ACR (absolute category rating), en la cual se realizan pruebas de audición a un grupo heterogéneo de personas, con diez grabaciones diferentes, las que se califican con una puntuación en el rango 5: excelente y 1: inaceptable y luego se obtiene la media.

En el camino extremo a extremo, el paquete de VoIP atraviesa los siguientes procesos que afectan la QoS diagramados en el GRÁFICO 1.6:

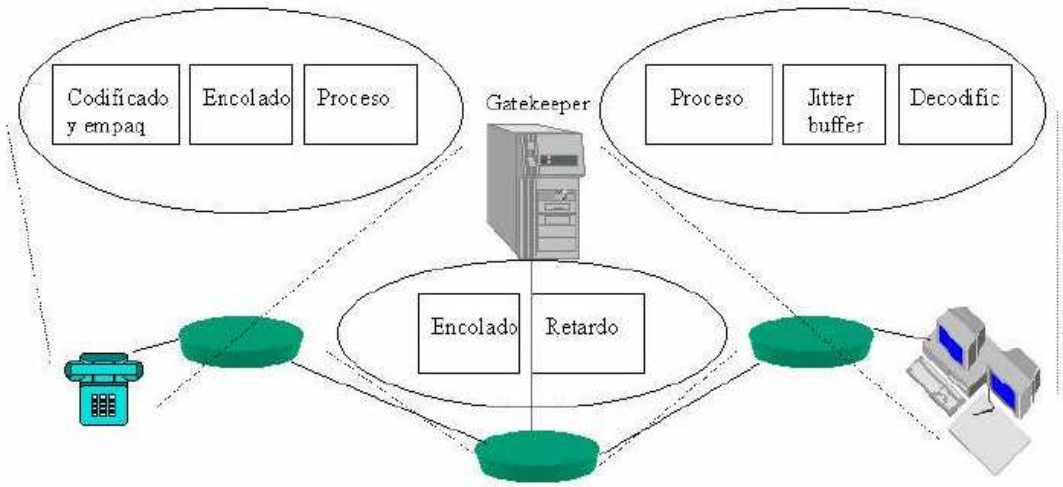


GRÁFICO 1.6 Procesos que afectan la QoS.

Los parámetros: Codificación del paquete, el encolado del paquete y el procesamiento del mismo, están relacionados estrechamente, las pruebas de retardo, entregan valores según el detalle de la tabla 1.4:

TABLA 1.4 Latencias introducidas por dispositivos de Networking			
Dispositivo de Networking		**Retardo mínimo (ms)**	**Retardo máximo (ms)**
Emisor	Codificado y empaquetado	30	50
	Encolado de paquetes	0	50
	Procesamiento	10	20
Ruteador	Encolado de paquetes	1	50
	Retardo de serialización	1	1
Receptor	Procesamiento	20	20
	Buffer de variación	30	300
	Decodificado	1	1
Retardo total (ms)		93	492
Retardo total promedio (ms)		292.5	

1.2.3.2 Mecanismos de Reducción de Ancho de Banda

Existen dos mecanismos válidos para lograr reducir el ancho de banda de las sesiones de VoIP.

- Compresión de cabeceras
- Supresión de Silencios

1.2.3.2.1 *Compresión de cabeceras*

Para una mejor transmisión de los datos en tiempo real, esta previsto el uso del Protocolo de Transporte Real (RTP), pero existen algunos inconvenientes.

Las cabeceras RTP/UDP/IP tienen 12, 8 y 20 bytes, respectivamente, aunque se puede comprimir a 2 ó 4 bytes utilizando la compresión de cabecera RTP, (RTP Comprimido, cRTP). De esta manera, en todo un tramo, se comprime los encabezados de los paquetes de voz, incluyendo el de IP/UDP/RTP (VoIP se

encapsula dentro de RTP, que se encapsula a su vez dentro de UDP. Por tanto, VoIP es transportado con una cabecera de paquete RTP/UDP/IP, como se indica en el GRÁFICO 1.7) logrando reducir los 40 bytes de encabezados.
Existe la recomendación de usar RTP Comprimido (cRTP), en los enlaces menores a los 768 Kbps. RTP Comprimido (cRTP), se utiliza en enlaces WAN, especialmente en enlaces punto-punto. Como la cabecera de UDP y RTP se reduce a un máximo de 4 bytes, no hay lugar para añadir en la cabecera la dirección IP. Por lo tanto, el paquete no puede ser enrutado y sólo puede ser utilizado en enlaces donde no resulte necesario direccionamiento IP.

La consecuencia de RTP Comprimido (cRTP), similar a cualquier forma de compresión, es que necesita más ciclos de procesado en el ruteador para tratar el paquete. El ruteador debe leer cada cabecera tan pronto llegue el paquete IP, y de esta forma, la información es enrutada a través de la LAN hasta el teléfono IP.

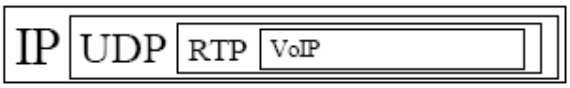


GRÁFICO 1.7 Encapsulamiento del Paquete VoIP

Sin embargo, hay algunos usos que se podrían ver afectados por la compresión. Un ejemplo es el impacto en usuarios que utilizan el módem. Los esquemas de compresión pueden interferir con el funcionamiento de módems confundiendo la codificación usada. El resultado podría ser que los módems nunca se sincronizan entre ellos, o que exhiben un rendimiento de procesamiento muy pobre.

Algunos gateways pudieron implementar una cierta inteligencia en ejecución que puede detectar el uso de módem e inhabilitar la compresión.

Otro argumento potencial es ocupar esquemas de compresión de discurso de bajo número de bit, tales como G.729 y G.723.1. Estos esquemas de codificación intentan reproducir el sonido subjetivo de la señal más que la forma de onda. Una mayor cantidad de pérdida de paquetes o de jitter es más sensible que la de una forma de onda no-comprimida.

1.2.3.2.2 *Supresión de Silencios*

Hay que considerar el parámetro denominado "supresión de silencio". Con este parámetro activado, se consigue que la transmisión de paquetes (uso de ancho de banda), se reduzca a las situaciones en que los agentes están hablando. El resto del tiempo (cuando no existe voz para transmitir) se libera el ancho de banda.

Considerando este aspecto, se puede afirmar que el tamaño medio de un paquete de voz durante una conversación es de 8 Kbps.

1.2.3.3 Características de la transmisión de paquetes de VoIP sobre WAN:

“De acuerdo a resultados empíricos, la calidad de la voz comienza a degradarse en un enlace WAN, cuando el retardo supera los 150 ms. Debemos considerar no solo el ancho de banda que ocupa el tráfico de VoIP sino también el tráfico de datos propiamente dicho. En enlaces de baja capacidad, es decir menores a 512 Kbps, se puede llegar a degradar la voz en forma notable cuando se transmiten los paquetes de VoIP que compiten con paquetes de datos o con otros paquetes de VoIP. Hay soluciones propietarias, como LFI (Link Fragmentation Interleave), utilizadas en enlaces de baja velocidad. Funcionan segmentando y entrelazando todos los paquetes para evitar la competencia con los pequeños paquetes de VoIP.

El proceso de serialización introduce un retardo dependiente del tamaño del paquete de VoIP, detallado en la tabla 1.5 y el GRÁFICO 1.8, donde se aplica la ecuación 1.7"[6]

$$D_S = \frac{PS}{BW} = \frac{64bytes \times \frac{8bits}{1byte}}{56000\frac{bits}{s}} = 9 \times 10^{-3} s = 9 ms$$

$$\begin{aligned} &donde: \\ &D_S : \text{Re}tardo\ de\ Serialización \\ &PS : C\arg a\ útil \\ &BW : Ancho\ de\ Banda\ de\ transmisión \end{aligned}$$

ECUACIÓN 1.7 Retardo de serialización

TABLA 1.5 Retardo de serialización

		Tamaño de paquete (PS) [bytes]						
		1	64	128	256	512	1024	1500
Ancho de Banda (BW) [Kbps]	56	143 us	9 ms	18 ms	36 ms	72 ms	144 ms	214 ms
	64	125 us	8 ms	16 ms	32 ms	64 ms	126 ms	187 ms
	128	62.5 us	4 ms	8 ms	16 ms	32 ms	64 ms	93 ms
	256	31 us	2 ms	4 ms	8 ms	16 ms	32 ms	46 ms
	512	15.5 us	1 ms	2 ms	4 ms	9 ms	16 ms	32 ms
	768	10 us	640 us	1.28 ms	2.56 ms	5.12 ms	10.24 ms	15 ms
	1536	5 us	320 s	640 us	1.28 ms	2.56 ms	5.12 ms	7.5 ms

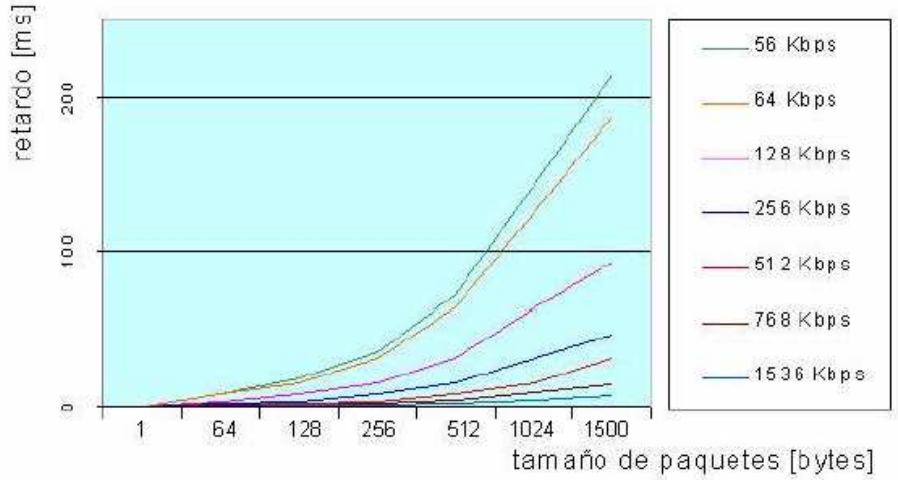


GRÁFICO 1.8 Retardo de serialización

[6] Gabriel Fernández Crocco, Tesinas de Belgrado, Pág. 13, http://www.ub.edu.ar/investigaciones/tesinas/33_crocco.PDF.

1.2.3.4 Características de la transmisión de paquetes de VoIP sobre LAN:

"Una de las principales desventajas para cualquier tráfico crítico respecto del tiempo y en particular los paquetes de VoIP en una LAN, es que los protocolos más utilizados en el nivel de enlace, Ethernet y Token Ring, trabajan con un tamaño de paquete variable. El equipamiento desarrollado para VoIP brinda sólo conectividad Ethernet, con un ancho de banda de transmisión (BW) de 10 Mbps. El tamaño de la carga útil del paquete (PS_{PL}) varía entre 46 y 1500 bytes y el encabezado (Ps_{OH}) ocupa entre 14 y 20 bytes.

Para calcular la tasa de paquetes Ethernet se aplica la ecuación 1.8, se reitera el mismo cálculo para los distintos tamaños de paquete en la tabla 1.6."[7]

$$PR = \frac{BW}{PS_{OH} + PS_{PL}} = \frac{10000000\frac{bits}{s}}{(64bytes + 20bytes) \times \frac{8bits}{1byte}} = 14880\,pps$$

$$\begin{aligned} &donde: \\ &PR: Tasa\ \ del\ \ paquete \\ &BW: Ancho\,de\,Banda\,de\,Transmisión \\ &PS_{OH}: Tamaño\,del\,encabezado \\ &PS_{PL}: Tamaño\,de\,la\ \ c\arg a\ \ útil \end{aligned}$$

ECUACIÓN 1.8 Tasa de paquetes Ethernet

TABLA 1.6 Tasa de paquetes Ethernet

Tamaño de carga útil [bytes]	Tamaño de encabezado [bytes]	Tamaño de paquetes [bytes]
64	20	14880
128	20	8445
256	20	4528
512	20	2349
768	20	1586
1024	20	1197
1280	20	961
1518	20	812

[7] Gabriel Fernández Crocco, Tesinas de Belgrado, Pág. 14, http://www.ub.edu.ar/investigaciones/tesinas/33_crocco.PDF.

La utilización del ancho de banda y la tasa de paquetes pueden o no coincidir debido a la variación del tamaño de los paquetes. La utilización se incrementa debido a la variación de dos factores, aumento de la cantidad de paquetes y/o el tamaño de la carga útil de los mismos.

En Ethernet, la disponibilidad de ancho de banda es dependiente del número de colisiones, en forma exponencial, debido a la forma de trabajo del mecanismo CSMA/CD de la norma Ethernet 802.3, así se representa en el GRÁFICO 1.9.

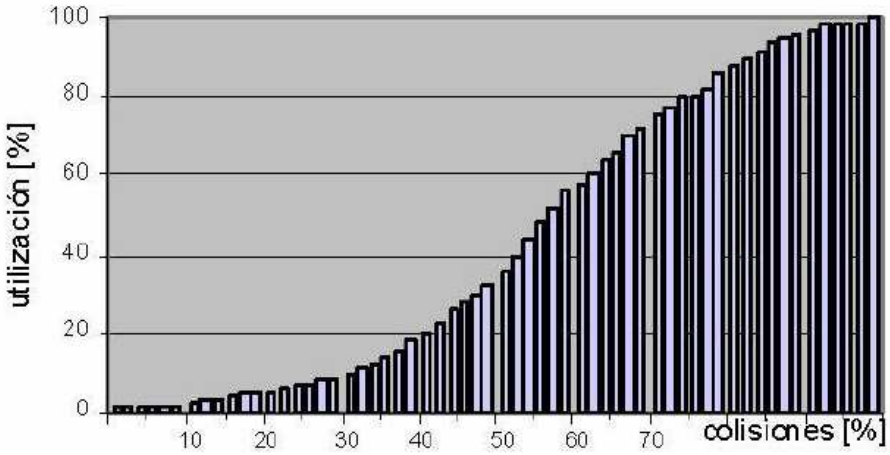


GRÁFICO 1.9 Utilización vs. Colisiones

1.2.4 DIRECCIONAMIENTO, SEÑALIZACIÓN, COMPRESIÓN Y TRANSMISIÓN DE LA VOZ EN REDES IP

Los procedimientos de señalización de voz sobre IP juegan un papel muy importante en la red, ya que establecen, mantienen, administran y proveen Calidad de Servicio (QoS) en la telefonía por paquetes. La señalización es utilizada para controlar y administrar la conexión entre dos puntos, ofreciendo funciones como supervisión, marcado, llamado y retorno de tonos de progreso. Los estándares normalmente utilizados en el señalamiento de VoIP se conocen como H.323 y los protocolos adicionales que se implementan son el Protocolo

de Iniciación de Sesión (SIP, por sus siglas en inglés), y el Protocolo de Control del Gateway de Transporte (MGCP, por sus siglas en inglés).

H.323 se refiere al conjunto de estándares empleados en la señalización y transmisión de voz en tiempo real, video y datos, dentro de una red de paquetes.

Su función consiste en especificar los componentes, protocolos y procedimientos que se implementan en el desarrollo de comunicaciones multimedia sobre redes IP.

Entre los componentes a los que se refiere el estándar, se encuentran terminales, "gateways", "gatekeepers" y unidades de control multi-punto.

Como veremos mas adelante, los "gateways" se utilizan para conectar dos redes diferentes, que pudieran ser una red H.323 y otra que no lo sea. Los "gatekeepers" constituyen el cerebro de una red H.323 y proveen servicios de direccionamiento, autorización y autenticación, administran también el ancho de banda de la red y en ocasiones, proveen servicios similares a los de un ruteador. Son estos dispositivos los que representan el punto central de todas las llamadas que se llevan a cabo en la red.

Por último, las unidades de control multi-punto proveen el soporte para mantener conferencias entre tres o más terminales H.323.

El Protocolo de Transporte en Tiempo Real (RTP, por sus siglas en inglés) forma parte del estándar H.323 y su objetivo es proveer servicios de audio y video en tiempo real de una terminal a otra. RTCP es el protocolo que acompaña a RTP, ofreciéndole funciones de control en el transporte. Es este último quien permite la retroalimentación en la calidad de la señal distribuida.

1.2.4.1 Direccionamiento en H.323[8]

Para hacer el direccionamiento, el estándar H.323 se basa en los protocolos RAS (Proceso de registro, Admisión y Estatus) y DNS (Servicio de Nombres de Dominio).

RAS es el protocolo que permite a una estación H.323 localizar otra estación H.323 a través del gatekeeper. Es decir sirve para el proceso de registro, control de admisión, control de ancho de banda, estado y desconexión. Los mensajes RAS viajan a través de un canal no confiable UDP. Las funciones definidas en el protocolo son las siguientes:

- *Descubrimiento del Gatekeeper*:[9] tiene lugar en los puntos finales H.323 para determinar el gatekeeper en el que dicho punto final debe registrarse, y puede ser de dos tipos:
  - Estático: el punto final sabe de antemano la dirección de transporte de su gatekeeper.
  - Dinámico: El punto final envía un mensaje GRQ (Petición de Gatekeeper) multicast en el que pregunta cual es el gatekeeper que le corresponde. Este mensaje será respondido por un GCF (Confirmación Gatekeeper) enviado por los gatekeepers disponibles.

  En general el punto final después de enviar un GRQ multicast, recibe la respuesta GCF de uno o varios gatekeepers de entre los cuales escogerá uno en el que desee registrarse. Los gatekeepers que no acepten el registro del punto final envían un mensaje de rechazo GRJ (Rechazo Gatekeeper). Si no se recibe ninguna contestación dentro de un intervalo de tiempo el punto final retransmite el mensaje GRQ.

---

[8] www.monografias.com, Voz sobre IP (VoIP), Pág. 4

[9] José M Huidrobo y David Roldan, "Integración de Voz y Datos", Cáp. 7, Pág. 185

- *Registro del punto final*: es un proceso empleado por los puntos finales para darse de alta en una zona e informar al gatekeeper de su dirección de transporte y del nombre con el que se identificará en dicha zona, para lo cual el punto final envía una solicitud de registro RRQ (Petición de Registro) a la dirección de transporte del canal RAS del gatekeeper, la misma que puede ser aceptada mediante el envío de un mensaje de confirmación RCF (Confirmación de Registro) ó rechazada mediante un mensaje RRJ (Rechazo de Registro). El gatekeeper debe asegurarse de que cada identificación utilizada se traduce como una dirección de transporte única. Un punto final puede cancelar su registro mediante el envió de un mensaje URQ (Cancelación de Registro) al gatekeeper, el cual confirma la cancelación de registro mediante un mensaje UCF (Confirmación de Cancelación); el proceso de cancelar el registro puede ser en forma inversa.

- *Localización del punto final*: es el proceso por el cual un gatekeeper o un punto final que disponen de la identificación de otro punto final obtienen la información de contacto del mismo.

- *Otras funciones de control*: Al canal RAS se lo puede emplear para llevar a cabo otros mecanismos de control, como el control de admisión, la restricción de la entrada de un punto final en una zona, el control de ancho de banda y el desenlace de control, que no es mas que dar de baja a un punto final de un gatekeeper y su zona correspondiente.

DNS es el servicio que traduce los nombres de Red en direcciones IP numéricas. Los nombres de Red son más manejables por los usuarios. Sin embargo, no son las direcciones reales en las que residen los recursos a los que se quiere acceder. Es por ese motivo por lo que se necesita traducir esa dirección en su equivalente numérico.

1.2.4.2 Señalización en H.323[10]

Existen dos tipos de señalización:

- *Señalización Directa:* los mensajes de señalización se intercambian directamente (sin la intervención del gatekeeper) entre los puntos finales utilizando las direcciones de transporte de señalización de llamadas CSTA (Dirección de Transporte de Señalización de Llamadas).

- *Señalización Indirecta:* se envía un mensaje utilizando la dirección de transporte del canal RAS, al gatekeeper y este encamina los mensajes de señalización.

Los protocolos que H.323 contempla para hacer las funciones de señalización son Q.931 (señalización inicial de llamadas), H.225 (control de la llamada) y H.245 (control para comunicaciones multimedia), los cuales se describen a continuación:

- Q.931 es la especificación de la capa 3, de la interfaz usuario – red de la RDSI para el control de la llamada básica. Se usa para establecer, mantener y finalizar una conexión lógica entre dos dispositivos dentro de una RDSI.

- H.225 se ocupa de las situaciones en las que la ruta de transmisión incluye alguna red de paquetes. En estas redes no se garantiza la Calidad de Servicio. Este protocolo describe como funciona el protocolo RAS y Q.931. El H.225 define como identificar cada tipo de codificador y discute algunos conflictos y redundancias entre RTCP y H.245.

[10] José M. Huidrobo y David Roldan, "Integración de Voz y Datos", Cáp. 7, Pág. 185

- H.245 es para señalización en comunicaciones multimedia. Hace posible el intercambio de mensajes extremo a extremo. Los mensajes de control H.245 se envían por canales de control.

  Es utilizado para comandos generales, indicaciones, control de flujo, gestión de canales lógicos, etc. Se usa en la interfaz GW-GW y GW-GK.

  El H.245 es una librería de mensajes con sintaxis del tipo ASN.1. En particular codifica los dígitos DTMF (Dual-Tone Multi Frequency) en el mensaje Indicación de Ingreso de Usuario (User Input Indication). Los mensajes H.245 pueden ser de dos tipos:

  - Mensajes de intercambio de características de los terminales: que es un proceso que emplean los terminales para intercambiar sus capacidades de transmisión y recepción con el extremo final.

  - Mensajes de señalización de canales lógicos: H.245 proporciona mensajes para abrir o cerrar un canal lógico, sea ésta una conferencia punto-punto o punto-multipunto.

1.2.4.3 Compresión de voz

"Los algoritmos de compresión usados en los ruteadores y en los gateways analizan un bloque de muestras digitalizadas, entregadas por el codificador de voz (voice CODEC). Estos bloques tienen una longitud variable que depende del codificador, por ejemplo el tamaño básico de un bloque del algoritmo G.729 es 10 ms, mientras que el tamaño básico de un bloque del algoritmo G.723.1 es 30ms. Un ejemplo de cómo funciona el sistema de compresión G.729 se indica en el GRÁFICO 1.10".[11]

[11] www.monografias.com, Voz sobre IP (VoIP), Pág. 6

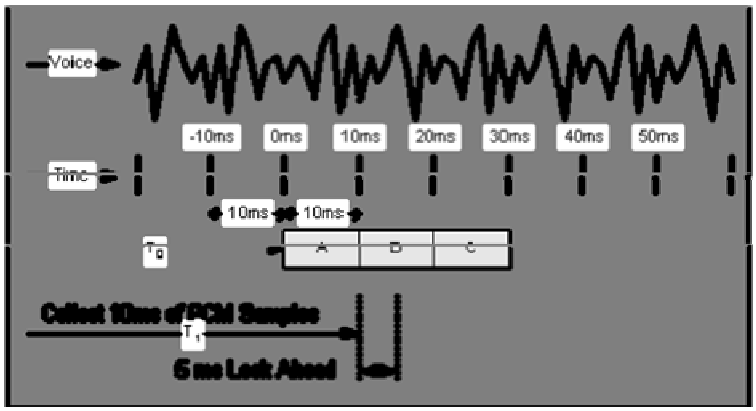


GRÁFICO 1.10 Funcionamiento de sistemas de compresión de voz

Para comprimir la voz, H.323 requiere los protocolos G.711 (modulación con PCM) y G.723 (doble velocidad para comunicaciones multimedia). Opcionalmente también tiene los protocolos G.728 (utiliza predicción lineal a 16 kbps), G.729 (predicción lineal a 8 kbps) y G.722 (codificación audio de 7 Khz.).

En general, la tabla 1.7 indica un cuadro comparativo de los códigos de la familia G.700:

TABLA 1.7 Códigos de la familia G.700

Tipo de CODEC	Velocidad de Transferencia de bits Kbps	Calidad de Voz	Demora
G.711	64	Muy Buena	Insignificante
G.726	40/32/24/16	De buena (40) a deficiente	Muy Baja
G.729	8	Buena	Baja
G.729A	8	Regular	Baja
G.723	6.4/5.3	De buena (6.4) a regular (5.3)	Alta
G.723.1	6.4/5.3	De buena (6.4) a regular (5.3)	Alta
G.728	16	Buena	Baja

1.2.4.3 Transmisión de voz

H.323 propone dos protocolos para posibilitar la transmisión de voz: UDP (User Datagram Protocol) y RTP (Real Time Protocol).

- UDP, es un protocolo de la capa de transporte (nivel 4 de OSI), no orientado a conexión.

- RTP proporciona servicios de recepción extremo a extremo cuando se trata de información que tiene que ser transmitida en tiempo real.

El GRÁFICO 1.11 indica la familia de protocolos para H.323:

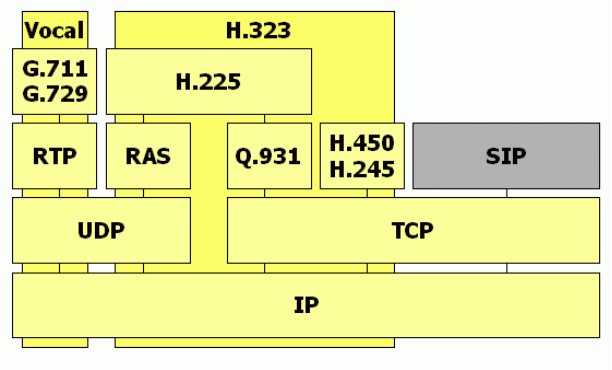


GRÁFICO 1.11 Familia de protocolos para H.323.

Las herramientas de las que se vale RTP para lograr transmisiones en tiempo real son el RTCP (RTP Control Protocol), que proporciona un feedback en relación a la calidad de distribución y la congestión, con esto, la empresa que ofrece el servicio puede monitorear la calidad y puede diagnosticar los problemas que pueda presentar la red.

Además, RTCP sincroniza el audio y el video, conoce el número de usuarios presentes en una conferencia y con esto calcula la tasa a la cual deben ser enviados los paquetes, todas estas opciones son obligatorias cuando RTP se usa en entornos multicast IP.

Pero existe otra aplicación opcional y es una administración de sesiones, con bajo manejo de información de control para aquellas aplicaciones donde existe un uso masivo de usuarios entrando y saliendo constantemente.

Para la compresión, RTP usa una aplicación llamada “vocoder”, pudiendo reducir de 64 kbps hasta a 8 kbps la tasa para digitalización y compresión de voz, produciendo un desmejoramiento en la calidad de la voz poco perceptible, además de esto usa H.323, G.729 y otros protocolos más para transmisiones en tiempo real.

RTP es capaz de correr sobre protocolos WAN de alta velocidad como ATM sin ningún problema, también puede correr en redes asimétricas como ADSL, cable-modem o por enlace satelital, pero cumpliendo con ciertas características de ancho de banda para ambas direcciones y uso exclusivo para la aplicación RTP (Real Time Transport Protocol).

A pesar de que TCP es un protocolo de transporte de información “pesada”, y que eventualmente podría llegar a transportar video y voz, este y otros protocolos como XTP son inapropiados por tres razones básicas:

- El hecho de que ante la pérdida de paquetes, este tipo de protocolos emplean retransmisión de paquetes. TCP no soporta multicast.

- El control de congestión de TCP hace reducir la ventana de transmisión cuando detecta pérdida de paquetes, y el audio y el video son aplicaciones cuya tasa de transferencia no permite disminuciones de este tipo en la ventana de transmisión.

- Adicionalmente otra desventaja es que los encabezados de estos protocolos son más largos que los de RTP.

1.3 COMPONENTES EN UNA RED H.323 PARA VoIP[12]

Actualmente podemos partir de una serie de elementos ya disponibles en el mercado y que, según diferentes diseños, nos permitirán construir las aplicaciones VoIP. Estos elementos son:

- Teléfonos IP.
- Adaptadores para PC.
- Hubs Telefónicos.
- Gateways (pasarelas RTC / IP).
- Gatekeeper.
- Unidades de audio-conferencia múltiple. (MCU Voz)

Los componentes de una red H.323, son los terminales, los Gateways, los Gatekeepers y las MCU, que se indican en el GRÁFICO 1.12

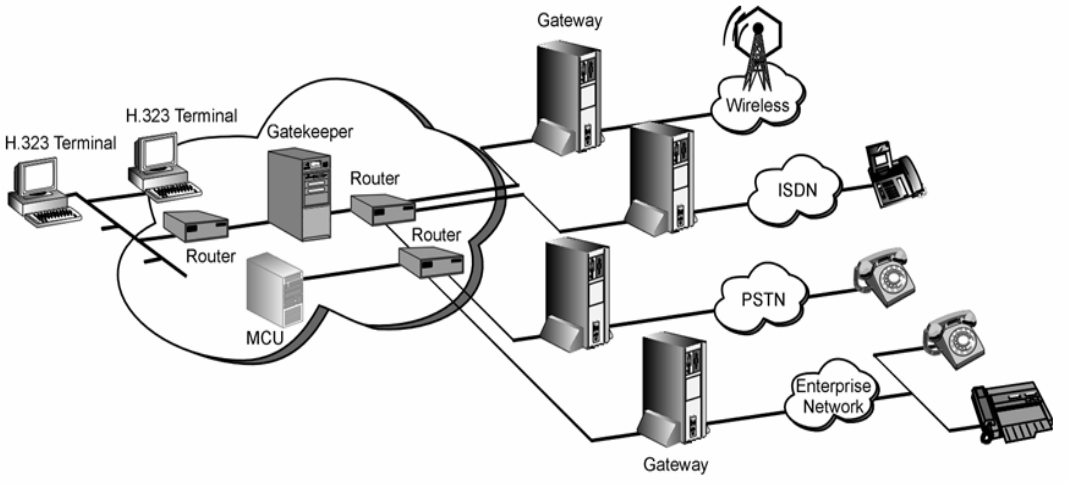


GRÁFICO 1.12 Componentes de una Red H.323

[12] José M. Huidrobo y David Roldan, "Integración de Voz y Datos", Cáp. 7, Pág. 183

1.3.1 TERMINALES

Como terminales, debemos entender el equivalente a los teléfonos actuales.

Un Terminal H.323, es un extremo de la red que proporciona comunicaciones bidireccionales en tiempo real con otro Terminal H.323, Gateway o Unidad de Control Multipunto (MCU). Esta comunicación consta de señales de control, indicaciones, audio, imagen en color en movimiento y /o datos entre los dos terminales.

Conforme a la especificación, un Terminal H.323 puede proporcionar sólo voz, voz y datos, voz y vídeo, o voz, datos y vídeo, entre los cuales constan teléfonos IP, teléfonos Analógicos, soft phone, entre otros.

El funcionamiento de todo Terminal debe incluir el tratamiento necesario de la señal para su envío por la red de datos. Deben realizar la captación, digitalización, y compresión de la señal.

Existen principalmente dos tendencias en este tipo de elementos, terminales hardware y terminales software:

- En los Terminales hardware tanto la apariencia, como la funcionalidad de cara al usuario es igual a los teléfonos actuales. Ya existen en el mercado terminales que se conectan directamente a la red local (LAN), conocidos como teléfonos IP.

- Por otro lado los terminales software, no son más que programas ejecutándose en nuestro computador personal. Las capacidades del software pueden ser muy superiores a las aportadas por una solución hardware.

1.3.2 GATEWAY Ó PASARELA DE VOZ

Un gateway ó pasarela se encarga de traducir los protocolos de establecimiento y liberación de llamadas y de la conversión de formatos de la información entre diferentes tipos de redes, así como de transferir la información entre redes H.323 y redes no H.323.

Los terminales se comunican con los gateways mediante el protocolo de señalización H.245 y el protocolo de señalización H.225. El Gateway convierte esos protocolos a los respectivos de la red no H.323 de forma totalmente transparente. La conversión de los formatos de video y audio no son necesarias siempre y cuando los terminales de ambos extremos encuentren un modo de comunicación común.

El gateway es un elemento esencial en la mayoría de las redes, pues su misión es la de enlazar la red VoIP con la PSTN o RDSI. Podemos considerar al gateway como una caja negra que por un lado tiene una interfaz LAN y por otro dispone de una o varias de las siguientes interfaces:

- FXO. Para conexión a extensiones de centrales pequeñas ó a la red telefónica básica.
- FXS. Para conexión a enlaces de centrales pequeñas o a teléfonos analógicos.
- E&M. Para conexión específica a centrales pequeñas.
- BRI. Acceso básico RDSI (2B+D)
- PRI. Acceso primario RDSI (30B+D)
- G703/G.704. (E&M digital) Conexión especifica a centrales pequeñas a 2 Mbps.

Los distintos elementos de una red H.323 pueden residir en plataformas físicas separada, o nos podemos encontrar con varios elementos conviviendo en la misma plataforma. De este modo es habitual encontrar juntos Gatekeeper y

Gateway. También es común implementar las funciones de Gateway en los ruteadores.

1.3.3 GATEKEEPER

El gatekeeper (GK) es una entidad que proporciona la traducción de direcciones y el control de acceso a la red de los terminales H.323, gateways y MCUs. El GK puede también ofrecer otros servicios a los terminales, gateways y MCUs, tales como gestión del ancho de banda y localización de los gateways o pasarelas.

El Gatekeeper realiza algunas funciones de control de llamadas, que preservan la integridad de la red corporativa de datos, las cuales se detallan a continuación:

- La primera es la traducción de direcciones. Las llamadas originadas en el seno de una red H.323 emplean un identificador para referirse al Terminal destino, mientras que las llamadas originadas fuera de la misma y recibidas por una pasarela utilizan un número de teléfono, el gatekeeper debe entonces traducir dicho número en la dirección de red del Terminal destino. De esta forma el punto final destino puede ser alcanzado utilizando la dirección de red H.323.

- La segunda es la Gestión del ancho de banda, que se implementa a través de mensajes RAS, de petición de ancho de banda (BRQ), de confirmación (BCF) y de rechazo (BRJ), fijando el número de conferencias que pueden estar dándose simultáneamente en la LAN y rechazando las nuevas peticiones por encima del nivel establecido, de manera tal que se garantice ancho de banda suficiente para las aplicaciones de datos sobre la LAN.

- La tercera es el Control de admisión. El gatekeeper controla la admisión de los puntos finales dentro de la red H.323. Para ello emplea mensajes RAS de petición de admisión (ARQ), de confirmación (ACF) y de rechazo (ARJ).

- La cuarta es la Gestión de zona, lo cual indica que el Gatekeeper proporciona todas las funciones anteriores para los terminales, Gateways y MCUs, que están registrados dentro de la denominada Zona de control H.323.

Además el gatekeeper puede ofrecer otras funciones como:

- Control de señalización: el GK puede encaminar mensajes de señalización entre puntos finales H.323, donde desempeña la función de monitor de llamadas para mejorar el control de las llamadas en la red.

- Autorización de llamadas: cuando un punto final envía un mensaje de señalización a un gatekeeper, este puede aceptar o rechazar la llamada.

- Gestión de la llamada: el gatekeeper mantiene información sobre las llamadas H.323 activas de modo que puede utilizar dicha información para la gestión del ancho de banda o desviando las llamadas a diferentes puntos finales con el fin de balancear la carga.

1.3.4 MCU (UNIDAD DE CONTROL MULTIPUNTO)

La Unidad de Control Multipunto está diseñada para soportar la conferencia entre tres o más puntos, bajo el estándar H.323, llevando la negociación entre terminales para determinar las capacidades comunes para el proceso de audio y vídeo y controlar la multidifusión.

La comunicación bajo H.323, contempla las señales de audio y vídeo. La señal de audio se digitaliza y se comprime bajo uno de los algoritmos soportados, tales como el G.711 o G.723, y la señal de vídeo (opcional) se trata con la norma H.261 o H.263. Los datos (opcional) se manejan bajo el estándar T.120 que permite la compartición de aplicaciones en conferencias punto a punto y multipunto.

Está formada por dos componentes lógicos:

- Controlador Multipunto (MC): encargado de la coordinación de control de llamadas para soportar conferencias entre tres o más puntos finales en una conferencia multipunto. Cada MCU dispone de un MC obligatoriamente mientras en los terminales, gateways y gatekeepers es opcional.

- Procesador Multipunto (MP): cuya misión es la mezcla de las señales de audio, video y/o datos procedentes de los puntos finales implicados en la multiconferencia. Es opcional en todos los componentes de la red H.323 excepto en los terminales.

1.3.5 EJEMPLOS DE COMUNICACIÓN EN VoIP CON EL ESTANDAR H.323[13]

A continuación se describe algunos ejemplos de establecimiento de llamadas de voz según el estándar H.323.

[13] Rafael estepa Alonso, Voz sobre IP (VoIP), Pág. 20

1.3.5.1 Llamada de voz entre dos terminales H.323: A llama a B

Primera Fase: Inicio de Llamada, se establece el identificador de la llamada, los usuarios de origen y destino, así como también el tipo de conexión, como se indica en el GRÁFICO 1.13

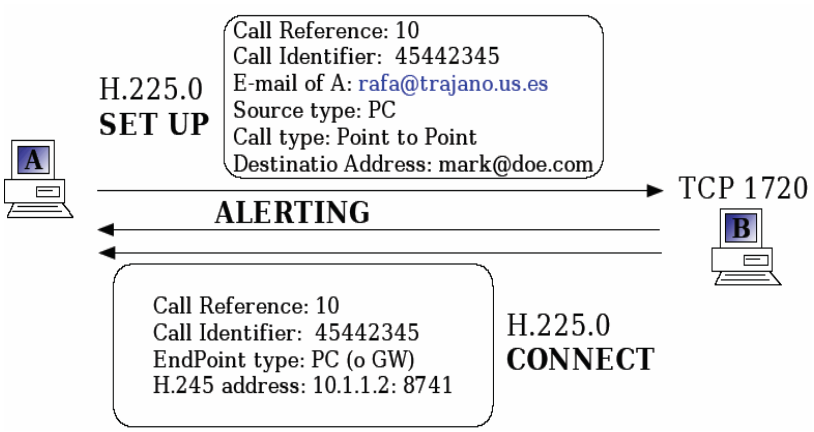


GRÁFICO 1.13 Inicio de la llamada.

Segunda Fase: Establecimiento del canal de control, que se negocia entre los dos terminales, como se indica en el GRÁFICO 1.14

- Definido en H.245. Permite negociación de capacidades

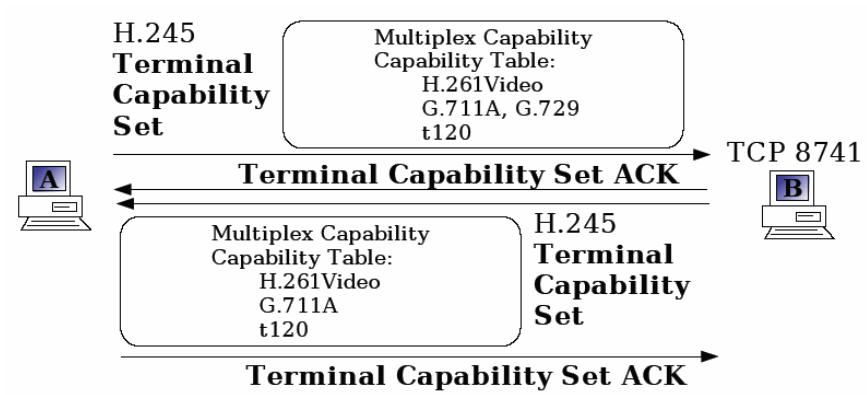


GRÁFICO 1.14 Establecimiento del canal de control

Tercera Fase: Comienzo de la llamada, se indica en el GRÁFICO 1.15.

- Apertura de canales lógicos unidireccionales para medios
- Canales T.120 para datos bidireccionales

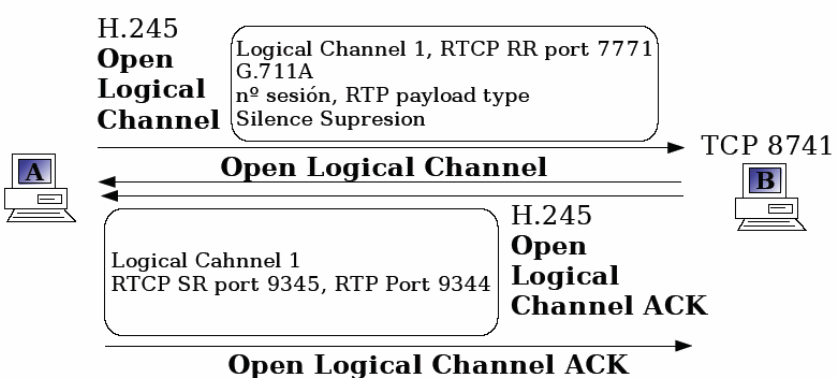


GRÁFICO 1.15 Comienzo de la llamada

Cuarta Fase: Diálogo, indicado en el GRÁFICO 1.16

- Apertura de canales lógicos unidireccionales para medios
- Puede haber varios canales lógicos abierto, sincronizados
- Sesiones: Audio, Video y Datos

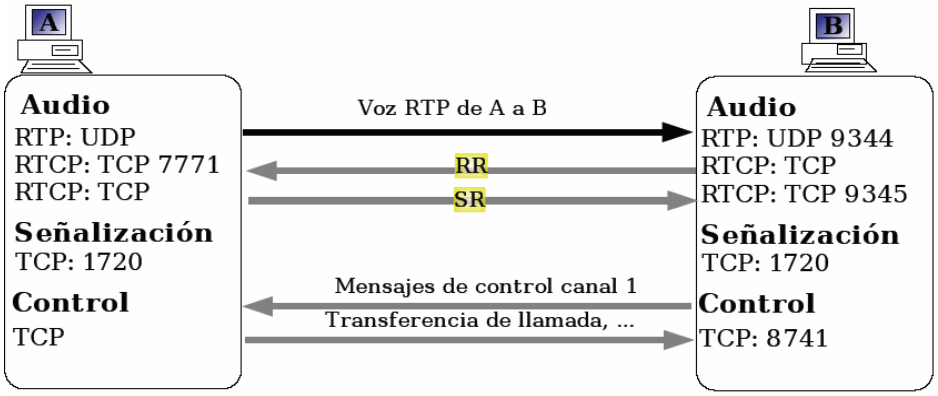


GRÁFICO 1.16 Diálogo

Quinta Fase: Finalización de la llamada.

- Secuencia lógica:
    - Enviar: Close Logical Channel por cada canal abierto
    - Recibir ACK de los anteriores
    - Enviar H.245 End Session Command
    - Recibir lo mismo para cerrar el canal H.245
    - Enviar: H.225.0 Release Complete

1.3.5.2 Llamada a un teléfono público desde Internet

Primera Fase: ¿Dónde está el GateKeeper?, este proceso se detalla a continuación:

- Configuración Manual
- Descubrimiento automático: Gatekeeper Request (GRQ)
    - Dirección multicast: 224.0.1.41
    - Indica tipo de terminal, alias, direc:port de la respuesta
- Respuesta: Gatekeeper Confirm (GCF)
    - Nombre, dirección y puerto unicast para canal RAS
- Registro en un GateKeeper (RRQ): direc:port para señalización
    - UDP 1719. El GK responde con (RCF). Identificador Único de Terminal.

Fase 2: Solicitar permiso para una nueva llamada, que realiza la elección del tipo de llamada, estimación de ancho de banda, y envío de mensajes que confirmen el estado activo de la llamada (mensajes Keep Alive). Como se indica en el GRÁFICO 1.17.

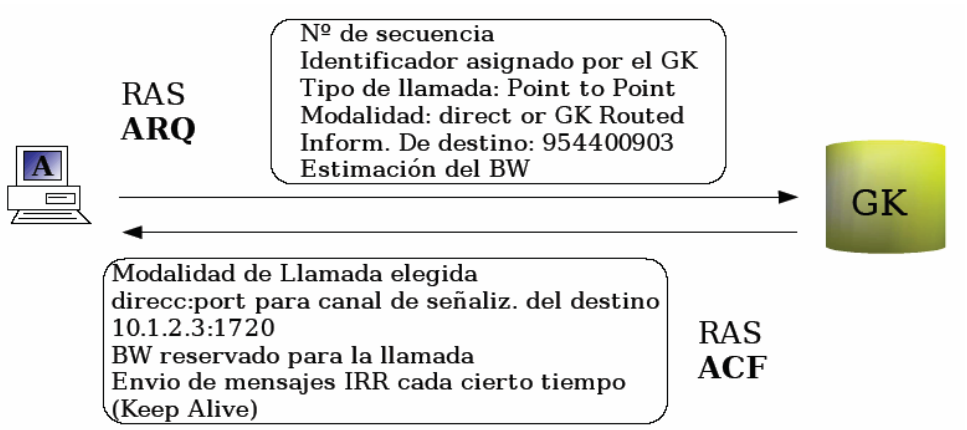


GRÁFICO 1.17 Permisos de nueva llamada.

Fase 3: Señalización de la llamada. GRÁFICO 1.18

- El GW puede solicitar al GK permiso para aceptar la llamada
  - ARQ/ACF, donde ARQ contiene el alias y la direc:port para señalización tanto del llamante como del llamado
- El GW conoce el destino mediante la primitiva SET UP=>Llamada RDSI

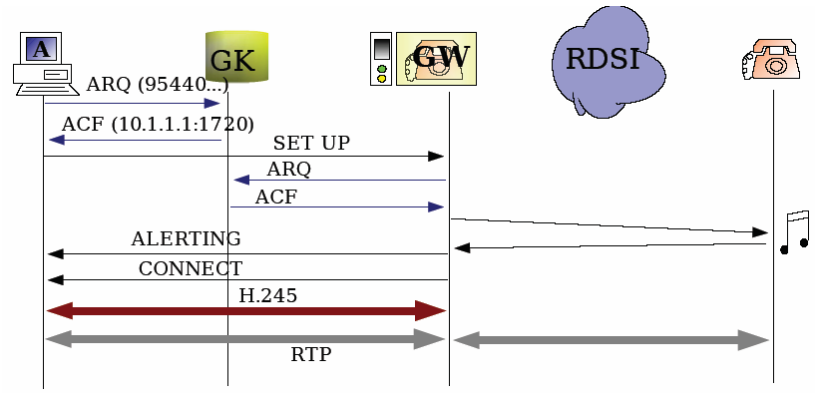


GRÁFICO 1.18 Señalización de llamada

Fase 4: Finalización de la llamada. GRÁFICO 1.19

- El gateway (GW), debe cerrar los canales lógicos

- El gatekeeper (GK), debe enviar una solicitud de desenganche (Desengage Request, DRQ), para que el GK libere el ancho de banda reservado
- Si un GK envía solicitud de desenganche (DRQ) a un terminal, éste debe mandar un EndSessionCom.

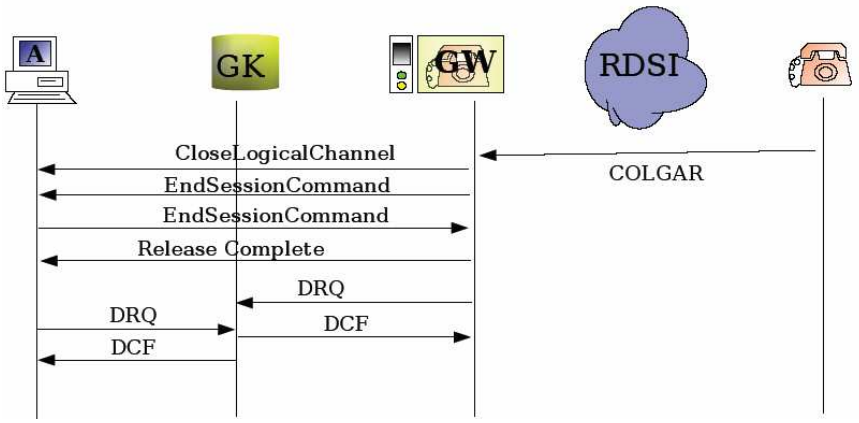


GRÁFICO 1.19 Finalización de llamada.

CAPÍTULO 2

EL PROTOCOLO DIFFSERV

2.1 INTRODUCCIÓN

Durante los últimos años, han surgido varios mecanismos para ofrecer redes de Calidad de Servicio (QoS). El principal objetivo de estos mecanismos es proporcionar un "servicio" de redes mejorado para los diversos tipos de aplicaciones en los extremos de la red.

Los administradores agregan continuamente nuevos recursos para tratar de responder al ritmo de la creciente demanda. El uso creciente de un nuevo tipo de aplicaciones multimedia, que demandan más recursos agudiza esta situación. Los mecanismos de QoS proporcionan un conjunto de herramientas que el administrador de redes puede utilizar para administrar el uso de recursos de red de una forma controlada y eficaz. Como resultado, se obtendrá un mejor servicio para las aplicaciones y para los usuarios. En resumen, QoS, significa disponibilidad de la red y eficiencia en la transmisión, que ayuda a mejorar el servicio a los usuarios de la red, al mismo tiempo que reduce los costos de ofrecer dichos servicios.

Las redes están construidas mediante la unión de dispositivos de red, tales como conmutadores y ruteadores. Estos dispositivos intercambian el tráfico entre ellos mediante interfaces, la capacidad de una interfaz para enviar tráfico constituye un recurso de red fundamental. Los mecanismos de QoS funcionan al establecer preferencias en la asignación de este recurso en favor de cierto tráfico.

Nuestro estudio está basado en la aplicación de Diffserv, que es uno de los principales métodos por agregación de tráfico y que lo único que hace es agrupar varios flujos de tráfico en diferentes clases.

2.2 MODELO DE LA ARQUITECTURA DIFFERENTIATED SERVICES[14]

La Arquitectura de Servicios Diferenciados (Diffserv) está basado en un modelo simple de tratamiento del tráfico, utilizado para grandes redes enrutadas. La sofisticada clasificación, marcado de los paquetes, política y operaciones de acondicionamiento necesitan sólo ser implementadas en los elementos de frontera de la red. El marcado de paquetes se realiza mediante la asignación de un código específico (DSCP – Diffserv CodePoint), que es todo lo que se necesita para identificar a cada clase de tráfico.

La clase de tráfico es la agregación de todos los flujos bajo el mismo criterio de clasificación. El DSCP indica un comportamiento específico que un paquete debe de recibir en cada enrutador. La diferenciación de servicios se logra mediante la definición de comportamientos específicos para cada clase de tráfico entre dispositivos de interconexión, hecho conocido como PHB (Per Hop Behavior) y que detallaremos de mejor manera mas adelante.

Esta arquitectura logra escalabilidad al implementar funciones de clasificación y condicionamiento sólo en los nodos del borde de la red, y aplicando conductas por salto a los agregados del tráfico (BA) que han sido apropiadamente marcados usando el campo DS en las cabeceras de IPv4 o IPv6.

Esta arquitectura sólo provee servicio diferenciado en una dirección del flujo de tráfico y es por ende asimétrica, ya que quien nos brinda el Servicio Diferenciado es el ISP.

[14] RFC 2475, “Arquitectura para Servicios Diferenciados”, Pág. 12

2.2.1 DOMINIO DE LOS SERVICIOS DIFERENCIADOS (DOMINIO DS)

Un dominio DS es un conjunto de nodos DS que operan con una política de aprovisionamiento de servicios común y con un conjunto de grupos PHB implementados en cada nodo. Está formado por nodos DS de frontera que clasifican y posiblemente condicionen el tráfico entrante para asegurarse que los paquetes que transitan al dominio estén apropiadamente marcados para seleccionar un PHB de los grupos implementados dentro del dominio. Los nodos seleccionan el comportamiento de envío basándose en su código DS, y lo hacen asociando éste valor a unos de los PHB soportados.

Los servicios proporcionados a través de un domino DS, se definen en un acuerdo de nivel de servicio (SLA). La inclusión de nodos que no soportan Diffserv dentro de un dominio DS puede resultar en un desempeño impredecible y puede impedir la habilidad de satisfacer el acuerdo del nivel de servicio (SLA), como se indica en el GRÁFICO 2.1.

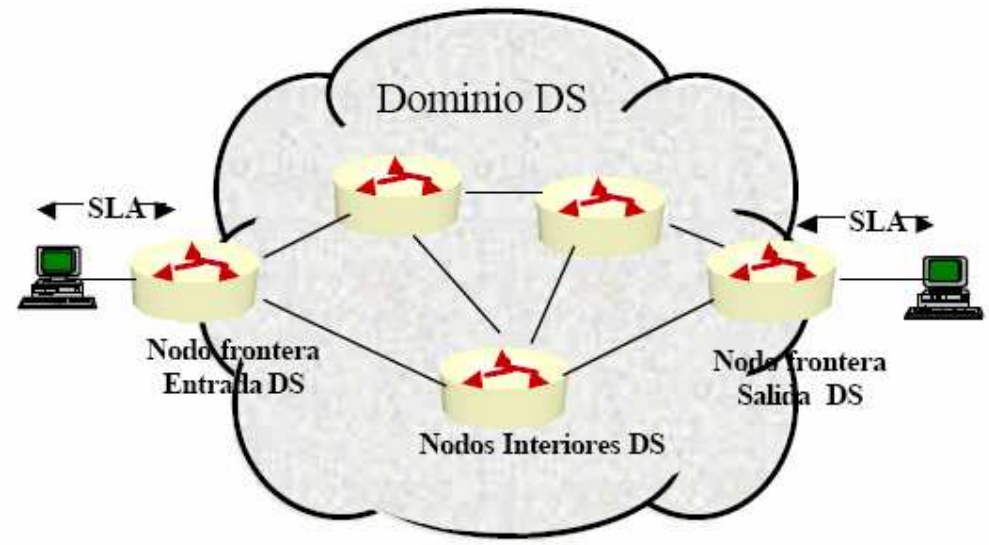


GRÁFICO 2.1 Dominio Diffserv.

Un dominio DS consiste en general de una o más redes bajo la misma administración.

Un Acuerdo de Nivel de Servicio (Service Level Agreement, SLA), es un contrato de servicios entre un proveedor de servicios y su cliente, como se muestra en el GRÁFICO 2.2, el cual define las responsabilidades del proveedor en términos del nivel de funcionamiento de la red (rendimiento, tasa de pérdidas, retrasos, variaciones) y la disponibilidad temporal, el método de medida, las consecuencias cuando los niveles de servicio no se consiguen o si los niveles de tráfico definidos son superados por el cliente, así como el precio de todos estos servicios. Evidentemente, y suele ser lo más común, el SLA puede incluir reglas de condicionamiento del tráfico.

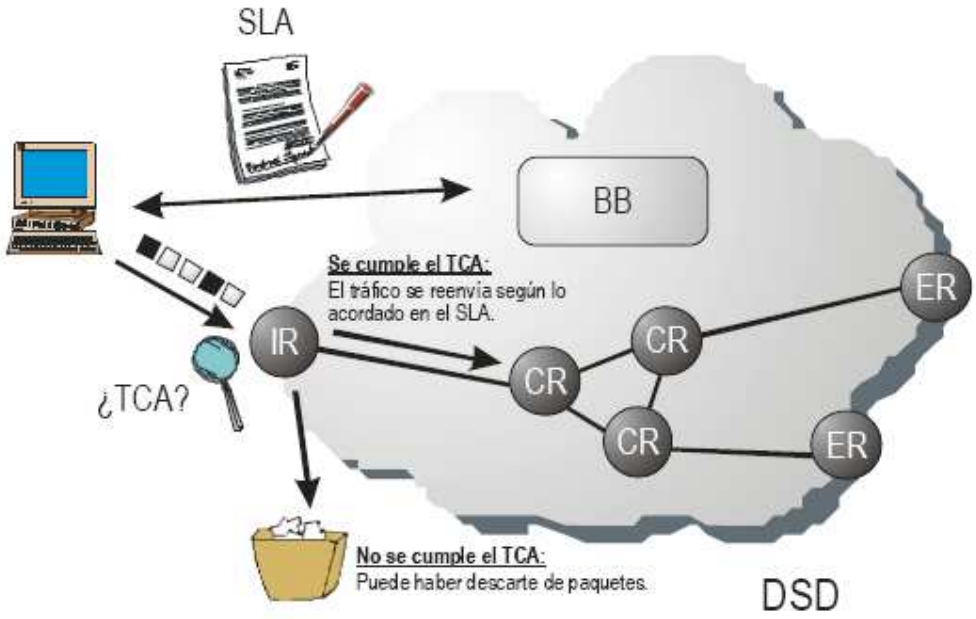


GRÁFICO 2.2 Diffserv – Service Level Agreement

2.2.1.1 Nodos DS de frontera e interiores

Los nodos DS de frontera interconectan el dominio DS con otros dominios que pueden o no soportar Diffserv. Los nodos interiores solo conectan con otros nodos interiores o de frontera dentro del mismo dominio DS.

Ambos tipos de nodos deben ser capaces de aplicar el PHB apropiado a los paquetes basándose en el código DS, sino, puede resultar en un comportamiento impredecible.

Los nodos DS de frontera puede que sean requeridos para desempeñar funciones de acondicionamiento de tráfico tal como se define en un Acuerdo de Acondicionamiento de Tráfico (TCA) entre su dominio DS y el dominio contiguo al cual conectan, tal como se indicó en el GRÁFICO 2.1.

Los nodos DS interiores deben poder desempeñar otras funciones tales como el re-marcado de códigos DS.

Los nodos DS interiores desempeñan funciones más complejas de clasificación y acondicionamiento de tráfico que son análogas a las de los nodos DS de frontera.

La arquitectura de un nodo DS interior se muestra en el GRÁFICO 2.3:

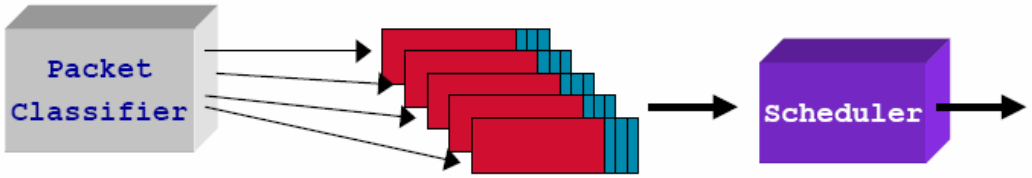


GRÁFICO 2.3 Arquitectura de nodos DS Interiores.

La clasificación de los paquetes se realiza de dos maneras dependiendo del tipo de cliente que esté conectado a los nodos de frontera. Cuando el cliente es una red de una organización, la clasificación o determinación de que PHB se le debe aplicar al paquete dentro de la red, se realiza tomando en cuenta los siguientes seis campos del paquete IP:

- Dirección IP de la fuente.
- Dirección IP del destino.
- Protocolo de transporte (TCP/UDP)
- Campo DiffServ (DS) en el paquete de llegada.
- Puerto del origen en el encabezado de TCP.
- Puerto destino en el encabezado de TCP.

Las reglas que mapean un paquete a un PHB determinado se llaman Reglas de Clasificación (classification rules). Las reglas de clasificación no necesariamente tienen que especificar los 6 campos. La regla puede ser una combinación de dos o más campos como por ejemplo, el protocolo de transporte y la dirección del puerto destino.

2.2.1.2 Nodos de Ingreso y Egreso

Los nodos de frontera actúan tanto como nodos DS de ingreso y egreso para el tráfico de distintas direcciones.

Un nodo DS de ingreso es responsable por asegurar que el tráfico entrante al dominio DS cumpla con el Acuerdo de Acondicionamiento de Tráfico (TCA) entre éste y el dominio contiguo al cual está conectado el nodo de ingreso. Un nodo DS de egreso puede desempeñar funciones de acondicionamiento de tráfico para reenviarlo al siguiente dominio directamente conectado, dependiendo de los detalles del TCA entre los dos dominios.

2.2.2 REGIÓN DE SERVICIOS DIFERENCIADOS (REGIÓN DS)

Una Región de Servicios Diferenciados es un conjunto de uno o más dominios DS contiguos, que se indica en el GRÁFICO 2.4.

Las Regiones DS son capaces de soportar diferentes servicios a lo largo de una ruta la cual atraviesa los dominios dentro de la región.

El dominio DS en una región DS, puede soportar diferentes grupos de PHB internamente. Sin embargo, para permitir servicios que se expanden a través de los dominios, los dominios DS contiguos deben cada uno establecer un Acuerdo de Nivel de Servicio (SLA), entre ellos que define (explicita ó implícitamente) cierta TCA, para especificar cómo el transito del tráfico de un dominio DS a otro es condicionado en la frontera entre los dos dominios DS.

Es posible que algunos dominios dentro de una región puedan adoptar una política de provisión de servicios común y soportar un conjunto de grupos de PHB similares y de esta manera eliminar la necesidad de acondicionar el tráfico entre los dominios.

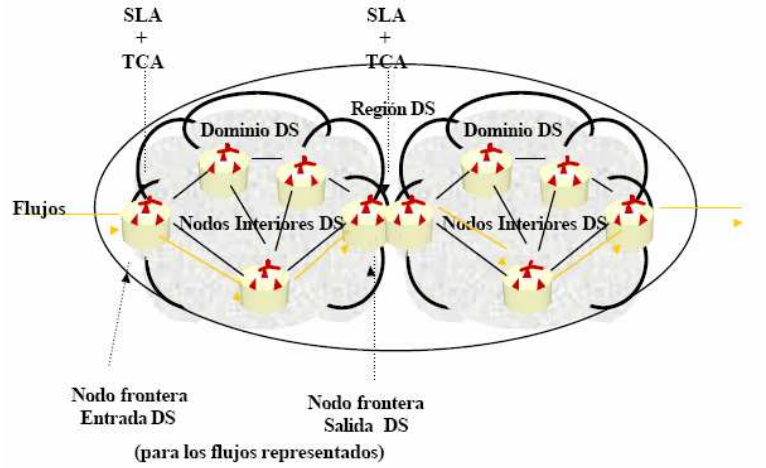


GRÁFICO 2.4 Región Diffserv.

2.2.3 CLASIFICACIÓN Y ACONDICIONAMIENTO DEL TRÁFICO EN UN DOMINIO DS

Los SLA pueden especificar la clasificación de paquetes y las reglas de re-marcado y también pueden especificar los perfiles de tráfico y acciones para

las cadenas de tráfico los cuales están dentro o fuera del perfil (in-out profile). El TCA entre los dominios proviene de éste SLA.

La política de clasificación de paquetes identifica el subconjunto de tráfico que puede llegar a recibir un servicio diferenciado, al ser condicionado y/o mapeado a uno o más comportamientos agregados dentro del dominio DS.

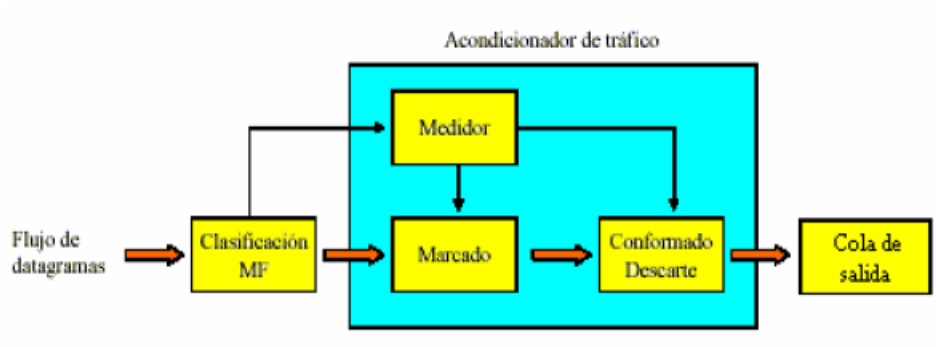


GRÁFICO 2.5 Vista lógica de un Clasificador y un Acondicionador de paquetes.

El condicionamiento de tráfico desempeña la medición, conformación, política y/o remarcado, tal como se muestra en el GRÁFICO 2.5, para asegurarse que el tráfico entrante al dominio DS respeta las reglas especificadas en el TCA, en concordancia con las políticas de provisión de servicios del dominio. El alcance del acondicionamiento de tráfico requerido depende del servicio específico ofrecido.

2.2.3.1 Clasificadores

Los clasificadores de paquetes seleccionan paquetes en una cadena de tráfico basándose en el contenido de alguna porción de la cabecera del paquete.

Se definen dos tipos de clasificadores.

- El Clasificador BA (Behavior Agregate), clasifica los paquetes basándose únicamente en el código DS.

- El MF (Multi-Archivo), clasifica a los paquetes basándose en el valor de una o más campos de la cabecera del paquete, como puede ser la dirección origen, la dirección destino, número de puertos de origen y de destino e incluso el campo DS.

Los clasificadores son usados para distinguir los paquetes y hacer que coincidan con una regla específica y enrutarlos hacia un elemento acondicionador de tráfico para de esta manera darle un procesamiento adicional. Además los clasificadores deben estar configurados de acuerdo al TCA.

2.2.3.2 Perfiles de Tráfico

Especifica las propiedades temporales de una corriente de tráfico seleccionada por el clasificador. Provee de reglas para determinar si un paquete está dentro o fuera del perfil.
Diferentes acciones de condicionamiento pueden ser aplicadas a los paquetes dentro y fuera del perfil. Los paquetes dentro del perfil pueden ser mandados sin ningún otro procesamiento o marcado o remarcado. Los paquetes fuera de perfil pueden ser encolados hasta que estén dentro del perfil (conformados), desechados (política) o remarcados con un código nuevo (re-marcado).

Hay que hacer notar que el perfil de tráfico es un componente opcional de un TCA.

2.2.3.3 Acondicionador de Tráfico

Puede contener los siguientes elementos: medidor, marcador, conformador y despachador. Una cadena de tráfico es seleccionada por un clasificador. Un medidor es usado para medir la corriente de tráfico en base a un perfil de tráfico. El estado del medidor respecto a un paquete en particular puede ser usado para afectar el marcado, despacho, o acción de conformación.

Cuando los paquetes salen del acondicionador de tráfico de un nodo DS frontera, el código DS de cada paquete debe setearse a un valor apropiado, según los requerimientos especificados por el usuario para ese tipo de tráfico.

2.2.3.3.1 Medidor

Miden las propiedades temporales de la cadena de paquetes seleccionada por el clasificador en base a un perfil de tráfico especificado en el TCA. Pasa información de estado a otras funciones de condicionamiento para tomar cierta acción para cada paquete tanto dentro como fuera del perfil.

2.2.3.3.2 Marcador

Setean el campo DS con un código particular, agregando el paquete marcado a un Behavior Agregate (BA). Puede que marque todos los paquetes que son dirigidos a él con un código particular o puede estar configurado para marcar un paquete a un código de un grupo de códigos usados para seleccionar un PHB en un grupo PHB. Cuando el marcador cambia el código en un paquete, se dice que ha “remarcado” el paquete.

2.2.3.3.3 Conformador

Retardan uno o todos los paquetes de una cadena de tráfico de manera de que la cadena cumpla con el perfil de tráfico estipulado. Usualmente tiene un buffer de tamaño finito y los paquetes pueden ser descartados si no hay suficiente espacio de buffer para aguantar a los paquetes retrasados.

2.2.3.3.4 Despachador

Descarta algunos o todos los paquetes en una cadena de tráfico, de manera que cumpla con el perfil de tráfico estipulado. Este proceso es conocido como "política". Un Despachador puede ser implementado como un caso especial de un Conformador, si ponemos el tamaño del buffer del conformador igual a 0 paquetes, lo que quiere decir que no se almacenará ningún paquete, pues todos los paquetes que se conformen se enviarán inmediatamente.

2.2.3.4 Ubicación de los acondicionadores de tráfico o clasificadores MF

Los acondicionadores de tráfico están usualmente localizados dentro de los nodos DS de frontera de ingreso y egreso, pero pueden ser ubicados en los nodos dentro de un dominio DS o dentro de un dominio que no sea Diffserv.

2.2.3.4.1 Dentro del dominio origen.

Se lo define como el dominio que contiene el nodo que origina el tráfico que recibe un servicio particular. El tráfico originado del dominio fuente a través de una frontera puede ser marcado por las fuentes de tráfico directamente o por medio de nodos intermediarios antes de que dejen el domino origen. Esto se conoce como "pre-marcado".

Por ejemplo, supongamos una compañía que tiene la política de que los paquetes de video, deben tener prioridad alta. El usuario puede marcar el campo DS de todos los paquetes de salida con un código DS que indique ese grado de prioridad alto, o alternativamente, el ruteador del primer salto, que está directamente conectado al host que genera estos paquetes, puede clasificar el tráfico y marcar los paquetes de video con el código DS correcto.

Hay ciertas ventajas en el hecho de marcar paquetes cerca del origen de tráfico. Primero, el origen del tráfico puede más fácilmente tomar en cuenta las preferencias de las aplicaciones en el momento de decidir qué paquetes deben recibir mejor tratamiento de envío. Además, la clasificación de paquetes es más simple, antes que el tráfico haya sido agregado a otros paquetes provenientes de otras fuentes, ya que el número de reglas de clasificación que deben ser aplicadas dentro de un nodo único es reducido.

El nodo frontera del dominio origen debe también monitorear la concordancia con el TCA, así como vigilar el conformado o re-marcado de paquetes según sea necesario.

2.2.3.4.2 En la Frontera de un Dominio DS

El tráfico puede ser clasificado, marcado y en otros casos condicionados en cualquiera de los enlaces de frontera entre dos dominios DS.

El Acuerdo de Nivel de Servicio (SLA), entre dominios, debe especificar cuál de ellos, tiene la responsabilidad de mapear el tráfico a agregados de comportamiento (BA) y acondicionar esos agregados en concordancia con el apropiado TCA.

Sin embargo, un nodo de ingreso debe asumir que el tráfico entrante puede no concordar con el TCA y debe estar preparado para reforzar el TCA de acuerdo a la política local.

Cuando los paquetes son pre-marcados y condicionados en un dominio superior, una clasificación y condicionamiento menores deben ser soportados en el dominio DS inferior.

Si un nodo de ingreso esta conectado a un dominio superior que no soporta Diffserv (DS), el nodo de ingreso DS debe poder cumplir con todas las funciones de condicionamiento de tráfico necesarias para que el tráfico entrante al dominio superior pueda ser manejado adecuadamente.

2.2.3.4.3 En Dominios que no soportan Diffserv.

El tráfico de origen o los nodos intermedios en un Dominio que no soporta Diffserv, pueden emplear acondicionadores de tráfico para pre-marcar los paquetes antes de que estos alcancen el ingreso del siguiente dominio DS.

2.2.3.4.4 En nodos DS interiores.

Aunque la arquitectura básica asume que la clasificación compleja y las funciones de acondicionamiento de tráfico están ubicadas únicamente en los nodos DS de frontera en el ingreso y egreso de la red, el desempeño de estas funciones en el interior de la red no esta prohibido.

2.3 DEFINICIÓN DEL CAMPO DIFFERENTIATED SERVICES (DIFFSERV)[15]

Un campo de cabecera, llamado DS, es definido para los Servicios Diferenciados, el cual sustituye las definiciones existentes del octeto tipo de servicio (ToS) de IP versión 4 (IPv4) y del octeto Clase de Tráfico de IP versión 6 (IPv6), como se muestra en los GRÁFICOS 2.5 y 2.6.

<table>
<tr><th colspan="2">1 byte</th><th>1 byte</th><th colspan="2">1 byte</th><th>1 byte</th></tr>
<tr><td>Vers.</td><td>IHL</td><td>TOS</td><td colspan="3">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td colspan="2">Flags</td><td>FO</td></tr>
<tr><td colspan="2">TTL</td><td>Protocol</td><td colspan="3">Header Checksum</td></tr>
<tr><td colspan="6">Source IPv4 address (4 bytes)</td></tr>
<tr><td colspan="6">Destination IPv4 address (4 bytes)</td></tr>
<tr><td colspan="2">Options</td><td colspan="4">Padding</td></tr>
</table>

GRÁFICO 2.6 Campo ToS de IPv4

<table>
<tr><th colspan="2">1 byte</th><th>1 byte</th><th colspan="2">1 byte</th><th>1 byte</th></tr>
<tr><td>Vers.</td><td>Traffic Class</td><td colspan="4">Flow Label</td></tr>
<tr><td colspan="3">Payload Length</td><td>Next Header</td><td colspan="2">Hop Limit</td></tr>
<tr><td colspan="6">Source IPv6 address (16 bytes)</td></tr>
<tr><td colspan="6">Destination IPv6 address (16 bytes)</td></tr>
<tr><td colspan="6">Extensions (variable)</td></tr>
</table>

GRÁFICO 2.7 Campo TC de IPv6

[15] Calidad de Servicios: Servicios Diferenciados, Pág. 4.

Seis bits del campo DS se usan como un *codepoint* (DSCP) para seleccionar el PHB (Per Hop Behavior) en cada interfaz. Un campo actualmente no usado de 2 bits (CU) esta reservado. El valor de CU es ignorado por los nodos que implementan Diffserv, cuando se determina el PHB que se aplicará a cada paquete.

La estructura del campo DS se indica a continuación en el GRÁFICO 2.8:

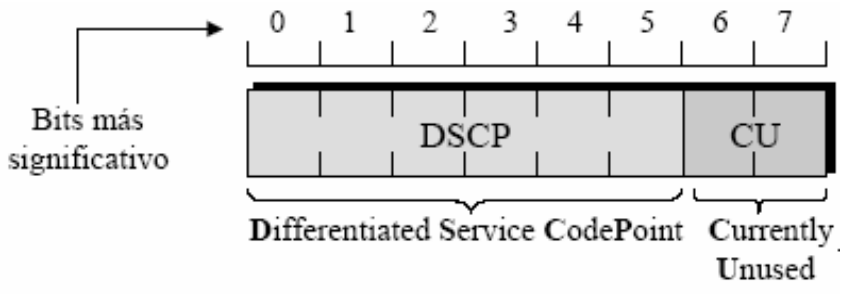


GRÁFICO 2.8 Campo DS.

El campo DSCP dentro del campo DS es capaz de tener hasta 64 valores distintos. Estas 64 combinaciones están divididas en tres pools para propósitos de asignación y administración:

- Un pool de 32 códigos estandarizados (Pool 1).
- Un pool de 16 códigos (Pool 2), a ser utilizados para uso experimental o local (EXP/LU).
- Un pool de 16 códigos (Pool 3), los cuales inicialmente están disponibles para uso experimental o uso local, pero que pueden ser preferentemente utilizados para estandarización en caso de que el Pool 1 se use en su totalidad.

Los pools están definidos en la tabla 2.1:

Pool	Código	Utilización
1	xxxxx0	Estandarizado
2	xxxx11	EXP/LU
3	xxxx01	EXP/LU (*)

(*) Pueden ser estandarizados para uso en el futuro.

TABLA 2.1 Pool de Códigos DSCP

La tabla 2.2, muestra las 64 combinaciones que puede tener el DSCP del campo DS:

DSCP	PHB	DSCP	PHB	DSCP	PHB	DSCP	PHB
000 000	CS0 (DB)	010 000	CS2	100 000	CS4	110 000	CS6
000 001	EXP/LU	010 001	EXP/LU	100 001	EXP/LU	110 001	EXP/LU
000 010	-	010 010	AF21	100 010	AF41	110 010	-
000 011	EXP/LU	010 011	EXP/LU	100 011	EXP/LU	110 011	EXP/LU
000 100	-	010 100	AF22	100 100	AF42	110 100	-
000 101	EXP/LU	010 101	EXP/LU	100 101	EXP/LU	110 101	EXP/LU
000 110	-	010 110	AF23	100 110	AF43	110 110	-
000 111	EXP/LU	010 111	EXP/LU	100 111	EXP/LU	110 111	EXP/LU
001 000	CS1	011 000	CS3	101 000	CS5	111 000	CS7
001 001	EXP/LU	011 001	EXP/LU	101 001	EXP/LU	111 001	EXP/LU
001 010	AF11	011 010	AF31	101 010	-	111 010	-
001 011	EXP/LU	011 011	EXP/LU	101 011	EXP/LU	111 011	EXP/LU
001 100	AF12	011 100	AF32	101 100	-	111 100	-
001 101	EXP/LU	011 101	EXP/LU	101 101	EXP/LU	111 101	EXP/LU
001 110	AF13	011 110	AF33	101 110	EF	111 110	-
001 111	EXP/LU	011 111	EXP/LU	101 111	EXP/LU	111 111	EXP/LU

TABLA 2.2 Códigos DSCP

Cada código DSCP mapea un PHB determinado.

2.3.1 COMPORTAMIENTO POR SALTO (PER - HOP BEHAVIOR, PHB)

Un Comportamiento por salto (Per-Hop Behavior, PHB), es una descripción observada externamente del comportamiento de envío de un nodo DS, aplicada a un Comportamiento Agregado (Behavior Agregate) DS en particular.

Algunas distinciones de comportamiento son útiles cuando se observa que múltiples agregados de tráfico compiten por: recursos de buffer y ancho de banda en un nodo. El PHB es el medio por el cual un nodo asigna recursos a los agregados de comportamiento.

El ejemplo más simple de un PHB es uno que garantiza una mínima asignación de ancho de banda de un porcentaje X%, del enlace a un BA (sobre un intervalo razonable de tiempo).

Este PHB puede ser medido en forma justa y simple, bajo una variedad de condiciones de competencia de tráfico. Un PHB ligeramente más complicado sería garantizar una asignación de ancho de banda, de cualquier exceso de capacidad del enlace, con un reparto proporcional justo.

Los PHBs, pueden ser especificados en términos de su prioridad de recursos comparada con otros PHBs, o en términos de sus características relativas de tráfico observable.

Estos PHBs pueden ser usados como bloques de construcción para asignar recursos y deben ser especificados como un grupo (grupo PHB) para lograr consistencia. Un PHB único, definido aparte, es un caso especial de un grupo PHB.

Los PHBs son implementados en nodos por medio de algunos mecanismos de manejo de buffer y despacho de paquetes. Son definidos en términos de las características de comportamiento relacionadas a las políticas de aprovisionamiento de servicios, y no en términos de mecanismos de implementación particulares.

Es probable que más de un grupo PHB sea implementado en un nodo y sea utilizado dentro de un dominio.

Un PHB es seleccionado en un nodo por medio del mapeado del código DS en un paquete recibido. Los PHBs estándares tienen un código recomendado. Sin embargo, el espacio total de código es más largo (contiene 8 bits) que el espacio disponible (DSCP ocupa 6 bits) para los códigos recomendados para los PHBs estándar, y el campo DS deja provisiones para mapeos configurables localmente.

Todos los códigos deben ser mapeados a algún PHB, y en ausencia de alguna política local, los códigos que no están mapeados a un PHB estándar, deben ser mapeados a un PHB por defecto.

Los PHBs que se pueden utilizar son los siguientes:

- Comportamiento por omisión (Default Behavior)
- Selector de clase
- Tránsito expedito (Expedited forwarding)
- Tránsito asegurado (Assured forwarding)

Se definen dos PHB estandarizados (Tránsito expedito y Tránsito asegurado), uno por defecto (Default Behavior) y un PHB selector de clase.

2.3.1.1 PHB por Defecto (Default PHB)

Todos los nodos que implementen Diffserv deben tener disponible un PHB por "Defecto". Este es el comportamiento común que tienen los ruteadores existentes y que se trata del comportamiento de mejor esfuerzo. Cuando un paquete que ingresa al nodo no esta marcado con un DSCP, lo que quiere decir que no pertenece a un PHB, se asume que estos paquetes pertenecen a este grupo. Estos paquetes pueden ser enviados dentro de la red sin estar

sujetos a una regla en particular y la red enviara la mayor cantidad de estos paquetes a la mayor brevedad posible, sujetos a otras políticas de recursos.

La implementación correcta de este PHB debería ser una disciplina de encolamiento, que envíe los paquetes marcados con este comportamiento, a la interfaz de salida cuando la misma no tenga que satisfacer los requerimientos de ningún otro PHB, es decir que estarían en al final del proceso de envío por la interfaz de salida. Una política razonable, a tener en cuenta, es que estos grupos no deben esperar indefinidamente para su reenvío, esto se puede lograr mediante un mecanismo en cada nodo que reserve un mínimo de recursos (buffer, ancho de banda, etc.) para este tipo de tráfico, o sea para este tipo de agregados de tráfico por defecto.

Todo esto permite que los nodos que no soportan la implementación de Diffserv, puedan seguir funcionando dentro de la red como lo hacen hasta hoy.

El código DSCP recomendado para el PHB por Defecto es el siguiente, 000000, y este valor es mapeado a su PHB correspondiente. Este código es compatible con el valor existente en la práctica en los nodos que no implementan Diffserv. Si un DSCP no esta mapeado a un PHB estandarizado o de uso local, este debe ser mapeado al PHB por Defecto.

Un paquete inicialmente marcado con el DSCP para PHB por Defecto puede ser re-marcado con otro DSCP en la frontera entre dominios DS así que este será re-enviado un diferente PHB dentro del dominio, posiblemente sujeto a algunas negociaciones entre los dominios contiguos.

2.3.1.2 Selector de Clase (Class Selector PHB)

Este comportamiento define hasta ocho clases distintas en la red. El formato del código toma en cuenta los primeros 3 bits del campo DS, XXX000. Los tres primero bits representan un número del 0 al 7.

El número de menor valor representa una prioridad menor (es decir, los tres primeros bits son cero, el cual corresponde al PHB por Defecto o de mejor esfuerzo) mientras que un número mayor representa una prioridad mayor.

No es necesario que un nodo (puede ser un nodo interno) soporte las ocho clases. Puede agrupar las clases para soportar por ejemplo 2 prioridades.

Los códigos con número 1 al 3 pueden representar una prioridad baja, mientras que los códigos con los números del 4 al 7 representan una prioridad alta. De esta forma, el nodo sigue siendo compatible con la especificación DiffServ, aún sin tener ocho clases definidas.

En la tabla 2.3, se indica las 8 posibles clases que se obtienen con el PHB selector de clase:

Tabla 2.3 Códigos para el selector de clase

Clase	**Código**	**Clase**	**Código**
0	000 000	4	100 000
1	001 000	5	101 000
2	010 000	6	110 000
3	011 000	7	111 000

2.3.1.3 PHB Tránsito Expedito (Expedited forwarding)[16]

Como ya se mencionó, los nodos de red que implementan Servicios Diferenciados, usan un código en la cabecera IP para seleccionar un PHB, el cual le da un tratamiento de envío específico para los paquetes.

El PHB Tránsito Expedito (EF PHB) puede ser usado para construir un servicio punto a punto de bajas pérdidas, baja latencia, bajo jitter (colas muy pequeñas) y ancho de banda asegurado, a través de dominios DS, por ello suele recibir el nombre de “Servicio Premium”.

Este servicio aparece en los puntos finales como una conexión punto a punto o como una “línea virtual alquilada”.

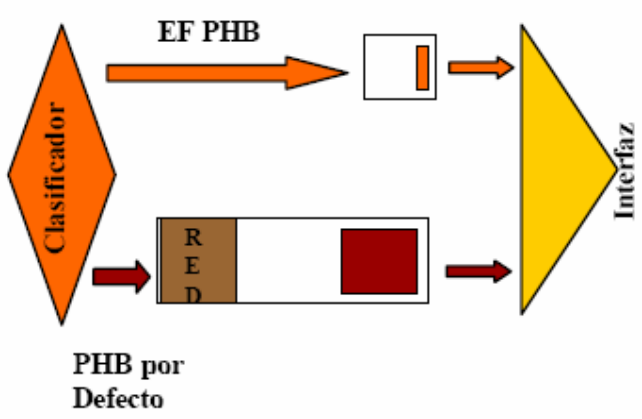


GRÁFICO 2.9 Implementación del EF PHB con el PHB por defecto

Pérdidas, latencia y jitter son todos debidos al encolamiento que el tráfico experimenta mientras transita la red. Por lo tanto, el proveer bajas pérdidas, latencia y jitter para algunos agregados de tráfico, significa asegurarse que el agregado no vea ninguna, o por lo menos una pequeña cola, tal como se indica en GRÁFICO 2.9.

[16] RFC 2598, Expedited Forwarding PHB

Las colas se incrementan cuando la tasa o velocidad del tráfico entrante excede la tasa de salida en algún nodo. Así un servicio que asegura la no existencia de colas para algunos agregados, es equivalente a limitar la velocidad, tal que en cada nodo de transito en la red, la máxima velocidad de arribo de un agregado de tráfico sea menor que la velocidad mínima de salida del agregado de tráfico.

Crear un servicio como éste consta de dos partes:

1. Configurar los nodos de forma que el agregado de tráfico tenga una velocidad de salida mínima *bien definida*. (*bien definida* significa independiente del estado dinámico del nodo. En particular, independiente de la intensidad de otro tráfico en el nodo).
2. Acondicionar el agregado de tal manera que su tasa de llegada en cualquier nodo sea siempre menor que la tasa de salida mínima configurada para ése nodo.

El PHB Tránsito Expedito provee la primera parte para este servicio, mientras que los acondicionadores de tráfico en las fronteras de la red proveen la segunda parte.

Este PHB no es parte imperativa en la arquitectura Diffserv, lo que quiere decir que un nodo no necesita implementar el PHB Tránsito Expedito para ser considerado como un nodo que cumple con Diffserv. Otros PHBs y grupos PHB pueden ser de utilizados, en el mismo nodo DS o dominio, con el PHB Tránsito Expedito.

2.3.1.3.1 Descripción del PHB Tránsito Expedito.

El PHB Tránsito Expedito, es definido como un tratamiento de envío para un agregado de tráfico Diffserv particular, donde la tasa de salida de los paquetes

del agregado de cualquier nodo Diffserv debe ser igual o exceder una velocidad ya especificada.

El tráfico con Tránsito Expedito debe recibir esta velocidad independiente de la intensidad de cualquier otro tráfico que intente transitar el nodo. Esta velocidad debe promediar al menos la velocidad configurada cuando se mide en cualquier intervalo de tiempo o ser mayor al tiempo que toma enviar un paquete MTU (Unidad Máxima de Transferencia) al enlace de salida a la velocidad configurada, esto significa que si tenemos una serie de paquetes del mismo tamaño que llegan a un nodo, éstos saldrán del nodo con la misma velocidad de entrada. La idea es reducir el exceso de retardo y jitter en lo posible.

El GRÁFICO 2.10 nos muestra el principio de operación del EF PHB.

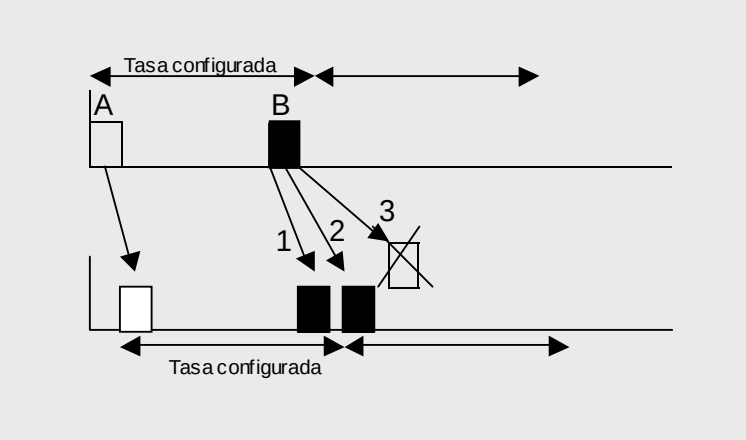


GRÁFICO 2.10 Modelado y descarte de paquetes

Cuando el paquete llega antes de su tiempo programado de llegado, existen tres opciones en los nodos de ingreso e internos para su tratamiento:

1. Reenviar el paquete inmediatamente
2. Reenviar el paquete en el tiempo configurado
3. Descartar el paquete

Las opciones que toman los nodos de acceso e internos son diferentes: los nodos de acceso por lo general tomaran las opciones 2 y 3 para evitar que la fuente se apropie de un mayor ancho de banda del que se tiene configurado. Para los nodos internos, es altamente recomendada la opción 1, ya que la aplicación de la opción dos podría provocar retardos acumulados.

El DSCP 101110 es el recomendado para este PHB.

Los paquetes marcados con un EF PHB, pueden ser remarcados en el borde de un dominio DS solo con otro DSCP que satisfaga el EF PHB.

2.3.1.3.2 Mecanismos de ejemplo para implementar el PHB Tránsito Expedito.

Varios mecanismos de despacho de cola pueden ser empleados para entregar el tipo de comportamiento descrito anteriormente, y así implementar el EF PHB.

Entre los cuales tenemos:

- Una cola con prioridad simple, la cual dará el comportamiento apropiado siempre y cuando no halla una cola con prioridad más alta, que pueda apropiarse del EF por más de un tiempo de paquete, a la tasa configurada.

- Planificador CBQ (Class Based Queue), que dé la prioridad al tráfico EF, hasta la velocidad ya establecida.

2.3.1.3.3 Consideraciones de Seguridad.

Para protegerse de ataques de rechazos de servicios, la frontera de un dominio DS, debe estrictamente aplicar la política a todos los paquetes marcados EF,

de manera que asegure, la tasa negociada con los dominios de frontera superiores. (Esta tasa debe ser menor o igual que la tasa de EF PHB configurada). Los paquetes en exceso de la tasa negociada deben ser descartados.

Si dos dominios adyacentes no han negociado una tasa EF, el domino inferior debe usar una tasa igual a 0, o sea descartar todos los paquetes marcados como EF.

Como el Servicio Premium punto a punto, descrito en el punto 2.3.1.3, construido a partir del EF PHB, requiere que el dominio superior establezca y aplique una política sobre el tráfico EF marcado, para conocer la tasa negociada con el dominio inferior, la política del domino inferior nunca debe dejar paquetes desechados. Así, cuando el dominio inferior descarte paquetes EF, estos descartes deben ser anotados como posibles violaciones de seguridad o serias desconfiguraciones.

2.3.1.4 PHB Tránsito Asegurado (Assured Forwarding)[17]

El grupo PHB Tránsito Asegurado (AF PHB), provee el envío de paquetes IP en N clases AF independientes. Dentro de cada clase AF, a un paquete IP se le asigna uno de M diferentes niveles de *precedencia de descarte*. Un paquete IP que pertenece a una clase AF *i*, y tiene una precedencia de descarte *j* es marcado con un código AF_{ij}, donde $1 <= i <= N$ y $1 <= j <= M$. En la actualidad existen cuatro clases ($N = 4$) con tres niveles de precedencia de descarte en cada clase ($M = 3$).

Un nodo DS debe implementar todas las cuatro clases AF. Los paquetes dentro de una clase AF deben ser enviados independientemente de los paquetes que pertenecen a otra clase AF.

[17] RFC 2597, Assured Forwarding PHB Group

Un nodo DS debe configurarse con un mínimo de recursos, espacio en el buffer y ancho de banda, a cada clase AF implementada en el nodo.

Una clase AF también puede ser configurada para recibir más recursos de envío cuando los recursos exceden los requeridos por otras clases AF o de otros grupos PHB.

Dentro de una clase AF, un nodo DS no debe enviar un paquete IP que tiene un valor de precedencia de descarte menor (p) que un paquete que tiene una precedencia de descarte mayor (q). Esto se puede lograr sin necesidad de descartar los paquetes ya encolados.

Dentro de cada clase AF, un nodo DS debe aceptar los tres códigos de precedencias de descarte y estos deben proporcionar al menos dos niveles diferentes de probabilidad de pérdida. En algunas redes, particularmente en redes empresariales, donde la congestión trasciende rara vez o en periodos muy cortos, puede ser razonable implementar solo dos niveles de probabilidad de pérdida en los nodos DS. Mientras que esto puede ser suficiente para algunas redes, tres niveles de probabilidad de pérdida deben ser soportados por los nodos DS dentro de un dominio DS donde la congestión es algo común.

Si un nodo DS solo implementa dos niveles de probabilidad de pérdida de paquetes para una clase AF x, el código AFx1 debe proporcionar la probabilidad de pérdida más baja y el código AFx2 y AFx3 deben proporcionar las probabilidades más altas de pérdida de paquetes.

Un nodo DS no debe reordenar los paquetes AF de un mismo flujo cuando estos pertenecen a la misma clase AF sin tener en cuenta su precedencia de descarte. No hay requerimientos cuantificables de medir el tiempo de retardo o las variaciones de retardo, asociado con el envío de paquetes AF.

El grupo AF PHB puede ser usado para implementar el servicio extremo a extremo y domain edge – a – domain edge.

2.3.1.4.1 Acciones de Condicionamiento de Tráfico

En la frontera de un dominio DS se puede controlar la cantidad de tráfico de cada clase AF que entra o sale del dominio con varios niveles de precedencia de descarte. El control de tráfico puede implicar conformado, descarte, aumento o disminución de la precedencia de descarte y reasignación de paquetes a otras clases AF. Sin embargo no se debe realizar ninguna acción que implique el reordenamiento de paquetes de un mismo microflujo.

2.3.1.4.2 Comportamiento de colas y descarte de paquetes

La implementación del AF PHB debe minimizar la congestión de larga duración dentro de cada clase AF, y a la vez permitir las congestiones de corta duración causadas por las ráfagas de tráfico. Esto requiere de un algoritmo de administración de cola activa.

Un ejemplo de esta clase de algoritmo es el Descarte Temprano Aleatorio (Random Early Drop - RED).

La implementación de AF debe detectar y responder a las congestiones de larga duración en cada clase AF mediante el descarte de paquetes, mientras que el manejo de las congestiones de corta duración se maneja mediante el encolamiento de paquetes. Esto implica la presencia de funciones de filtrado que monitoree el nivel de congestión instantánea.

El algoritmo de descarte de paquetes debe tratar todos los paquetes dentro de una sola clase y nivel de prioridad idéntico. Esto implica que para un nivel de

congestión dado, la tasa de descarte de los paquetes de un microflujo en particular que se encuentren dentro de un nivel de prioridad será proporcional al porcentaje de flujo del monto total del tráfico que atraviesa ese nivel de prioridad.

La notificación de congestión regresa a los nodos finales, y así el nivel de paquetes descartados de cada precedencia de descarte en relación a la congestión debe ser gradual y no abrupto para permitir a todo el sistema llegar a un punto estable de operación. Una forma de hacer esto es usando el algoritmo Random Early Drop (RED).

RED se basa en el descarte aleatorio de paquetes tras la detección temprana de la congestión, una vez superado un umbral del tamaño medio de la cola, el cual será explicado a continuación.

2.3.1.4.3 Algoritmo RED.

A la llegada de un paquete se estima el tamaño medio de la cola de salida Q, como se indica en el GRÁFICO 2.11:

- Si Q < umbral_min Funcionamiento normal: Almacenamiento del paquete.
- Si umbral_min < Q < umbral_max → Evitar congestión: El paquete se descarta con prob. creciente linealmente con Q:

  Pdrop=Pmax*(Q – umbral_min) / (umbral_max – umbral_min)
- Si Q > umbral_max → Congestión: Se descarta el paquete.

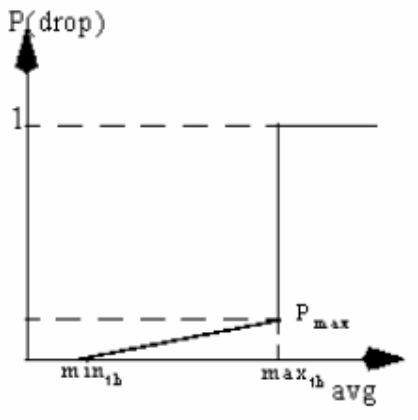


GRÁFICO 2.11 Algoritmo RED

El PHB Assured Forwarding (AF) PHB define un método por el cual los BAs pueden darse con diferentes garantías de envío hacia delante.

Por ejemplo, el tráfico de red puede clasificarse por ejemplo en un servicio denominado Servicio Olímpico, como se muestra en el GRÁFICO 2.12, y que consta de las siguientes clases:

- Oro: El tráfico de esta categoría dispone del 50 % del ancho de banda.
- Plata: reserva el 30% del ancho de banda.
- Bronce: con el 20% del ancho de banda.

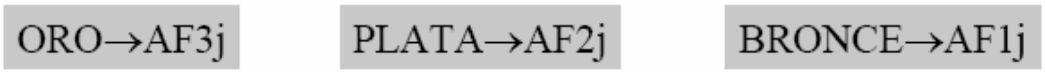


GRÁFICO 2.12 Servicio Olímpico.

Además, este PHB define cuatro clases: AF1, AF2, AF3, y AF4. A cada clase se le asigna una cantidad específica del espacio del buffer y ancho de banda de la interfaz, de acuerdo con la política establecida. En cada clase AF, se pueden especificar tres valores de precedencia: 1, 2 y 3.

Con lo cual los códigos DSCP recomendadas para uso general de las clases AF_{ij}, donde i corresponde a la clase y j corresponde a la precedencia, están dados a continuación:

AF11 =	001010	AF12 = 001100	AF13 = 001110
AF21 =	010010	AF22 = 010100	AF23 = 010110
AF31 =	011010	AF32 = 011100	AF33 = 011110
AF41 =	100010	AF42 = 100100	AF43 = 100110

Es importante señalar que no es necesario implementar los tres niveles de descarte. Si el operador de la red, no espera que existan muchas condiciones de congestión, el número de niveles de descarte se puede compactar a dos.

La tabla 2.4, resume los valores DSCP recomendados para el PHB AF.

Tabla 2.4 Códigos DS recomendados para AS PHB

	Clase 1	**Clase 2**	**Clase 3**	**Clase 4**
Baja probabilidad de descarte	001010	010010	011010	100010
Media probabilidad de descarte	001100	010100	011100	100100
Alta probabilidad de descarte	001110	010110	011110	100110

2.3.1.4.4 Consideraciones de Seguridad.

Para lograr protegerse a si mismo de ataques de negación del servicio, el proveedor del dominio DS debe limitar el tráfico entrante al dominio a los perfiles subscritos. También, para poder proteger el enlace del consumidor de los ataques de negación de servicio en el dominio DS el proveedor del dominio DS debe permitir a los consumidores especificar como los recursos del enlace son ubicados en los paquetes AF. Sí el servicio ofrecido requiere que el tráfico marcado con un DSCP AF este limitado por algunos atributos tales como

direcciones de origen y destino, es responsabilidad del nodo de ingreso a al red verificar la validez de dichos atributos.

CAPÍTULO 3

EL PROTOCOLO DIFFSERV BAJO EL SISTEMA OPERATIVO SISTEMA OPERATIVO LINUX

3.1 INTRODUCCIÓN

Uno de los puntos muy fuertes del Sistema Operativo Linux, es su gran capacidad de llevar a cabo tareas de conectividad de red, incluso con recursos modestos en hardware puede ser un hábil servidor de red y convivir con cualquier configuración de red que ya tengamos funcionando en nuestra red. A medida que se mejora, tanto el sistema operativo como el hardware del computador, el Sistema Operativo Linux, aprovechará todos los recursos convirtiéndose en un potente centro de servicios.

El entorno nativo de red del Sistema Operativo Linux, es el TCP/IP, por lo que la navegación en Internet y en general en redes basadas en esta familia de protocolos será muy fácil. El Sistema Operativo Linux puede actuar tanto de simple cliente hasta como una potente estación de trabajo a un bajo costo.

A nivel físico, el Sistema Operativo Linux puede conectarse con otros Sistemas Operativos Linux o con cualquier otro sistema, usando para ello cualquier interfaz física: cableado serie, paralelo, módems, tarjetas RDSI, Frame Relay, Ethernet o Token Ring, etc.

En cuanto a protocolos de red, ya hemos dicho que su entorno es TCP/IP, pero puede acceder a redes basadas en IPX (Novell), Apple Talk (Macintosh) SMB (Red LanManager para conectar con Windows para trabajo en grupo).

TCP/IP es un conjunto de protocolos, que permite a sistemas de todo el mundo comunicarse en una única red conocida como Internet. Con el Sistema Operativo Linux, TCP/IP y una conexión a la red, puede comunicarse con usuarios y computadores por toda la Internet, mediante correo electrónico, noticias (USENET news), transferencias de ficheros con FTP y mucho más.

Actualmente hay muchos Sistemas Operativos Linux, tanto clientes como servidores, que se encuentran conectados a Internet.

La mayoría de las redes TCP/IP usan Ethernet para la transmisión de datos. El Sistema Operativo Linux, da soporte a muchas tarjetas de red Ethernet e interfaces para computadores personales.

Pero dado que no todo el mundo tiene una conexión Ethernet en casa, el Sistema Operativo Linux también proporciona SLIP (Serial Line Internet Protocol), el cual permite conectarse a Internet a través de un modem.

Para poder usar SLIP, necesitará tener acceso a un servidor de SLIP, o sea un computador conectado a la red que permite el acceso de entrada por teléfono. Muchas empresas y universidades tienen servidores SLIP disponibles. De hecho, si su Sistema Operativo Linux dispone de conexión Ethernet y de modem, puede configurarlo como servidor de SLIP para otros usuarios.

Entre los servicios para los cuales se puede utilizar el Sistema Operativo Linux, se encuentran los siguientes: Servicios de almacenamiento de datos, servicio de impresoras y acceso a Internet, como servidor Web, mail y FTP, etc.

Para entender como el Sistema Operativo Linux puede ayudarnos en nuestro intento de construir una arquitectura de Servicios Diferenciados, es muy importante primero entender como el Sistema Operativo Linux procesa los paquetes.

3.2 FLUJO DE TRÁFICO EN EL SISTEMA OPERATIVO LINUX[18]

En el Sistema Operativo Linux cada tarjeta de Red (NIC: Network Interface Card), es manejada por un Driver de Red, el cual controla el hardware. El driver de Red actúa como un mecanismo de intercambio de paquetes entre el código de interconexión del Sistema Operativo Linux y la red física. Esto se ilustra en el siguiente GRÁFICO 3.1.

GRÁFICO 3.1 Diagrama de Bloques de interconexión entre el Sistema Operativo Linux y la red física

En este diagrama, el código de interconexión Sistema Operativo Linux puede pedir al driver de Red enviar paquetes a la red física, y viceversa. Esto nos interesa porque el Sistema Operativo Linux, se puede usar como un ruteador para re-enviar paquetes desde una interfaz a otra en el mismo computador, para lo cual se requiere de 2 NICs, como se muestra en el GRÁFICO 3.2.

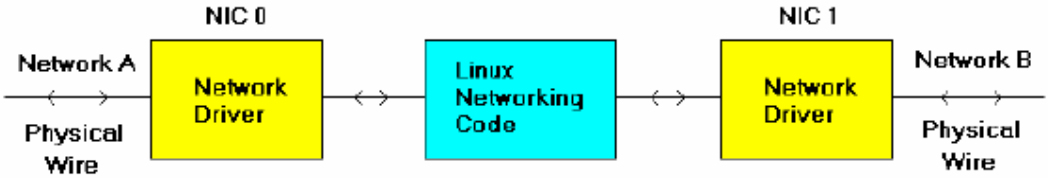


GRÁFICO 3.2 Diagrama de Bloques de un Computador con dos tarjetas de Red.

Como vemos los paquetes pueden entrar desde la red física A a la NIC 0, el driver de Red de la NIC 0, le entrega los paquetes recibidos al código de

[18] Bert Hubert, "Enrutamiento Avanzado y Control de Tráfico en Linux", http://www.redes-linux.com/manuales/routing/Enrutamiento-avanzado-y-control-de-trafico-en-Linux.pdf

Interconexión del Sistema Operativo Linux, en donde se solicita al driver de Red de la NIC 1 que re-envié los paquetes a la red física B o viceversa.

3.2.1 MECANISMO DE INTERCONEXIÓN DEL SISTEMA OPERATIVO LINUX

A continuación damos una breve descripción del comportamiento que hemos mencionado anteriormente.

El GRÁFICO 3.3, nos indica como el núcleo del Sistema Operativo Linux (Kernel), procesa los paquetes entrantes, y como este genera paquetes para ser enviados a la red.

El demultiplexor de entrada examina los paquetes que ingresan para determinar si los paquetes están destinados para el nodo local, si es así estos son enviados a la siguiente capa más alta para procesamiento adicional, si no es así este envía los paquetes al bloque de re-envío.

El bloque de re-envío, el cual también puede recibir paquetes generados localmente desde la capa superior, busca en la tabla de ruteo y determina el siguiente salto para el paquete. Después de esto, encola el paquete para ser transmitido a la interfaz de salida. En este punto, el control de tráfico en Linux entra en juego.

El control de tráfico del Sistema Operativo Linux, puede ser usado para construir una combinación compleja de disciplinas de cola, clases y filtros que controlan los paquetes que son enviados en la interfaz de salida.

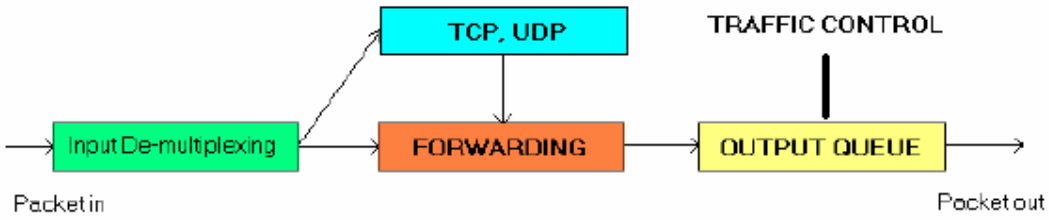


GRÁFICO 3.3 Proceso de ingreso y salida de paquetes en la Red

El momento de utilizar el Sistema Operativo Linux como un ruteador, no interesa el procesamiento local de paquetes ya que no se aceptarán paquetes y no se van a generar paquetes de forma local, en este caso el Sistema Operativo Linux solo será un ruteador, el cual solo re-envía los paquetes o quizá los descarta.

3.2.2 CONTROL DE TRÁFICO EN EL SISTEMA OPERATIVO LINUX

El proceso que sigue el núcleo del sistema operativo, para procesar los datos recibidos desde la red y generar nuevos datos para enviar a la red, se muestra en el GRÁFICO 3.4, ahí podemos observar de manera aproximada que los paquetes que llegan a la interfaz de entrada son puestos para verificar que cumplan con las políticas.

Mediante la política se descartan los paquetes indeseables. Después los paquetes que cumplen las políticas son directamente enviados a la red o pasan a las capas superiores según la pila del protocolo para procesarlas. Estas capas superiores pueden también generar datos propios y pasarlos a las capas inferiores para las tareas de encapsulación, ruteo y eventualmente transmisión.

El *re-envío* implica la selección de la interfaz de salida, la selección del próximo salto, ecapsulación, etc. una vez que todo esto se ha hecho, los paquetes son

encolados en su respectiva interfaz de salida. Este es el segundo punto en donde el Control de Tráfico del Sistema Operativo Linux entra en acción.

El Control de Tráfico en el Sistema Operativo Linux puede, entre otras cosas, decidir si los paquetes son encolados ó si estos son desechados, un ejemplo de ello sería si la cola ha alcanzado una longitud límite, o si el tráfico excede una velocidad límite.

También puede decidir en que orden se envían los paquetes, esto es útil para dar prioridad a ciertos flujos de datos, además puede retardar el envío de paquetes, etc. Una vez que el Control de Tráfico ha liberado un paquete para su envío el driver del dispositivo lo toma y lo envía a la red.

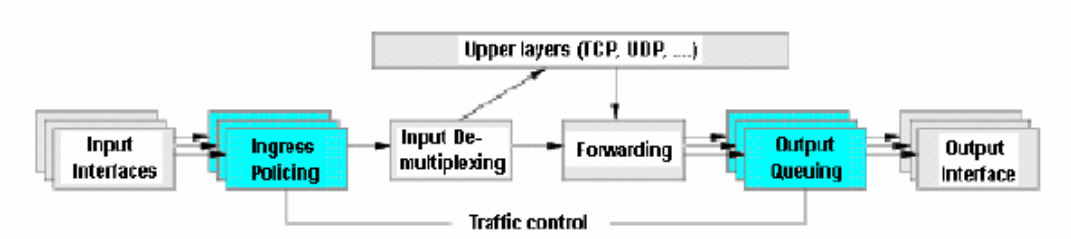


GRÁFICO 3.4 Procesamiento de datos de red.

Como ya hemos dicho en el caso de que se use el Sistema Operativo Linux, solo como ruteador, la ruta de los paquetes será tal como se indica en el GRÁFICO 3.5

GRÁFICO 3.5 Ruta de los paquetes a través de Linux

Como observamos, en los bloques celestes es en donde se ejerce el control sobre nuestros paquetes. Estos bloques son llamados *El código de control de tráfico del kernel del Sistema Operativo Linux*. Las políticas de entrada serán el

primer punto de control; aquí se descartan los paquetes indeseables. El segundo punto será la cola de salida, aquí los paquetes son encolados y luego descartados, retrasados o priorizados de acuerdo a la regla que esté implementada.

El código de control de tráfico en el kernel del Sistema Operativo Linux, consiste en los siguientes componentes principales:

- disciplinas de cola (queueing disciplines)
- clases (dentro de una disciplina de cola)
- filtros (filters)
- vigilancia (policing y los conceptos relacionados)

Cada dispositivo de red tiene una disciplina de cola asociada a él, que controla como son tratados los paquetes encolados en ese dispositivo, tal como se indica en el GRÁFICO 3.6

GRÁFICO 3.6 Disciplina de cola simple sin clases.

La disciplina de cola, a groso modo, describe la forma en que van a ser tratados los paquetes de datos dentro del buffer de la tarjeta de red.

El GRÁFICO 3.6 muestra el símbolo que utilizaremos para una disciplina de cola sin una estructura interna visible. Por ejemplo, *sch_fifo* es una disciplina de cola así de simple, que solo consiste en una sola cola, donde todos los paquetes son almacenados en el orden que han sido encolados, y que es vaciada tan rápido como el dispositivo puede enviar.

Otras disciplinas más elaboradas pueden utilizar filtros para distinguir entre diferentes clases de paquetes y procesar cada clase de una manera específica, por ejemplo, dando prioridad a una clase por encima de las otras.

El GRÁFICO 3.7 muestra un ejemplo de esa disciplina de cola. Puede verse que varios filtros mapean a la misma clase.

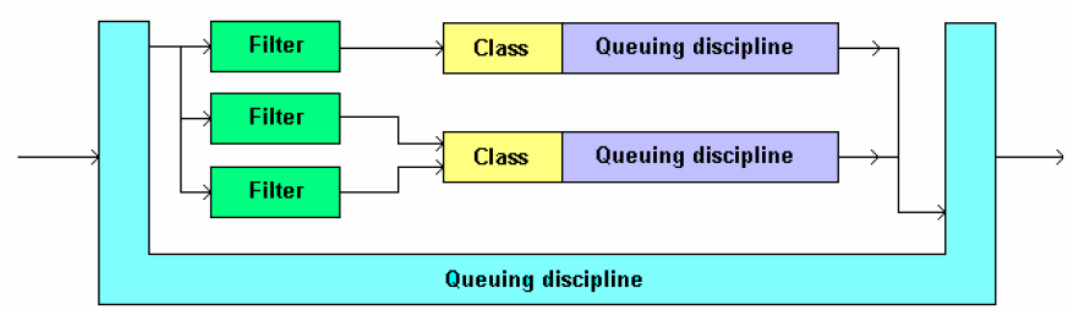


GRÁFICO 3.7 Disciplina de cola simple con múltiples clases.

Las disciplinas de cola y las clases, están íntimamente relacionadas de la siguiente manera: la presencia de clases y sus semánticas son propiedades fundamentales de la disciplina de cola. En contraste con eso, los filtros pueden ser combinados arbitrariamente con disciplinas de cola y clases, sin importar si la disciplina de cola tiene clases o no.

Pero la flexibilidad no termina aún, las clases normalmente no almacenan sus paquetes ellas mismas, utilizan otra disciplina de cola para que se encargue de eso. Esa disciplina de cola puede ser elegida arbitrariamente de un conjunto de disciplinas de cola variadas y esas disciplinas pueden tener clases, y esas clases pueden utilizar disciplinas de colas y así sucesivamente, todo esto depende de los requerimientos que tenga la red.

Los paquetes son encolados de la siguiente manera: cuando se llama a la función de encolar, de una disciplina de cola, esta ejecuta los filtros, uno tras de otro hasta que, en alguno de ellos encuentre una coincidencia. Luego encola el paquete en la clase correspondiente, lo cual generalmente significa llamar a la

función encolar de la disciplina de cola que está dentro de esa clase. Los paquetes que no coinciden con alguno de los filtros generalmente son asignados a una clase por defecto.

Típicamente, cada clase posee o es dueña de una cola, pero en principio también es posible que varias clases compartan la misma cola o que se utilice una sola cola para todas las clases de una disciplina de cola. Sin embargo es importante aclarar que los paquetes no cargan una identificación explícita de a que clase han sido asignados, por tanto las disciplinas de cola que cambian la información por clase, cuando desencolan los paquetes, pueden no trabajar correctamente si las colas "interiores" son compartidas, a menos que tengan la capacidad de repetir la clasificación del desencolado al encolado, que lleva a cabo en su interior, de alguna manera.

Usualmente cuando se encolan los paquetes, el flujo correspondiente puede ser vigilado, por ejemplo descartando paquetes que exceden cierta velocidad.

En el GRÁFICO 3.8, se muestra como los elementos del control de Tráfico del Sistema Operativo Linux, se relacionan con la Arquitectura de Diffserv[19]. Se puede notar que las clases pueden jugar dos roles, porque determinan el resultado final de una clasificación y también pueden ser parte de un mecanismo que implementa algún comportamiento de encolado o scheduling.

[19] MODELO DE LA ARQUITECTURA DIFFERENTIATED SERVICES, Capítulo 2, Pág. 47.

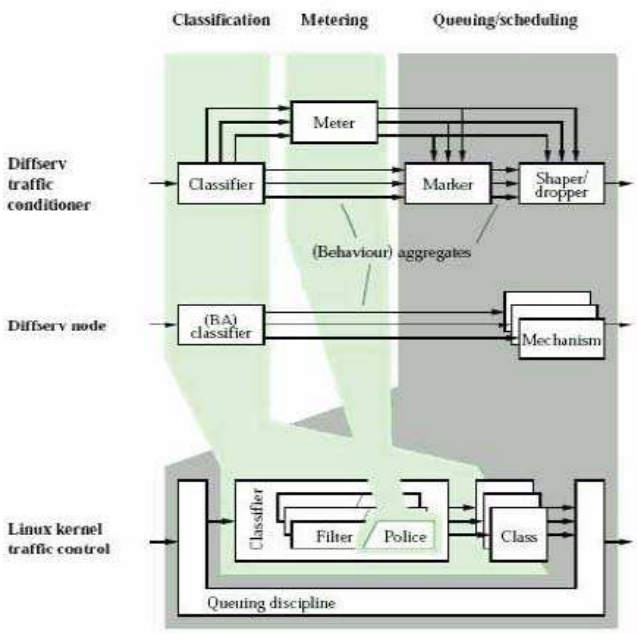


GRÁFICO 3.8 Relación de los elementos de la Arquitectura Diffserv con el Control de Tráfico del Kernel de Sistema Operativo Linux.

3.2.2.1 Elementos del Control de Tráfico en Sistema Operativo Linux.

3.2.2.1.1 Conformado (Shaping)

Un "shaper" o conformador retarda los paquetes para satisfacer una velocidad estipulada. Se utilizan para limitar el tráfico, de manera que no supere una velocidad, generalmente para suavizar el tráfico de ráfagas. Utilizan mecanismos de token bucket, el cual consiste en un algoritmo de regulación de tráfico, cuyo comportamiento se basa en el envío de testigos (tokes), a través del bucket (balde).

El bucket (balde) contiene testigos (tokens), que son generados a una tasa de *p* testigos por segundos, hasta un máximo de *C* testigos, de forma que cada testigo da derecho a transmitir un paquete. Así, este algoritmo permite ahorrar

testigos cuando no hay datos que transmitir, para luego enviar ráfagas de al menos C paquetes.

3.2.2.1.2 Calendarizador (Scheduling)

Un "scheduler" o calendarizador ordena o desordena los paquetes antes de ser desencolados. Scheduling es el mecanismo por el cual los paquetes son ordenados o desordenados entre la entrada y la salida de una cola.

Los mecanismos de scheduling, tratan de compensar varias condiciones de red. Un algoritmo del tipo "Encolado Justo" (Fair queuing - SFQ) intenta prevenir que un solo flujo se apodere del uso de la red. Un algoritmo tipo Round Robin (WRR), le da a cada flujo un turno para salida de la cola.

3.2.2.1.3 Clasificador (Classifying)

Los clasificadores ordenan o separan el tráfico entre colas. Clasificación es el mecanismo por el cual los paquetes son separados para tener diferente tratamiento, posiblemente diferente colas de salida.

En el Sistema Operativo Linux, el modelo permite que, un paquete fluya a través de una serie de clasificadores dentro de una estructura de control de tráfico y que sea clasificado de acuerdo a las políticas.

3.2.2.1.4 Politicas (Policing)

Los "policers" miden y limitan el tráfico en una cola en particular. Policing o políticas, como un elemento del control de tráfico, es simplemente un mecanismo por el cual el tráfico es limitado. Una política aceptará tráfico a una

determinada velocidad, y luego realizará alguna acción si el tráfico excede la velocidad fijada. Una de las opciones puede ser que el tráfico sea descartado o puede ser reclasificado.

Una política es una pregunta tipo si/no acerca de que hacer con el tráfico que ingresa en una cola. Si bien una política utiliza mecanismos de token bucket no tiene la capacidad de retardar el tráfico como un "shaper".

3.2.2.1.5 Descartador (Dropping)

Descarta un paquete, un flujo o una clasificación.

3.2.2.1.6 Marcador (Marking)

Es el mecanismo por el cual un paquete es alterado o marcado. No confundir con el marcado provisto por *fwmark* (marca de Firewall en Sistema Operativo Linux).

Los mecanismos de marcado del control de tráfico instalan una marca en el paquete mismo, la cual es usada y respetada por otros ruteadores dentro de un mismo dominio administrativo (DiffServ).

La tabla 3.1 muestra los componentes del Control de Tráfico en Sistema Operativo Linux.

Tabla 3.1 Componentes del Control de Tráfico en Sistema Operativo Linux.

Elemento Tradicional	Componente de Linux
Shaping	Una *class* ofrece capacidades de shaping.
qdisc	Una *qdisc (queue discipline)* es un scheduler.
Classifying	El objeto *filter* realiza la clasificación utilizando un objeto *classifier*. En Linux los *classifiers* no pueden existir fuera de los *filter*.
policing	Un *policer* solo existe como parte de un *filter*.
dropping	Para hacer drop del tráfico se requiere un *filter* con un *policer* que utilice "drop" como acción.
marking	Se utiliza *dsmarkqdisc*.

3.3 INTRODUCCIÓN A iproute2

La mayoría de las distribuciones ó tipos de Sistemas Operativos Linux, usan actualmente las venerables órdenes **arp**, **ifconfig** y **route,** aunque funcionan, muestran cierto comportamiento inesperado a partir del Sistema Operativo Linux 2.2. Por ejemplo, los túneles GRE son parte integral del enrutado hoy día, pero precisan herramientas completamente diferentes.

Con iproute2, los túneles son una parte integral que ya se incluye, en el juego de herramientas de iproute2.

Los núcleos del Sistema Operativo Linux 2.2 y superiores incluyen un subsistema de red completamente rediseñado. Este nuevo código de red proporciona al Sistema Operativo Linux, un rendimiento y características con poca competencia en el panorama general de los Sistemas Operativos.

En realidad, el nuevo código de enrutado, filtrado y clasificación tiene más posibilidades, que el que proporcionan muchos ruteadores y cortafuegos dedicados y productos de control de tráfico.

Según se crean nuevos conceptos de red, la gente encuentra maneras de implementarlos encima de la infraestructura existente en los SO. Este continuo apilamiento de conceptos ha llevado a un código de red lleno de comportamientos extraños. En el pasado, Sistema Operativo Linux emuló la forma de Sun OS de gestionar muchas de estas cosas, pero no era ideal.

Esta nueva infraestructura hace posible expresar claramente características que antes estaban más allá del alcance de Sistema Operativo Linux.

3.3.1 REVISIÓN DE iproute2

El Sistema Operativo Linux, tiene un sistema sofisticado para proporcionar ancho de banda llamado Traffic Control. Este sistema soporta varios métodos de clasificación, priorizado, compartición y limitación tanto de tráfico entrante como saliente.

Para poder utilizar esta potente utilidad disponible en Linux, nos debemos asegurar de que estén instaladas las herramientas necesarias. Este paquete se llama «iproute».

Algunas partes de iproute precisan que estén activas ciertas opciones del núcleo. Es importante señalar que la mayoría de las versiones de RedHat vienen con la mayoría de capacidades de control de tráfico en el núcleo de serie.

Por defecto iproute2 ya está configurado en el SO, las órdenes **ifconfig** y **route** actuales ya usan las llamadas a sistema avanzadas, pero en su mayoría con

configuraciones por defecto que deben configurarse para que trabajen adecuadamente según sea el entorno.

La herramienta ***ip*** se encuentra dentro del conjunto de herramientas de iproute2, y es una herramienta central con la cual podemos ver nuestras interfaces de la siguiente manera:

```
[nombre del host]$ ip link list
1: lo: <LOOPBACK,UP> mtu 1518 qdisc no queue
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: dummy: <BROADCAST,NOARP> mtu 1500 qdisc noop
   link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
3: eth0: <BROADCAST,MULTICAST,PROMISC,UP> mtu 1400 qdisc pfifo_fast qlen 100
    link/ether 48:54:e8:2a:47:16 brd ff:ff:ff:ff:ff:ff
4: eth1: <BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:e0:4c:39:24:78 brd ff:ff:ff:ff:ff:ff
    3764: ppp0: <POINTOPOINT,MULTICAST,NOARP,UP> mtu 1492 qdisc pfifo_fast qlen 10
    link/ppp
```

Este es un ejemplo tomado de un ruteador NAT casero. En el cual observamos primero a la interfaz loopback, la cual es recomendable que esté configurada en el computador, el tamaño del MTU (Maximum Transfer Unit) es de 1518 bytes, como se indica en la segunda línea del comando, y además no tiene implementado ninguna disciplina cola, esto tiene sentido ya que la interfaz loopback es una interna en el núcleo.

Después observamos dos interfaces de red físicas que corresponden a una interfaz ethernet.

Notemos que entre la información de cada interfaz está ausente las direcciones IP, de esta forma iproute separa los conceptos de «enlace» y «dirección IP», pero sin embargo, nos muestra las direcciones MAC, que corresponden al identificador en hardware de nuestras interfaces ethernet.

Además de mostrarnos las interfaces que tenemos, la herramienta ip nos puede mostrar las direcciones IP de las mismas, como se muestra a continuación:

```
[nombre del host]$ ip address show
1: lo: <LOOPBACK,UP> mtu 1518 qdisc no queue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
2: dummy: <BROADCAST,NOARP> mtu 1500 qdisc noop
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
3: eth0: <BROADCAST,MULTICAST,PROMISC,UP> mtu 1400 qdisc pfifo_fast qlen 100
    link/ether 48:54:e8:2a:47:16 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.1/8 brd 10.255.255.255 scope global eth0
4: eth1: <BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:e0:4c:39:24:78 brd ff:ff:ff:ff:ff:ff
3764: ppp0: <POINTOPOINT,MULTICAST,NOARP,UP> mtu 1492 qdisc pfifo_fast qlen 10
    link/ppp
    inet 212.64.94.251 peer 212.64.94.1/32 scope global ppp0
```

Como se observa en este ejemplo el resultado nos ofrece mayor información, la cual nos muestra todas las direcciones IP, que pertenecen a cada interfaz.

Por ejemplo examinemos la interfaz eth0, la información nos indica que la dirección IP inet es 10.0.0.1/8, donde el /8 indica el número de bits correspondientes a la Dirección de Red, la cual es la Red 10.0.0.0, con una máscara de red (netmask) de 255.0.0.0.

Con lo cual cualquier computador de esta red, por ejemplo el computador 10.250.3.13, es accesible de forma directa desde la interfaz eth0.

Con la interfaz ppp0 sucede lo mismo, su dirección es 212.64.94.251, sin máscara de subred, esto significa que tenemos una conexión punto a punto y que cada dirección, con la excepción de 212.64.94.251, es remota, además nos indica que en el otro extremo del enlace hay una única dirección, 212.64.94.1/32, en donde el /32 nos indica que no existen bits de red.

Mediante el uso la herramienta ip que se muestra a continuación podemos sabe por donde son enrutados nuestros paquetes hacia fuera:

```
[nombre del host]$ ip route show
212.64.94.1 dev ppp0 proto kernel scope link src 212.64.94.251
10.0.0.0/8 dev eth0 proto kernel scope link src 10.0.0.1
127.0.0.0/8 dev lo scope link
default via 212.64.94.1 dev ppp0
```

Las primeras 4 líneas indican explícitamente lo que quedó implícito con *ip address show*, y la última línea nos dice que los paquetes serán enrutados al resto del mundo mediante la interfaz con dirección IP 212.64.94.1, que corresponde a nuestra pasarela por defecto.

3.3.2 PROTOCOLO DE RESOLUCIÓN DE DIRECCIONES (ADDRESS RESOLUTION PROTOCOL, ARP)

ARP es usado en un computador en red para averiguar la dirección MAC de otra computador en la misma red local, ya que las direcciones IP, no indican esta información, para ello se utiliza ARP, que es un protocolo de nivel de red.

Tomemos un ejemplo muy sencillo. Supongamos que tengo una red compuesta de varias computadoras, dos de ellas son *A* con la dirección IP 10.0.0.1 y *B* con la dirección IP 10.0.0.2. Ahora A quiere hacer ping hacia B, para ver si está en la red, pero, en este punto, la computadora A no tiene idea de dónde está la computador B. De manera que cuando A decide hacer ping hacia B, necesita realizar una consulta ARP. Esta consulta ARP es algo así como si A gritase en la red « ¡B (10.0.0.2)! ¿Dónde estás?» Como resultado de esto, cada computador de la red escuchará el grito de A, pero sólo B (10.0.0.2) responderá. El computador B enviará entonces una respuesta ARP directamente a la computadora A, que viene a ser como si B dijese, «A (10.0.0.1), estoy aquí en 00:60:94:E9:08:12». Después de esta sencilla

transacción que sirve para localizar a la computadora B en la red, la computadora A, es capaz de comunicarse con la computadora B.

La memoria caché ARP se puede observar utilizando el siguiente comando:

```
[nombre del host /home/src/iputils]# ip neigh show
9.3.76.42 dev eth0 lladdr 00:60:08:3f:e9:f9 nud reachable
9.3.76.1 dev eth0 lladdr 00:06:29:21:73:c8 nud reachable
```

Como podemos observar la computadora *espa041* con dirección IP: 9.3.76.41, sabe dónde encontrar a la computador *espa042* con dirección IP: 9.3.76.42; y a la computadora *gateway* con dirección IP: 9.3.76.1.

Ahora añadamos otra computadora a la caché arp, esto lo realizamos haciendo ping desde el computador espa041 a la computadora espa043, utilizando el siguiente comando:

```
[nombre del host /home/paulsch/.gnome-desktop]# ping -c 1 host3
PING host3 (9.3.76.43) from 9.3.76.41 : 56(84) bytes of data.
64 bytes from 9.3.76.43: icmp_seq=0 ttl=255 time=0.9 ms
--- host3 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.9/0.9/0.9 ms
[nombre del host /home/src/iputils]# ip neigh show
9.3.76.43 dev eth0 lladdr 00:06:29:21:80:20 nud reachable
9.3.76.42 dev eth0 lladdr 00:60:08:3f:e9:f9 nud reachable
9.3.76.1 dev eth0 lladdr 00:06:29:21:73:c8 nud reachable
```

Como resultado de que host1 haga ping con host3, se ha añadido la dirección MAC de host3 a la caché arp/neighbor. De manera que mientras no caduque la entrada de host3, en la caché, como resultado de la ausencia de comunicación entre ambas, el host1 sabe dónde encontrar a host3 y no necesita enviar una consulta ARP.

Ahora, eliminemos a host3 de nuestra caché arp:

```
[host1 /home/src/iputils]# ip neigh delete 9.3.76.43 dev eth0
[host1 /home/src/iputils]# ip neigh show
9.3.76.43 dev eth0 nud failed
9.3.76.42 dev eth0 lladdr 00:60:08:3f:e9:f9 nud reachable
9.3.76.1 dev eth0 lladdr 00:06:29:21:73:c8 nud stale
```

Ahora host1 ha vuelto a olvidar dónde encontrar a host3 y necesitará enviar otra consulta ARP cuando necesite comunicarse de nuevo con host3.

También podemos observar en el listado anterior que gateway (9.3.76.1) ha cambiado al estado *stale*, que significa que la localización mostrada todavía es válida, pero tendrá que ser confirmada en la primera transacción que se haga con ese computador.

3.3.3 REGLAS DE ENRUTAMIENTO (BASE DE DATOS DE NORMAS DE ENRUTADO)

En una red grande, el administrador de red tiene que encargarse de las necesidades y requerimientos de las personas que se encuentran laborando en dicha red, estos requerimientos deben ser atendidos de forma diferente. La base de datos de normas de enrutamiento (routing policy database), le permite hacerlo teniendo varios conjuntos de tablas de enrutamiento.

Para usar esta característica del Sistema Operativo Linux, debemos asegurarnos que el núcleo este compilado con las opciones «IP: advanced router» e «IP: policy routing».

Cuando el núcleo necesita tomar una decisión de encaminamiento, busca la tabla que necesita consultar. Por defecto, hay tres tablas. Con la ayuda de la herramienta ip podemos modificar las tablas principal y local.

Las reglas por defecto son:

```
[nombre del host]$ ip rule list
0: from all lookup local
32766: from all lookup main
32767: from all lookup default
```

La lista se encuentra según la prioridad de todas las reglas, además observamos que todas son aplicables a todos los paquetes, lo cual se indica con la frase «from all».

Mediante la creación de nuevas reglas podemos generar diferentes tablas que nos permitan saltarnos las reglas generales de enrutamiento del sistema y lograr una mejora sustancial en el enrutamiento de los paquetes en una red.

3.3.3.1. Reglas de Enrutamiento por origen sencillas

Para explicar esta norma de encaminamiento tomaremos el siguiente ejemplo:

Supongamos una residencia con varias familias, en donde se tiene dos conexiones de cable módems conectados a un ruteador Linux, que tiene configurado el servicio de Traslación de Direcciones de Red (Network Address Translation, NAT), y supongamos que una familia sólo visita hotmail y desea pagar menos, lo cual es totalmente razonable, pero en este caso usará el cable módem de menos prestaciones.

El cable módem «rápido», de mejores prestaciones, se conoce como 212.64.94.251 y es un enlace PPP que tiene en su otro extremo el computador 212.64.94.1. El «lento», de menores prestaciones, es conocido por varias IP, por ejemplo 212.64.78.148, y es un enlace que conecta con el computador 195.96.98.253.

Las tablas que se encuentran configuradas en el ruteador Linux con NAT, son las siguientes:

La tabla local:

```
[nombre del host]$ ip route list table local
broadcast 127.255.255.255 dev lo proto kernel scope link src 127.0.0.1
local 10.0.0.1 dev eth0 proto kernel scope host src 10.0.0.1
broadcast 10.0.0.0 dev eth0 proto kernel scope link src 10.0.0.1
local 212.64.94.251 dev ppp0 proto kernel scope host src 212.64.94.251
broadcast 10.255.255.255 dev eth0 proto kernel scope link src 10.0.0.1
broadcast 127.0.0.0 dev lo proto kernel scope link src 127.0.0.1
local 212.64.78.148 dev ppp2 proto kernel scope host src 212.64.78.148
local 127.0.0.1 dev lo proto kernel scope host src 127.0.0.1
local 127.0.0.0/8 dev lo proto kernel scope host src 127.0.0.1
```

La tabla por defecto está vacía.

La tabla «main», ó principal:

```
[nombre del host]$ ip route list table main
195.96.98.253 dev ppp2 proto kernel scope link src 212.64.78.148
212.64.94.1 dev ppp0 proto kernel scope link src 212.64.94.251
10.0.0.0/8 dev eth0 proto kernel scope link src 10.0.0.1
127.0.0.0/8 dev lo scope link
default via 212.64.94.1 dev ppp0
```

Con estos antecedentes, generaremos una nueva regla que llamaremos «John», para nuestra familia hipotética. Aunque podemos trabajar con números, es mucho más sencillo si utilizamos el comando: /etc/iproute2/rt_tables, que se muestra a continuación.

```
# echo 200 John >> /etc/iproute2/rt_tables
# ip rule add from 10.0.0.10 table John
# ip rule ls
0: from all lookup local
```

32765: from 10.0.0.10 lookup John
32766: from all lookup main
32767: from all lookup default

Ahora todo lo que queda es generar la tabla John, y refrescar la caché de rutas, esto lo realizamos mediante el comando *flush caché,* que actualiza la tabla de enrutamiento para que hagan efecto los cambios y aparezca la nueva regla John:

ip route add default via 195.96.98.253 dev ppp2 table John
ip route flush caché

3.3.3.2. Reglas de Enrutamiento con varios enlaces de salida/proveedores

En el GRÁFICO 3.9, se indica una configuración común, en la que hay dos proveedores que conectan una red local al Internet.

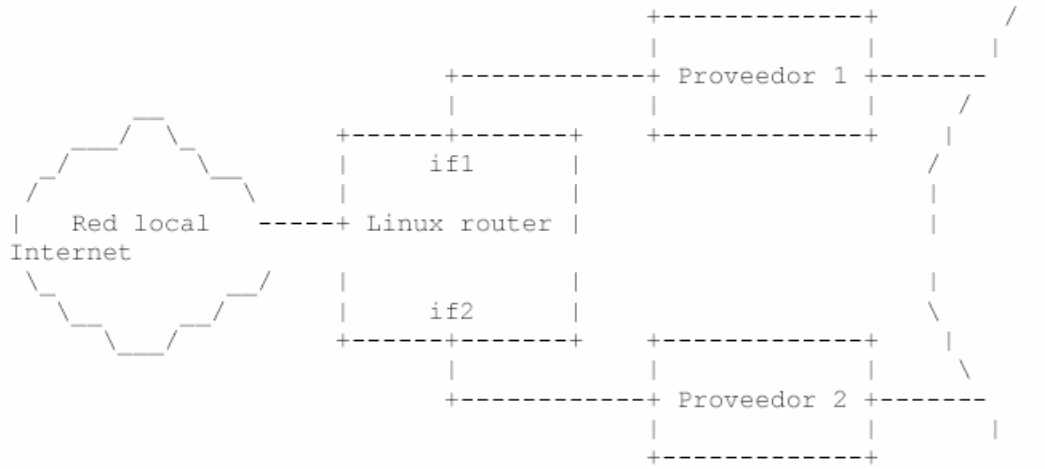


GRÁFICO 3.9 Configuración de proveedores hacia Internet

Normalmente surgen dos preguntas dada esta configuración:

1. Cómo enrutar respuestas a paquetes que vienen de un proveedor particular de vuelta por el mismo camino.
2. Cómo equilibrar el tráfico que va por los dos proveedores.

3.3.3.2.1. Acceso dividido

La respuesta a la primera pregunta se da de la siguiente manera:

Primero establezcamos algunos nombres simbólicos. Digamos que $IF1 es el nombre de la primera interfaz, if1 en la figura, y $IF2 el nombre de la segunda. Sean entonces $IP1 la dirección IP asociada con $IF1 y $IP2 la IP asociada con $IF2. Entonces, digamos que $P1 es la dirección IP de la pasarela que nos lleva al Proveedor 1, y $P2 la IP de la pasarela al Proveedor 2. Por último, $P1_NET será la red IP donde está $P1, y $P2_NET la red IP donde está $P2.

Luego creamos dos tablas de enrutamiento adicionales T1 y T2, y Las añadimos a /etc/iproute2/rt_tables.

Las configuraciones de las dos tablas se muestran a continuación:

```
ip route add $P1_NET dev $IF1 src $IP1 table T1
ip route add default via $P1 table T1
ip route add $P2_NET dev $IF2 src $IP2 table T2
ip route add default via $P2 table T2
```

Con esta configuración se ha montado una ruta hacia una pasarela, y una ruta por defecto mediante ella, tal como sería el caso con un único proveedor, pero ponemos las rutas en tablas separadas, una por proveedor. Podemos Observar que basta la ruta hacia la red, ya que le indica cómo encontrar cualquier computador dentro de esa red, lo que incluye la pasarela.

Después configuramos la tabla de enrutamiento principal (tabla main). Se recomienda encaminar los paquetes a la computadora más cercana, mediante la interfaz conectada a esa computadora.

Luego configuramos la ruta por defecto de preferencia mediante el comando *ip route add default via $,* como se indica a continuación:

```
ip route add default via $P1
```

A continuación, configuramos las reglas de enrutamiento, las cuales escogen qué tabla de ruteo se usa:

```
ip rule add from $IP1 table T1
ip rule add from $IP2 table T2
```

Estas órdenes se aseguran de que todas las respuestas que se envíen al tráfico proveniente de una interfaz en particular serán contestadas por esta interfaz.

Esta es la configuración más básica y funcionará para todos los procesos que estén funcionando en el propio ruteador, y para la red local.

3.3.3.2.2. Balanceo de carga

La respuesta a la segunda pregunta en realidad no es difícil luego de haber aprendido ha configurar un acceso dividido como se indicó previamente.

Lo que debemos hacer es, en lugar de escoger uno de los proveedores como la salida por defecto, debemos configurar la ruta por defecto para que sea multicamino (multipath route), el núcleo por defecto balanceará las rutas sobre los dos proveedores.

Tenga en cuenta que el balanceo no será perfecto, ya que se basa en rutas, y las rutas están en caché. Esto significa que las rutas usadas más a menudo siempre irán sobre el mismo proveedor.

3.3.4 ENCAPSULACIÓN DE RUTEO GENÉRICO (GENERIC ROUTING ENCAPSULATION, GRE)

Hay tres tipos de túneles en Sistema Operativo Linux: los túneles IP sobre IP, los túneles GRE y túneles que se realizan fuera del núcleo, por ejemplo, PPTP.

3.3.4.1 Túneles IP sobre IP

Es el túnel con mayor trayectoria y uso en Sistema Operativo Linux, precisa dos módulos del núcleo, ipip.o y new_tunnel.o.

Para explicar el funcionamiento de este tipo de túneles supongamos que tenemos tres redes: las redes internas A y B, y una red intermedia C.

La red A se identifica con los siguientes parámetros:

network 10.0.1.0
netmask 255.255.255.0
gateway 10.0.1.1

La puerta de enlace predeterminada tiene la dirección 172.16.17.18 en la red C.

La red B tiene los siguientes parámetros:

network 10.0.2.0
netmask 255.255.255.0
gateway 10.0.2.1

En donde el gateway tiene la dirección 172.19.20.21 en la red C.

Hasta el momento la red C se asume que dará paso a cualquier paquete que vaya de la red A hacia la red B y viceversa.

Con este panorama debemos proseguir de la siguiente forma:

Primero debemos asegurarnos de que los módulos necesarios para el túnel estén instalados:

```
insmod ipip.o
insmod new_tunnel.o
```

A continuación, configuramos el ruteador de la red A de la siguiente manera:

```
ifconfig tunl0 10.0.1.1 pointopoint 172.19.20.21
route add -net 10.0.2.0 netmask 255.255.255.0 dev tunl0
```

Y en el de la red B:

```
ifconfig tunl0 10.0.2.1 pointopoint 172.16.17.18
route add -net 10.0.1.0 netmask 255.255.255.0 dev tunl0
```

El momento que se deja de usar el túnel debemos configurar el túnel de la siguiente manera:

```
ifconfig tunl0 down
```

Con esta configuración en cada ruteador lo que hacemos es simplemente conectar dos redes IPv4 que normalmente no podrían comunicarse entre ellas. Cabe recalcar que los túneles IP-sobre-IP de Sistema Operativo Linux no funcionan con otros sistemas operativos o ruteadores.

3.3.4.2 Túneles GRE (Generic Routing Encapsulation)

GRE es un protocolo para el establecimiento de túneles a través de Internet, desarrollado por CISCO, y que está definido en los RFC 1701 y RFC 1702, el cual nos permite transportar tráfico multicast e IPv6, lo cual no se puede hacer con los túneles IP-sobre-IP.

Para realizar esto se necesita que el módulo ip_gre.o, este instalado en el Sistema Operativo Linux.

3.3.5 DISCIPLINAS DE COLAS (QDISCS) PARA GESTIÓN DEL ANCHO DE BANDA.

Las versiones del Kernel de Sistema Operativo Linux 2.2/2.4 vienen con las herramientas necesarias para gestionar el ancho de banda en formas comparables a los sistemas dedicados de alto nivel para gestión de ancho de banda.

Para evitar confusiones, debemos conocer que el comando *tc*, que se utiliza para llevar a cabo las funciones del control de tráfico, hace referencia al subsistema de colas de paquetes en una red o dispositivo de red, usa las siguientes reglas para la especificación de ancho de banda:

mbps = 1024 kbps = 1024 * 1024 bps => byte/s
mbit = 1024 kbit => kilobit/s.
mb = 1024 kb = 1024 * 1024 b => byte
mbit = 1024 kbit => kilobit.

Internamente, los números se almacenan en bps y bytes.

3.3.5.1 Las colas y disciplinas de cola explicadas

Con el encolamiento podemos controlar la manera en que se envían los datos. Es importante recalcar que solo tenemos control sobre los datos que enviamos más no sobre lo nos envían otros usuarios de la red Internet, un ejemplo más cotidiano sería como el buzón de correo que se utiliza para las cartas.

Sin embargo, la red Internet se basa en TCP/IP, la cual tiene algunas características que nos pueden ayudar.

A breves rasgos podemos resumir el comportamiento de TCP/IP de la siguiente manera: TCP/IP no tiene manera de saber la capacidad de la red entre dos sistemas, de manera que simplemente empieza a enviar datos más y más rápido, al comienzo lo hace de forma lenta, y cuando se empiezan a perder paquetes reduce la marcha.

Si tiene un ruteador y desea evitar que ciertas computadoras dentro de su red descarguen demasiado rápido, necesita dar forma (shape) a la interfaz *interna* del ruteador que envía los datos a sus computadores.

Por ejemplo si tiene una tarjeta de red de 100Mbps y un ruteador con un enlace de 256kbps, tiene que asegurarse de que no envía más datos de los que el ruteador puede manejar. Por otro lado, será el ruteador el que controle el enlace y ajuste el ancho de banda disponible. Necesitamos «poseer la cola» por decirlo así, y ser el enlace más lento de la cadena. Por suerte, esto es muy posible.

3.3.5.2 Disciplinas de cola simples, sin clases

Las disciplinas de cola sin clases son aquellas que por lo general aceptan datos y se limitan a reordenarlos, retrasarlos, o descartarlos.

Esto se puede usar para ajustar el tráfico de una interfaz entera, sin subdivisiones.

La disciplina más usada es la **qdisc pfifo_fast**, que es la disciplina por defecto en el Sistema Operativo Linux.

Esto también explica por qué estas características avanzadas son tan robustas.

Las disciplinas de cola sin clase no son más que simplemente otra cola.

Cada una de estas colas tiene puntos fuertes y debilidades específicos.

3.3.5.2.1 pfifo_fast

Como su nombre indica, First In, First Out (el primero que entra es el primero que sale), ningún paquete recibe un tratamiento especial.

Esta cola tiene 3 de lo que llamamos «bandas» ó grupos, que corresponden a las divisiones que se realiza de acuerdo a la prioridad determinada por el campo ToS, y que se detalla en la tabla 3.3. Dentro de cada banda, se aplican las reglas FIFO. Sin embargo, no se procesará la banda 1 mientras haya paquetes esperando en la banda 0. Lo mismo se aplica para las bandas 1 y 2.

El núcleo obedece el campo Type of Service que hay en cada paquete, y maneja cuidadosamente los paquetes marcados con mínimo retardo colocándolos en la banda 0.

No podemos configurar la disciplina de cola qdisc pfifo_fast ya que es el la cola por defecto fija, y la configuración por defecto es la siguiente:

priomap

Determina el orden de las prioridades de los paquetes y la manera en que el núcleo asigna los paquetes a cada banda. La asignación de la prioridad se basa en el octeto TOS de cada paquete, el campo TOS se muestra en el GRÁFICO 3.10:

```
   0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+-----+
|                 |                       |     |
|   PRECEDENCE    |          TOS          | MBZ |
|                 |                       |     |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

GRÁFICO 3.10 Campo TOS

La prioridad que se le da a cada paquete, según los cuatro bits del campo TOS, se muestra en la tabla 3.2:

Tabla 3.2 Correspondencia de la prioridad según el campo ToS

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

El comando *Tcpdump -v -v* muestra el byte en el que se encuentra el campo TOS completo, no sólo sus cuatro bits, que corresponde al valor de la primera columna que se indica en la tabla 3.3.

Tabla 3.3 Valores de TOS y sus correspondientes bandas

TOS	Bits	Significado	Prioridad Sistema Operativo Linux		Banda
00000000	0	Servicio normal	0	Mejor esfuerzo	1
00000010	1	Minimizar coste monetario	1	Relleno	2
00000100	2	Maximizar fiabilidad	0	Mejor esfuerzo	1
00000110	3	mmc+mr	0	Mejor esfuerzo	1
00001000	4	Maximizar transferencia	2	En masa	2
00001010	5	mmc+mt	2	En masa	2
00001100	6	mr+mt	2	En masa	2
00001110	7	mmc+mr+mt	2	En masa	2
00010000	8	Minimizar retrasos	6	Interactivo	0
00010010	9	mmc+md	6	Interactivo	0
00010100	10	mr+md	6	Interactivo	0
00010110	11	mmc+mr+md	6	Interactivo	0
00011000	12	mt+md	4	Int. En masa	1
00011010	13	mmc+mt+md	4	Int. En masa	1
00011100	14	mr+mt+md	4	Int. En masa	1
00011110	15	mmc+mr+mt+md	4	Int. En masa	1

La segunda columna contiene el valor de los cuatro bits TOS relevantes, seguidos por su significado traducido. Por ejemplo, el 15 significa que un paquete espera un Mínimo costo monetario, la Máxima fiabilidad, la Máxima transferencia y un Retraso mínimo.

La cuarta columna indica la manera en que el núcleo Sistema Operativo Linux interpreta los bits del TOS, mostrando qué prioridad les asigna.

La última columna indica el resultado del *priomap* por defecto. Esto significa, por ejemplo, que a la prioridad 4 se le asigna la banda número 1. El priomap también le permite listar prioridades mayores que no se corresponden a asignaciones del TOS (como el número 7), sino que se configuran por otros medios.

El siguiente listado de aplicaciones, indica como cada aplicación debería activar los bits TOS:

TELNET

1000 (minimizar retraso)
FTP
Control 1000 (minimizar retraso)
Datos 0100 (maximizar transferencia)
TFTP
1000 (minimizar retraso)
SMTP
Fase de órdenes 1000 (minimizar retraso)
Fase de datos 0100 (maximizar transferencia)
Domain Name Service
Consulta UDP 1000 (minimizar retraso)
Consulta TCP 0000
Transf. De zona 0100 (maximizar transferencia)
NNTP
0001 (minimizar coste monetario)
ICMP

Errores	0000
Peticiones	0000 (la mayoría)
Respuestas	igual que las peticiones (la mayoría)

txqueuelen:

La longitud de esta cola se obtiene de la configuración de la interfaz, que se puede ver y modificar con ifconfig o ip. Por ejemplo para establecer la longitud de la cola a 10, se debe ejecutar la siguiente línea de comando:

ifconfig eth0 txqueuelen 10.

3.3.5.2.2 Token Bucket Filter

El Token Bucket Filter (TBF) es una disciplina de cola qdisc sencilla que se limita a dejar pasar paquetes que lleguen a una velocidad que no exceda una impuesta administrativamente, pero con la posibilidad de permitir ráfagas cortas que excedan esta tasa.

La implementación de TBF consiste en un buffer (el bucket o balde), que se llena constantemente con piezas virtuales de información denominadas tokens, a una velocidad específica (token rate), los tokens corresponden a bytes. El parámetro más importante del bucket es su tamaño, que corresponde al número de tokens que puede almacenar.

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

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

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

- Los datos llegan al TBF a una velocidad *mayor* a la de los tokens. Esto significa que el bucket se quedará pronto sin tokens, lo que causará que el TBF se acelere a sí mismo por un rato, a esto se le llama una situación sobre límite. Si siguen llegando paquetes, empezarán a ser descartados.

Esta última situación es muy importante, porque permite ajustar administrativamente el ancho de banda disponible a los datos que están pasando por el filtro.

La acumulación de tokens permite ráfagas cortas de datos extralimitados para que pasen sin pérdidas, pero cualquier sobrecarga restante causará que los paquetes se vayan retrasando constantemente, y al final sean descartados.

La implementación de esta disciplina de cola se realiza mediante algunos controles, de los cuales los que siempre están disponibles son los siguientes:

limit o latency:

limit es el número de bytes que pueden ser encolados a la espera de que haya tokens disponibles. También se puede especificar esto estableciendo el parámetro latency, el cual indica el periodo máximo de tiempo que puede pasar un paquete en el TBF. Este último cálculo tiene en cuenta el tamaño del bucket, la velocidad y posiblemente el peakrate (la tasa de picos, si se ha configurado).

burst/buffer/maxburst:

Tamaño del bucket, en bytes. Esta es la máxima cantidad de bytes para los que puede haber tokens disponibles instantáneamente. En general, grandes velocidades precisan grandes búferes. Por ejemplo para una velocidad de 10Mbps y sobre un procesador Intel, necesitará al menos un búfer de 10Kbyte. Si el búfer es demasiado pequeño, se descartarán paquetes debido a que llegan más tokens por tick del temporizador de los que caben en el bucket.

mpu:

Un paquete de tamaño cero no usa un ancho de banda cero. En ethernet, ningún paquete usa menos de 64 bytes. La Minimum Packet Unit (mpu) determina el uso mínimo de tokens por paquete.

rate:

El ajuste de la velocidad. Si el paquete contiene tokens y se le permite estar vacío, por defecto tendrá velocidad infinita. Si esto no es aceptable, use los siguientes parámetros:

peakrate:

Si hay tokens disponibles, y llegan paquetes, por defecto se envían inmediatamente, a velocidad infinita. La velocidad de picos se puede usar para especificar cuán rápido se le permite al bucket vaciarse.

Sin embargo, debido a la resolución por defecto de 10ms del temporizador de Unix, con paquetes de 10.000 bits, en promedio, estaremos limitados a una tasa de picos de:

$$\frac{10000\,bit}{10\,ms} = \frac{10000\,bit}{0.01\,s} = 1000000\,bit/s = 1Mbps$$

mtu/minburst:

La peakrate de 1Mbps no es muy útil si la velocidad normal es mayor que 10.000 bits. Es posible tener una velocidad de picos mayor enviando más paquetes por fracción del temporizador, lo que significa de forma efectiva que hemos creado un segundo bucket, este segundo bucket contiene por defecto un único paquete, por lo que no es un bucket realmente.

Para tener un mejor panorama acerca de esta disciplina de cola realizaremos una configuración de ejemplo.

Esta es una configuración sencilla pero muy útil:

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

Decimos que es útil ya que si tenemos un dispositivo de red con una cola grande, como un módem para DSL o un cable módem, y queremos comunicarnos con él mediante un dispositivo rápido, como una interfaz ethernet, nos encontraremos conque enviar cualquier cosa destruye completamente la interactividad.

Esto se debe a que enviar datos llena la cola del módem, que probablemente es enorme porque realmente ayuda a conseguir una buena transferencia de datos al enviar. Pero esto no es lo que queremos; lo que necesitamos es tener una cola no tan grande de manera que la interactividad se mantenga y de esta forma podamos hacer otras cosas mientras se envían los datos.

La línea anterior reduce los envíos a una tasa que no conlleve a formar colas en el módem, la cola estará en Sistema Operativo Linux, donde podemos ajustarla a un tamaño limitado.

3.3.5.3 Disciplina de cola con Clases.

Las **qdisc** con clases son muy útiles si tenemos diferentes tipos de tráfico a los que se quiere dar un tratamiento separado.

3.3.5.3.1 El flujo dentro de las Disciplinas de colas con clases y sus clases.

Cuando entra tráfico dentro de una **qdisc** con clases, hay que enviarlo a alguna de las clases que contiene se necesita *clasificarlo*. Para determinar qué hay que hacer con un paquete, se consulta a los *filtros*. Los filtros asociados a esa **qdisc** devuelven entonces una decisión, y la **qdisc** la usa para encolar el

paquete en una de las clases. Cada subclase puede probar otros filtros para ver si se imparten más instrucciones. En caso contrario, la clase encola el paquete en la **qdisc** que contiene.

Aparte de contener otras **qdisc**, la mayoría de las **qdisc** con clases también realizan el conformado de los paquetes. Esto es útil tanto para reordenar paquetes, como para controlar la velocidad.

3.3.5.3.2 La familia qdisc: raíces, controladores, hermanos y padres

Cada interfaz tiene una *qdisc raíz* de salida, que por defecto es la disciplina de colas pfifo_fast sin clases que se mencionó anteriormente. A cada qdisc y clase se le asigna un controlador *handle*, que puede usar en posteriores sentencias de configuración para referirse a la qdisc. Aparte de la qdisc de salida, la interfaz también puede tener una de entrada, que dicta las normas sobre el tráfico que entra.

Los controladores de estas qdisc consisten en dos partes, un número mayor y un número menor: <mayor>:<menor>. Es costumbre darle a la qdisc de raíz el nombre «1:», que es lo mismo que «1:0». El número menor de una qdisc siempre es 0.

Las clases deben tener el mismo número mayor que sus padres. Este número mayor tiene que ser único dentro de una configuración de salida o entrada. El número menor debe ser único dentro de una qdisc y sus clases.

Una jerarquía típica de clases, puede ser como la que se muestra en el GRÁFICO 3.11:

```
          1: qdisc raíz
           |
          1:1   clase hija
        /  |  \
       /   |   \
      /    |    \
   1:10  1:11  1:12   clases hijas
    |      |     |
    |     11:    |    clase terminal
    |            |
    10:         12:   qdisc
   /   \       /   \
10:1  10:2   12:1  12:2   clases terminales
```

GRÁFICO 3.11 Jerarquía Típica de clases

En este esquema un paquete se clasifica en una cadena como ésta:

1: -> 1:1 -> 12: -> 12:2

Ahora el paquete reside en una cola de una **qdisc** asociada a la clase 12:2. En este ejemplo, se asocia un filtro a cada nodo del árbol, y cada cual escoge qué rama toma en su paso. Sin embargo, también es posible el siguiente camino:

1: -> 12:2

En este caso, un filtro asociado a la raíz decidió enviar el paquete directamente a 12:2.

Cuando el núcleo decide que necesita extraer paquetes para enviarlos a la interfaz, la qdisc 1: raíz recibe una petición de desencolar, que se pasa a 1:1, que a su vez la pasa a 10:, 11:, y 12:, cada una de las cuales consulta a sus descendientes, e intenta hacer dequeue (desencolamiento) sobre ellos. En este caso, el núcleo necesita recorrer todo el árbol, porque sólo 12:2 contiene un paquete.

Como podemos observar las clases anidadas solo hablan a sus **qdisc** paternas, y nunca a la interfaz. Sólo la **qdisc** raíz recibe peticiones de

desencolado del núcleo, la consecuencia de esto es que las clases nunca desencolan más rápido de lo que sus padres permiten. De esta manera se puede tener SFQ como clase interna, que no hace ajustes, sólo reordena, y tenemos una **qdisc** externa, que es la que hace los ajustes.

Dentro de las qdisc con clases, algunas de las más utilizadas son las siguientes:

- PRIO
- CBQ
- HTB

3.4 NETFILTER E iproute (MARCADO DE PAQUETES)

Hasta ahora hemos visto cómo funciona **iproute**, Netfilter nos permite filtrar paquetes, o cambiar sus cabeceras. Una capacidad especial es que podemos marcar un paquete con un número. Esto se hace con la estructura *--set-mark*.

Por ejemplo, esta orden marca todos los paquetes destinados al puerto 25, correo saliente:

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

Ahora digamos que tenemos varias conexiones, una que es rápida por lo tanto más cara, y otra más lenta pero de menor costo. Lo aconsejable es que hacer salir el correo mediante la ruta más barata.

Utilizaremos el mismo ejemplo anterior para ilustrar esto, ya hemos marcado los paquetes con un «1», y ahora le daremos instrucciones a la base de normas de enrutamiento para que actúe sobre esto:

```
# echo 201 mail.out >> /etc/iproute2/rt_tables
# ip rule add fwmark 1 table correo.salida
# ip rule ls
0: from all lookup local
32764: from all fwmark 1 lookup correo.salida
32766: from all lookup main
32767: from all lookup default
```

Ahora generaremos la tabla correo.salida con una ruta al enlace lento pero barato:

```
# /sbin/ip route add default via 195.96.98.253 dev ppp0 table# correo.salida
```

Con esto está listo. Si quisiéramos hacer algunas excepciones, hay muchas maneras de conseguirlo. Podemos modificar la orden de netfilter para excluir ciertas computadoras, o podemos insertar una regla con mayor prioridad que apunte a la tabla principal para dichas computadoras.
También podemos usar estas características para obedecer a los bits TOS marcando los paquetes con un tipo diferente de servicio con diferentes números, y creando reglas que actúen sobre esto. De esta manera podría dedicar, una línea RDSI a las sesiones interactivas.

Pero como todo en Sistema Operativo Linux esta opción de marcado de paquetes necesita activarse en el núcleo:

```
IP: advanced ruteador (CONFIG_IP_ADVANCED_RUTEADOR) [Y/n/?]
IP: policy routing (CONFIG_IP_MULTIPLE_TABLES) [Y/n/?]
IP: use netfilter MARK value as routing key (CONFIG_IP_ROUTE_FWMARK) [Y/n/?]
```

Con estas instrucciones el equipo queda listo para utilizar NETFILTER.

CAPÍTULO 4

APLICACIÓN DEL PROTOCOLO DIFFSERV EN REDES DE VoIP

4.1 INTRODUCCIÓN

En este capítulo llevaremos a cabo el levantamiento de los servicios diferenciados en un servidor Sistema Operativo Linux para aplicarlo a una red ya establecida, la cual se detalla más adelante. La implementación de Diffserv se puede realizar con versiones de kernel a partir de la 2.2, la cual ya cuenta con el conjunto de herramientas necesarias para llevar a cabo la implementación.

Cabe indicar que el análisis que realizaremos solo serán ejemplos de redes ya establecidas, en las que se garantiza, que la voz esté dentro de parámetros adecuados, que sean aceptables para el receptor, todo esto sin tener que realizar un incremento en el ancho de banda ya que el objetivo principal es brindar un nuevo servicio sin necesidad de incrementar los recursos de la red.

Para ello se lleva a cabo el levantamiento de Diffserv mediante la línea de comandos del Sistema Operativo Linux, utilizando las herramientas provistas por el núcleo de Sistema Operativo Linux como iproute2 y netfilter.

4.2 ¿POR QUÉ NECESITAMOS QoS EN LAS REDES IP?[20]

Las redes IP reparten paquetes con un tipo de servicio conocido como “best effort” (BE), lo cual equivale a “lo más posible, lo antes posible”. Los paquetes con este tipo de servicio tienen la misma expectativa de tratamiento a medida que transitan la red. Se caracteriza porque la complejidad se encuentra en los “hosts” de los extremos, convirtiendo a los ruteadores del núcleo de la red en “tontos”. Sólo miran la cabecera del paquete recibido, buscan en la tabla de ruteo y definen el siguiente salto.

Si llegase a ocurrir congestión, se retardan o descartan los paquetes. Esto hace muy escalable la red. Es suficiente para aplicaciones como mail, ftp y navegación, pero no para otras aplicaciones que no toleran retardos variables o pérdida de datos, como es el caso de servicios de voz y video en tiempo real.

Hay una convergencia de servicios no tradicionales: telefonía, radio, televisión, video conferencia, etc., los cuales tienen otras exigencias. Una solución se podría pensar, es agregar más ancho de banda, pero esto no es suficiente, ya que el tráfico es típicamente en ráfagas, produciendo congestiones temporales y retardos y pérdidas.

Por lo tanto la clave está en dotar a Internet de la capacidad de diferenciar los distintos tipos de paquetes de datos, que la atraviesan, para obtener QoS. El objetivo de la Calidad de Servicio en una red es cuantificar el tratamiento que un paquete debe esperar a medida que circula por la red. El objetivo de una QoS diferenciada, es el dar a ciertos paquetes un mejor trato y a otros un peor trato.

- [20] Adrián Delfino y Sebastián Rivero, Diffserv: Servicios Diferenciados Monografía de Evaluación de Performance en Redes de Telecomunicaciones, Pág. 7, http://iie.fing.edu.uy/ense/asign/perfredes/trabajos/trabajos_2003/diffserv/Trabajo%20Final.pdf

Hay que tener en cuenta que QoS no puede crear ancho de banda adicional, sino que debe procurar manejar el tráfico de mejor manera, de tal forma que con el ancho de banda disponible se soporte las necesidades de un amplio rango de aplicaciones que necesitan un mejor tratamiento que lo que nos brinda el sistema actual best effort.

En el GRÁFICO 4.1, se indica la relación de la latencia en función del ancho de banda para diversas aplicaciones.

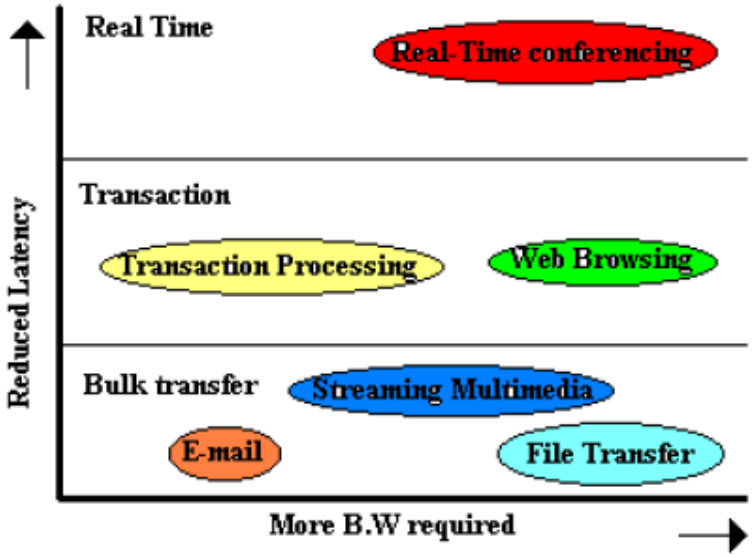


GRÁFICO 4.1 Diagrama de latencia en función del BW para diversas aplicaciones.

Los proveedores de servicios en general, publican un Acuerdo del Nivel de Servicio (SLA), para sus servicios, que contiene los niveles de calidad medibles, que un cliente espera del tráfico de la clase en particular que se encuentra manejando.

Aproximadamente hace una década, la llegada del tráfico de paquetes de voz y video, motivó ciertos intentos de crear distintos niveles de servicio para el tráfico de paquetes.

En la actualidad, razones económicas han surgido para diferenciar el tráfico; Internet se ha puesto mucho más en uso y se ha convertido en una misión crítica para ciertas compañías.

Las mayores aplicaciones de diferenciación de tráfico que se esperan son las siguientes:

- Distinguir la importancia del tráfico dentro de una nube de red
- Permitir una buena calidad de voz sobre redes IP y
- Permitir a los proveedores de servicios, el vender servicios que compitan con líneas alquiladas.

4.2.1 QoS EN REDES IP

El modelo DiffServ (servicios diferenciados), el cual proporciona QoS (Calidad de Servicios), se basa en separar los paquetes de datos, en agregados de tráfico 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.

4.3 DIFFSERV Y CONTROL DE TRÁFICO EN EL SISTEMA OPERATIVO LINUX[21]

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

Para manejar mecanismos de Calidad de Servicio, los nuevos kernels del Sistema Operativo 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.

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 una computadora con Sistema Operativo Linux. La política por defecto es FIFO, pero existen otras basadas en prioridades, Round Robin, RED, DSMARK, entre otras.

- Las clases contenidas en las disciplinas de cola, permiten realizar separaciones del tráfico para, por ejemplo, aplicar una política diferente a cada clase.

[21]Bert Hubert, "Enrutamiento Avanzado y Control de Tráfico en Sistema Operativo Linux", http://www.redes-linux.com/manuales/routing/Enrutamiento-avanzado-y-control-de-trafico-en-Linux.pdf

③ Filtros o clasificadores: cuya función es asignar el tráfico a las distintas clases en función del criterio de selección que se especifique.

4.3.1 DISCIPLINAS CON CLASES

Las disciplinas de colas con clases son aquellas que permiten anidar otras disciplinas a ellas. De esta forma pueden darse complicadas jerarquías de disciplinas en forma de árbol. A partir de una disciplina de cola con clases (llamada normalmente raíz) se pueden anidar otras disciplinas con clases y a su vez dentro de estas se pueden seguir produciendo anidamientos.

4.4 DSMARK[22]

Es una disciplina de colas definida para implementar los servicios diferenciados en Sistema Operativo Linux. En rigor no es una disciplina de colas, aunque se trate como una de ellas, sino un marcador de paquetes en su campo DSCP. Su comportamiento es muy sencillo en comparación con otras disciplinas de cola del Control de Tráfico de Sistema Operativo Linux.

Es importante aclarar que al contrario de otras disciplinas de cola, DSMARK no da forma o controla el tráfico. Tampoco prioriza, retarda, reordena o descarta paquetes. Solo marca los paquetes usando el campo DS, esto quiere decir, que al contrario que PRIO o HTB, que garantizan una QoS en términos de latencia y ancho de banda en función de la prioridad de cada clase, si se emplea DSMARK sin ninguna otra disciplina como HTB o PRIO, su único cometido es el marcado o remarcado del TOS, para usar este posteriormente dentro de la red, como identificador de clase y por tanto, como identificador de prioridad.

[22] http://www.opalsoft.net/QoS/DS.htm, Differentiated Service on Sistema Operativo Linux HOWTO, Pág. 99

En el GRÁFICO 4.2, se indica el diagrama de bloques tradicional de la disciplina de cola DSMARK.

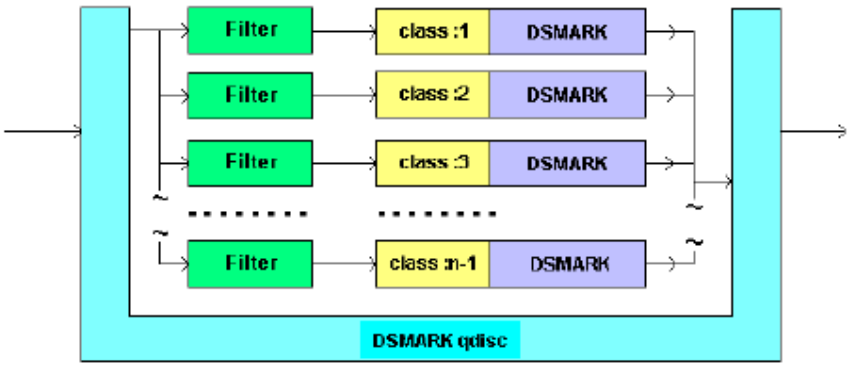


GRÁFICO 4.2 Diagrama Tradicional de DSMARK

4.4.1 TRABAJANDO CON DSMARK

Como se especificó en el capítulo 2, que se centró en el funcionamiento de DiffServ, indicamos de dos tipos de nodos: nodos externos (o limítrofes) e internos. Son dos puntos importantes en el camino del tráfico. Ambos tipos realizan una clasificación del paquete cuando llega al nodo, mediante la asignación de un código DSCP, el cual puede ser usado en diferentes lugares a lo largo del proceso de Servicios Diferenciados antes de que se envíe el paquete a la red, por lo que es necesario almacenar el resultado de la clasificación original.

Para llevar a cabo este almacenamiento, la disciplina de cola DSMARK, proporciona una estructura llamada **sk_buff**, que incluye un campo nuevo llamado **skb->tc_index** donde se almacena el resultado de la clasificación inicial que puede usarse como ya dijimos en varios momentos del tratamiento DS.

El valor de **skb->tc_index** lo establecerá inicialmente la qdisc DSMARK, sacándola del campo DS de la cabecera IP de cada paquete recibido. Aparte, el clasificador **cls_tcindex** leerá todo o parte del valor de **skb->tcindex** y lo usará para escoger las clases.

Para entender de mejor manera esto, veamos la orden de la qdisc DSMARK y los parámetros que en ella se manejan, los cuales se indican en el GRÁFICO 4.3.

```
# tc add qdisc dev eth0 handle <hd> root dsmark \
    indices <id> [ default_index <did> ] [ set_tc_index ]
```

GRÁFICO 4.3 Estructura del comando DSMARK

En donde:

- ***índices***: tamaño de la tabla de pares (máscara, valor). El tamaño máximo es 2^n, siendo n>=0.
- ***default_index***: La entrada por defecto del índice de la tabla si el clasificador no encuentra coincidencia.
- ***Set_tc_index***: instruye a la disciplina dsmark para que obtenga el campo DS y lo almacene en **skb->tc_index**.

Como podemos ver DSMARK es una disciplina cola llena de clases, las cuales son numeradas desde 1, 2, 3, ……., ***n***-1, donde ***n*** es un parámetro que define el tamaño de la tabla interna requerida para implementar el comportamiento de la disciplina de cola. Este parámetro es conocido como índice y es definido según los requerimientos del usuario.

Llamemos ***q*** al número de cola, el elemento ***q:0*** es el elemento principal de cola que realmente importa. Los elementos desde ***q:1*** hasta ***q:n-1*** son las clases de la disciplina de cola (representados en el GRÁFICO 4.2 como bloques de color amarillo). Cada una de estas colas es seleccionada mediante

el uso de un filtro (representados en el GRÁFICO 4.2 como bloques de color verde) vinculado a la disciplina de cola, es decir los paquetes seleccionados por el filtro son ubicados en la respectiva clase DSMARK.

4.4.2 ¿CÓMO Y DÓNDE SE MARCAN LOS PAQUETES CON DSMARK?[23]

Los paquetes son marcados en el campo DS con un valor entero, que se definen para cada clase cuando el usuario configura la disciplina de cola. Los paquetes son marcados antes de que abandonen la disciplina de cola para ser ubicados en el dispositivo de red de salida.

Para crear una nueva disciplina de cola DSMARK usamos el comando ya mostrado en el GRÁFICO 4.3.

```
# tc add qdisc dev eth0 handle <hd> root dsmark \
    indices <id> [ default_index <did> ] [ set_tc_index ]
```

GRÁFICO 4.3 Estructura del comando DSMARK

Los parámetros marcados con color Amarillo indican los valores que el usuario debe ingresar.

A continuación indicamos lo expresado anteriormente: este comando setea nuestro DSMARK como la ruta de cola de interfaz ethernet 0 de nuestro servidor.

- ③ *<hd>* es el parámetro que maneja el número de la cola.
- ③ *<id>* es el tamaño de la tabla interna que define el número de clases (id-1) contenido en la cola.

[23] http://www.opalsoft.net/QoS/DS.htm, Differentiated Service on Sistema Operativo Linux HOWTO, Pág. 101

③ <*did*> es el índice por defecto que maneja los paquetes cuando no se encuentran configurados en ninguna clase, este parámetro es opcional. El

parámetro ***set_tc_index*** será detallado más adelante.

Consideremos el ejemplo mostrado en el GRÁFICO 4.4, para explicar como se configura los parámetros utilizados en la creación de una nueva disciplina de cola DSMARK.

```
# tc qdisc add dev eth0 handle 1:0 root dsmark indices 32
```

GRÁFICO 4.4 Configuración básica de DSMARK

Este comando crea una disciplina de cola DSMARK que tiene como ruta de salida la interfaz ethernet 0. La disciplina de cola es numerada como 1:0 y contiene una tabla de 32 elementos, desde el elemento 0 hasta el elemento 31, los elementos 1 al 31 se pueden utilizar como clases desde 1:1 hasta 1:31.

En el GRÁFICO 4.5, se indica una configuración anidada de clases contenidas en una disciplina de cola DSMARK:

```
# tc qdisc add dev eth0 parent 1:1 handle 2:0 dsmark \
    indices 8 default_index 7
```

GRÁFICO 4.5 Configuración anidada de clases dentro de una cola DSMARK

Este comando crea una disciplina de cola DSMARK, que tiene configurada la clase 1:1 (de otra disciplina de cola) como padre. La disciplina de cola dentro de la clase 1:1 es numerada como 2:0 y contiene una tabla de 8 elementos. El índice por defecto está numerado como 7, que corresponde a la clase DSMARK 2:7. Todo este ejemplo nos muestra que cada clase dentro de una disciplina de cola puede contener un número infinito de anidamientos de colas.

Una vez explicado como se configura las colas y sus respectivas clases, centrémonos en el proceso de marcado de paquetes, para lo cual nos ayudaremos de una representación esquemática de la tabla interna de DSMARK, la cual se indica en el GRÁFICO 4.6.

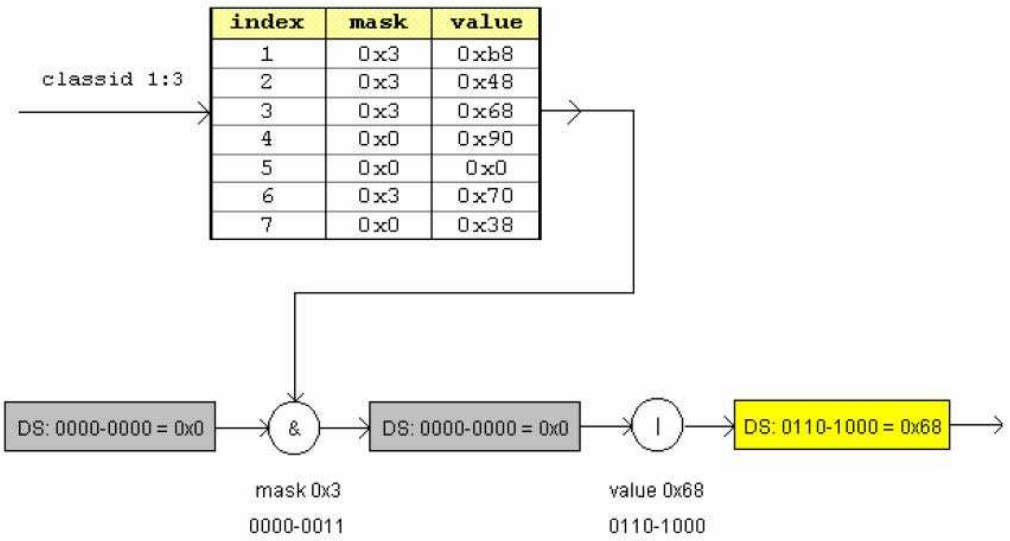


GRÁFICO 4.6 Tabla interna de la disciplina de cola DSMARK.

En el GRÁFICO 4.6, podemos observar que los datos, ingresan por la izquierda con un número de clase correspondiente a un valor de índice, que en este caso es 1:3, asumiendo que la disciplina de cola DSMARK raíz, es 1:0. La tabla interna tiene dos columnas denominadas *máscara* y *valor*, que contienen valores enteros hexadecimales.

Los valores enteros seleccionados según el índice son: máscara = 0x3 y valor = 0x68.

Una vez determinados estos dos valores, la disciplina de cola DSMARK extrae del paquete el valor entero del campo DS para aplicar la siguiente operación:

Nuevo_DS = (Anterior_DS & máscara) | valor

En donde **“&”** y **“|”** corresponden a los operadores de las funciones **and** y **or** respectivamente. Este nuevo valor del campo DS es puesto de regreso en el campo DS del paquete.

Como podemos ver en el bloque gris de la izquierda del GRÁFICO 4.6, un paquete está entrando, con el campo DS seteado con el valor 0x0, que corresponde al DSCP de mejor esfuerzo.

DSMARK se encarga de que el paquete deje la cola con su campo DS marcado con el valor 0x68 correspondiente al DS de Tránsito Asegurado con clase AF 31.

Nos preguntaremos el porque se escoge el valor de máscara = 0x3, esta operación se hace con el fin de preservar los dos bits más significativos de la derecha, para mantener intacta la condición ECN desde el inicio.

El valor 0x68 que sirve para realizar la operación *or,* nos sirve para setear el valor del campo DS del paquete de salida en 0x68.

A continuación explicaremos el procedimiento para llenar la tabla interna de la disciplina de cola DSMARK y como se relacionan los valores del índice y el par máscara – valor.

Para esto es útil representar tanto el valor binario y héxadecimal de los campos DSCP y DS, estos valores se encuentran indicados en la tabla 4.1.

Tabla 4.1 Valores binarios y hexadecimales de DSCP y DS

CLASE	DP (Discard Precedence)	DSCP	b-DSCP	h-DSCP	b-DS	h-DS
AF1	1	001010	0000-1010	0xa	0010-1000	0x28
	2	001100	0000-1100	0xc	0011-0000	0x30
	3	001110	0000-1110	0xe	0011-1000	0x38
AF2	1	010010	0001-0010	0x12	0100-1000	0x48
	2	010100	0001-0100	0x14	0101-0000	0x50
	3	010110	0001-0110	0x16	0101-1000	0x58
AF3	1	011010	0001-1010	0x1a	0110-1000	0x68
	2	011100	0001-1100	0x1c	0111-0000	0x70
	3	011110	0001-1110	0x1e	0111-1000	0x78
AF4	1	100010	0010-0010	0x22	1000-1000	0x88
	2	100100	0010-0100	0x24	1001-0000	0x90
	3	100110	0010-0110	0x26	1001-1000	0x98
EF		101110	0010-1110	0x2e	1011-1000	0xb8

Para formar esta tabla utilizamos los valores de DSCP, definidos en el capítulo 2 en la tabla 2.2, y a continuación adicionamos dos bits cero a la izquierda del DSCP para obtener el valor b-DSCP, luego adicionamos dos bits cero a la derecha para obtener el valor b-DS, los valores hexadecimales se obtienen de hacer la respectiva conversión de cada uno.

Con esta corta explicación, y teniendo los valores de máscara y valor, podemos resolver cualquier tipo de problema, por ejemplo:

Si queremos que cualquier paquete que ingrese a la disciplina de cola salga con una clase DS AF23, debemos utilizar en nuestra tabla interna una máscara = 0x0 y un valor = 0x58.

Los valores de máscara y valor se obtienen de analizar el resultado que deseamos obtener, para este caso deseamos que todos los paquetes que ingresen con cualquier valor de DSCP sean cambiados a un valor de DSCP AF23. Supongamos que ingresa un paquete con DSCP AF11, DSCP = 001010, para obtener el resultado de AF23 debemos hacer la siguiente operación:

DSCP original	AND	máscara	=			OR	valor	=	DSCP nuevo
0010-1000	&	0000-0000	= 0000-0000	→	0000-0000	|	0101-1000	=	0101-1000

Como podemos ver la única forma de obtener un AF23, es haciendo que el valor de la máscara sea 0x0, de esta manera al hacer la operación AND tenemos como resultado 0x0, y después de esto para obtener un DSCP con el valor AF23 debemos utilizar un valor de 0x58 en la operación OR.

A continuación planteamos otro ejemplo:

Deseamos cambiar la precedencia de descarte de cualquier paquete AF entrante a un valor de precedencia de descarte 2 y además queremos preservar los bits ECN.

Para implementar esto debemos tomar un valor de máscara tal que luego de hacer la operación AND nos permita conservar los tres primeros bits del DSCP, correspondientes al selector de clase AF, y conservar también los 2 bits ECN, esto se logra con el código 11100011 ó 0xe3. Lo único que nos falta es cambiar la precedencia de descarte de los paquetes a 2, esto se logra con un valor de 0x10, este valor lo obtenemos de la siguiente manera: los bits de precedencia de descarte son el 4to y el 5to respectivamente, con este antecedente podemos notar fácilmente que la única forma de mantener una precedencia de descarte es haciéndolo con un valor de 00010000, los valores en color rojo indican los bits 4 y 5 del campo DSCP que corresponden a los bits de precedencia de descarte.

Para entender de mejor manera como funciona esto, se indica a continuación un ejemplo:

Deseamos cambiar el valor de la clase de cualquier paquete AF entrante a un valor de clase 3, ósea AF3x. La precedencia de descarte y los bits ECN deben conservarse.

La solución es la siguiente: debemos conservar los bits de precedencia de descarte y los bits ECN, esto se logra haciendo que la máscara tenga el valor 1 en el cuarto y quinto bits (correspondientes a la precedencia de descarte) y en los dos últimos bits (correspondientes a los bits ECN), el resto deben ser ceros, máscara = 00011111, esto corresponde al valor máscara = 0x1f. Lo único que nos falta es lograr que la clase tenga siempre el valor de 3, para esto debemos hacer que el valor tenga siempre el segundo y tercer bits siempre en uno, esto corresponde a un valor = 01100000 ó 0x60.

Con estos ejemplos ya sabemos como formar nuestra tabla interna para la implementación de la disciplina de cola DSMARK, cabe señalar que los campos de máscara y valor siempre dependerán de las necesidades que queramos satisfacer en nuestra red.

Un ejemplo del ingreso de los valores de la tabla interna para valores de máscara y valor, que se indica en la tabla 4.2, se configuraría de la siguiente manera:

Tabla 4.2 Ejemplo de valores a ingresar en la tabla interna.

índice	Máscara	Valor
1	0x0	0xb8
2	0x3	0x58
3	0xe3	0x10
4	0x1f	0x60
5	0x0	0x70
6	0x3	0x30
7	0x0	0x0

Los comandos se muestran en el GRÁFICO 4.7, y corresponden a valores supuestos para este ejemplo:

```
#tc qdisc add dev eth0 handle 1:0 root dsmark indices 8

#tc class change dev eth0 classid 1:1 dsmark mask 0x0 value 0xb8
#tc class change dev eth0 classid 1:2 dsmark mask 0x3 value 0x58
#tc class change dev eth0 classid 1:3 dsmark mask 0xe3 value 0x10
#tc class change dev eth0 classid 1:4 dsmark mask 0x1f value 0x60
#tc class change dev eth0 classid 1:5 dsmark mask 0x0 value 0x30
#tc class change dev eth0 classid 1:6 dsmark mask 0x3 value 0x70
#tc class change dev eth0 classid 1:7 dsmark mask 0x0 value 0x0
```

GRÁFICO 4.7 Comandos de configuración para una tabla interna de 7 clases

El primer comando crea la disciplina de cola DSMARK, los siguientes siete comandos construyen la tabla interna de la disciplina de cola con sus respectivos valores de máscara y valor.

Hasta el momento hemos configurado los valores de los campos máscara y valor que nos servirán para crear la tabla interna que permite el marcado de los paquetes entrantes a la disciplina de cola DSMARK. Pero todavía nos falta determinar como se realiza la ubicación de los paquetes entrantes dentro de la clase correspondiente.

Este trabajo es realizado por el filtro vinculado a cada clase, para ello existen varios tipos de filtros que pueden ser utilizados de acuerdo a las necesidades de la red, en nuestro caso particular indicaremos el filtro clasificador u32, que detallamos a continuación.

4.4.2.1 Utilización del filtro u32 para la clasificación de los paquetes.

Sabemos que en las disciplinas de colas con clases, es necesario llevar a cabo una clasificación del tráfico. Es decir, es necesario determinar a qué clase debe ir cada paquete IP.

Para ello utilizamos el concepto de 'filtro' que asociaremos a cada disciplina de colas en la que sea necesario llevar a cabo una clasificación.

Las posibilidades a la hora de filtrar los paquetes son múltiples, incluso hay algunos tipos de filtros que son específicos de determinadas disciplinas de colas. Para nuestro proyecto de titulación haremos referencia al filtro u32, que es uno de los más significativos.

4.4.2.1.1 Filtro u32[24]

Este tipo de filtro proporciona una versatilidad enorme, ya que permite muchos criterios a la hora de llevar a cabo el filtrado, lo cual hace que sea uno de los más ampliamente utilizados. Por definición, este tipo de filtro permite filtrar en función de cualquier conjunto de bits, tanto de la cabecera del paquete IP, como de la cabecera del segmento de datos. Sin embargo, este tipo de utilización es bastante confusa y complicada, por lo que normalmente se suelen utilizar formas más directas para estos filtros. Así, algunas de estos criterios directos son:

- Dirección IP de origen y/o destino del paquete.
- Protocolo utilizado: tcp, udp, icmp, gre, etc.
- Puertos de origen y destino utilizados.
- Valor del campo TOS de la cabecera IP.

La línea de órdenes del programa tc filter, que se usa para configura el filtro, consiste en tres partes: especificación del filtro, selector y acción. La especificación del filtro puede definirse así:

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

[24] Bert Hubert, "Enrutamiento Avanzado y Control de Tráfico en Sistema Operativo Linux", Pág. 80, http://www.redes-linux.com/manuales/routing/Enrutamiento-avanzado-y-control-de-trafico-en-Linux.pdf

- El campo ***protocol*** describe el protocolo al que se aplicará el filtro.
- El campo ***preference*** (alternativamente se puede usar priority) establece la prioridad del filtro escogido. Esto es importante, porque puede tener varios filtros (listas de reglas) con diferentes prioridades. Se pasará por cada lista en el orden en que se agreguen las reglas, y entonces se procesarán las listas de menor prioridad (numero "preference" más alto).
- El campo ***parent*** define la raíz de la cola DSMARK (por ejemplo, 1:0), a la que se asociará el filtro.

Las opciones descritas anteriormente se aplican a todos los filtros, no sólo a los u32.

A continuación veamos unos ejemplos de aplicación para entender de mejor manera lo antes explicado:

```
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
match ip src 1.2.3.0/24 \
match ip dst 4.3.2.0/24 flowid 1:10
```

Este filtro seleccionará aquellos paquetes IP que tengan como dirección IP origen cualquiera de la red 1.2.3.0/24 y como dirección destino cualquiera de la red 4.3.2.0/24.

```
# tc filter add dev eth0 parent 10:0 protocol ip prio 1 u32 \
match ip src 4.3.2.1/32
match ip dport 80 0xffff flowid 10:1
```

Este filtro seleccionará aquellos paquetes IP que tenga como dirección origen 4.3.2.1 y puerto destino el 80.

4.4.3 CLASIFICADOR tcindex[25]

El clasificador *tcindex* fue específicamente diseñado para implementar la arquitectura de Servicios Diferenciados en el Sistema Operativo Linux.

[25] http://www.opalsoft.net/QoS/DS.htm, Differentiated Service on Sistema Operativo Linux HOWTO, Pág. 107

Este clasificador basa su comportamiento en el campo ***skb->tc_index,*** localizado en un buffer temporal denominado ***sk_buff***, este espacio de buffer es creado para cada paquete que entra o que va a ser generado por el propio kernel del Sistema Operativo Linux.

Este clasificador solo puede ser usado con las siguientes disciplinas de cola: GRED, DSMARK e INGRESS.

En el GRÁFICO 4.8, se muestra en forma esquemática el funcionamiento del clasificador *tcindex.*

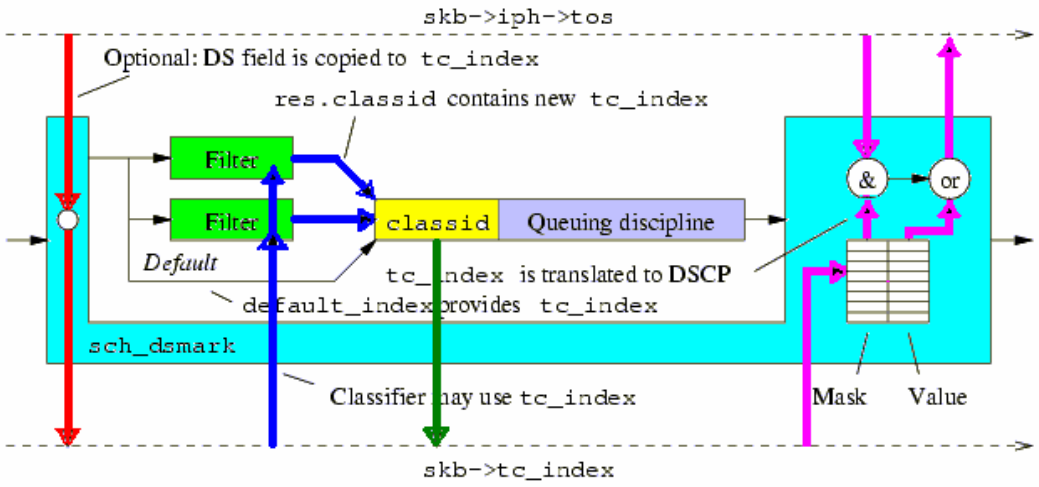


GRÁFICO 4.8 Disciplina de cola DSMARK

Se accede al buffer temporal mediante el uso de un puntero llamado ***skb***. El valor ***skb->tc_index*** es seteado por la disciplina de cola DSMARK cuando se configura el parámetro ***set_tc_index***. El parámetro ***skb->iph->tos***, el cual contiene el valor del campo DS del paquete, es copiado dentro del campo ***skb->tc_index***, esto está representado por la línea roja del GRÁFICO 4.8

Para conseguir este comportamiento usamos el siguiente comando, mostrado en el GRÁFICO 4.9

```
# tc qdisc add dev eth0 handle 1:0 root dsmark indices 32 \
      set_tc_index
```

GRÁFICO 4.9 Comando para crear una disciplina de cola DSMARK con clasificador tcindex

Pero la utilidad de ***skb->tc_index*** no termina ahí, este campo es también leído por el clasificador especial ***tcindex***, entonces el filtro aplica una operación en una copia del valor, el valor original no es modificado, el valor final obtenido de esta operación es pasado a los elementos de filtro para que encuentren una coincidencia. Una vez que se tiene una coincidencia, el identificador de clase correspondiente a este filtro es regresado a la disciplina de cola como el identificador de clase resultante.

En el GRÁFICO 4.10, se la configuración para una disciplina de cola DSMARK, con clasificador **tcindex**, que maneja tres clases.

```
# tc qdisc add dev eth0 handle 1:0 root dsmark indices 16 \
      set_tc_index
# tc class change dev eth0 classid 1:1 dsmark mask 0x1f value 0x40
# tc class change dev eth0 classid 1:2 dsmark mask 0x1f value 0x40
# tc class change dev eth0 classid 1:3 dsmark mask 0x1f value 0x40
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 tcindex \
      mask 0xfc shift 2 pass_on
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 \
      handle 10 tcindex classid 1:1
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 \
      handle 12 tcindex classid 1:2
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 \
      handle 14 tcindex classid 1:3
```

GRÁFICO 4.10 Comandos de configuración del filtro principal tcindex

Primero se debe realizar la siguiente operación al valor del ***skb->tc_index*** para completar el lazo del clasificador:

*(**skb->tc_index & p.mask**)* >> ***p.shift***

Los valores de máscara y shift son valores enteros que se deben definir en el filtro principal, tal como se puede observar en la quinta línea de comando del GRÁFICO 4.10, en donde el valor de la **p.mask** es 0xfc y el de **p.shift** es 2.

En el GRÁFICO 4.10, la primera línea de comando, configura la disciplina de cola DSMARK raíz, el momento que se configura el comando ***set_tc_index*** el valor del campo DS del paquete es copiado al campo ***skb->tc_index*** cuando el paquete ingresa a la cola.

La segunda línea de comando, es el filtro principal. Para nuestro ejemplo tiene 3 elementos. En el filtro principal definimos los valores de **mask** = 0xfc y **shift** = 2. Este filtro lee el valor ***skb->tc_index***, el cual contiene el valor del campo DS del paquete, aplica las operaciones correspondientes, y el valor resultante es pasado a los filtros para que encuentren una coincidencia.

El comando ***pass_on,*** es utilizado para asegurarnos que si no se obtiene una coincidencia en el primer filtro pase a buscar a los siguientes filtros.

Para entender de mejor manera esto, supongamos que tenemos un paquete con un valor DS de 0x30, que corresponde a una clase AF12, esta entrando a la cola. El valor 0x30 es copiado por DSMARK en el campo ***skb->tc_index***.

A continuación el clasificador lee este campo, aplica las operaciones correspondientes:

(**skb->tc_index & p.mask**) >> **p.shift** = (0x30 & 0xfc) >> 2 =
(00110000 & 11111100) = 00110000 >> 2 = 00001100 = 0xc

Como podemos apreciar, la operación **shift**, lo único que hace es desplazar dos bits a la derecha, colocando ceros a la izquierda.

Para nuestro ejemplo el valor final después de estas operaciones es 0xc que corresponde al valor decimal 12. Este valor es pasado a los filtros para obtener una coincidencia. El primer filtro no coincide porque corresponde al valor decimal 10, entonces pasa al siguiente elemento en donde si coincide ya que corresponde al valor decimal 12, a continuación el identificador de clase que regresa a la disciplina de cola será 1:2. Este proceso esta representado en el GRÁFICO 4.8 por la línea de color azul.

Finalmente el identificador de clase va de regreso al campo ***skb->tc_index***, pero el valor que se copia en este campo es el menor del identificador de clase, o sea 2 de la ***classid 1:2***, este es ahora el nuevo valor contenido en el campo ***skb->tc_index***.

Este proceso esta representado en el GRÁFICO 4.8 por la línea vertical de color verde.

Finalmente la disciplina de cola DSMARK utiliza el valor del campo ***skb->tc_index*** como un índice para conseguir un par ***máscara – valor*** de la tabla interna configurada en las líneas de comandos 2,3 y 4 del GRÁFICO 4.9.

Con este par seleccionado se procede a modificar el campo DS del paquete aplicando una combinación de operaciones AND-OR, para conseguir el resultado deseado, de acuerdo a los requerimientos de la red.

Para entender de mejor forma todo lo explicado anteriormente, utilizaremos el siguiente ejemplo de configuración de DSMARK, para ello volveremos a utilizar el GRÁFICO 4.10.

```
# tc qdisc add dev eth0 handle 1:0 root dsmark indices 16 \
      set_tc_index
# tc class change dev eth0 classid 1:1 dsmark mask 0x1f value 0x40
# tc class change dev eth0 classid 1:2 dsmark mask 0x1f value 0x40
# tc class change dev eth0 classid 1:3 dsmark mask 0x1f value 0x40
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 tcindex \
      mask 0xfc shift 2 pass_on
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 \
      handle 10 tcindex classid 1:1
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 \
      handle 12 tcindex classid 1:2
# tc filter add dev eth0 parent 1:0 protocol ip prio 1 \
      handle 14 tcindex classid 1:3
```

GRÁFICO 4.10 Comandos de configuración de DSMARK

Este ejemplo no es muy práctico como debería serlo, pero bastará para explicar de forma sencilla el funcionamiento de esta disciplina de cola compleja.

La primera línea de comando, configura la disciplina de cola DSMARK raíz 1:0, las siguientes tres líneas de comandos, definen las clases que está disciplina de cola raíz manejará, cabe señalar que estos tres comandos crean la tabla interna con los pares **máscara – valor** que serán utilizados por la disciplina de cola DSMARK, para realizar el remarcado de los paquetes, mediante el uso del índice correspondiente.

A continuación, se configura el filtro principal, para que haga el remarcado de los paquetes que están con un DSCP AF1x a AF2x, es decir que los paquetes que ingresan marcados como AF1x, van a recibir el tratamiento necesario para que a la salida tengan su campo DS marcado con un valor AF2x.

En resumen se hace una copia del valor del campo DS del paquete entrante, a continuación se lo copia en el ***skb->tc_index***, de la disciplina de cola DSMARK, luego se invoca al filtro principal.

Supongamos que el paquete entrante, viene marcado con la clase AF12, después de copiar el valor del campo DS, el nuevo valor ***skb->tc_index*** será 0x30.

El filtro principal toma una copia de este valor (0x30) y aplica las operaciones correspondientes con los valores configurados de máscara y shift, según esto tendríamos las siguientes operaciones:

(0x30 & 0xfc) >> 2 = (00110000 & 11111100) >> 2 = 00110000 >> 2 = 00001100 = 0xc = 12

Como podemos observar, el resultado es el valor decimal 12, el cual se pasa a los elementos del filtro hasta encontrar una clase que corresponda con este valor, para el ejemplo descrito es el valor de identificador de clase 1:2, el cual regresa a la disciplina de cola, en donde DSMARK se encarga de separar este valor (1:2), para que se copie el menor valor de regreso en el campo ***skb->tc_index***, es decir el valor decimal 2.

La disciplina de cola DSMARK, lee el nuevo valor del campo ***skb->tc_index*** al momento de que el paquete esta dejando la cola, luego con este valor, que corresponde al índice (2), ingresa a su tabla interna para obtener un par **máscara – valor,** que le permita realizar las operaciones de AND y OR, para el ejemplo señalado el índice 2 contiene los siguientes valores de máscara y valor: 0x1f y 0x40 respectivamente:

(0x30 & 0x1f) | 0x40 = (00110000 & 00011111) | (01000000) =
(00010000 | 01000000) = 01010000 = 0x50

El resultado de estas operaciones es 0x50, que corresponde a un clase AF22, con lo que se cumple con el propósito de remarcado.

Finalmente debemos entender que lo más importante de todo este proceso es que la disciplina de cola DSMARK, lee el valor del campo ***skb->tc_index*** para encontrar un identificador de clase o índice que le permita obtener un par

máscara – valor de su tabla interna, para realizar el remarcado del campo DS del paquete que está siendo desencolado. Este proceso esta representado en el GRÁFICO 4.8 por las líneas de color púrpura.

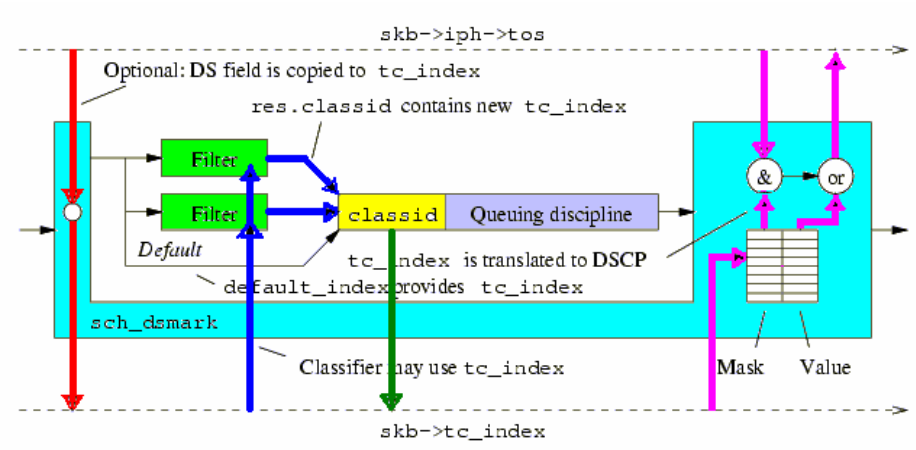


GRÁFICO 4.8 Disciplina de cola DSMARK

Para el caso más especifico como el de la voz sobre redes IP, hay que tener en cuenta que el marcado original de los paquetes se realizará en los equipos de VoIP, con valores preestablecidos para luego realizar el cálculo de los valores que serán utilizados dentro de la tabla interna de la disciplina de cola DSMARK, y de esta forma remarcar el campo DS de los paquetes con el valor correspondiente a cada flujo de tráfico según el requerimiento de los usuarios de la red.

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Se explicó como se puede obtener Calidad de Servicio con una optimización del ancho de banda, pues en la actualidad se obtiene la misma, realizando una segmentación del canal del cliente, lo cual conlleva a desperdiciar al ancho de banda disponible, pues este canal está disponible para VoIP aún cuando no se tengan paquetes que transmitir, mientras que con Diffserv el ancho de banda es usado en su totalidad, solo que se le da prioridad a los paquetes de voz o los que desee el usuario, pues depende del marcado que se le de cada paquete para darle un tratamiento específico.

- Hay que tener en cuenta que en gran medida, la Calidad de Servicio (QoS) que se pretende conseguir, dependerá de los acuerdos SLA a los que lleguemos con nuestro proveedor ISP, pues ellos son los que darán el tratamiento final al paquete, una vez que abandone nuestra red, y estos atraviesen otros ISPs o Internet, por lo que para obtener una calidad óptima de servicio, se requeriría que todos los puntos por lo cuales atraviesa nuestro paquete, hasta llegar a su destino, tengan los mismos niveles de acuerdo de servicio.

- Hay que tener en cuenta que la Calidad de Servicio provista por los Servicios Diferenciados (Diffserv) es unidireccional, es decir que el principal beneficiario de esto será el receptor de la conversación, puesto que solo está asegurado el paquete de ida y no de vuelta a menos que

el receptor use también Diffserv o algún otro mecanismo para asegurar Calidad de Servicio en la comunicación.

- Se pudo apreciar las facilidades que nos brinda el Sistema Operativo Linux al momento de marcar el campo DS de los paquetes, lo cual incide en un abaratamiento de costos, pues se puede convertir a un servidor normal en nuestro Ruteador, para el marcado de los paquetes, y de esta manera manejar la red según los requerimientos de los usuarios.

- Una de las grandes ventajas que nos brinda Diffserv, es el hecho de usarlo para servicios adicionales, no solo para voz, como puede ser el caso de Video bajo Demanda (VoD) ó juegos en red, en donde el Servidor que presta estos servicios es el encargado de asegurar la Calidad de Servicio para sus usuarios finales, para lo cual el sistema Diffserv, se adapta de forma ideal.

- En la Actualidad, el avance en la tecnología de VoIP, nos permite contar con una gama de equipos que facilitan al usuario el tratamiento de los paquetes, lo cual conlleva a un mejor tratamiento de los paquetes. Al momento de prestar el servicio de Diffserv a los distintos paquetes que emiten estos dispositivos.

- Un proveedor de Internet (ISP), puede integrar las conexiones pertenecientes a diferentes VPNs, dentro de un mismo agregado, recibiendo todas las conexiones el mismo tratamiento de Calidad de Servicio sin importar el tráfico que cursen por las mismas.

- Una de las dificultades que se puede tener al momento de implementar la Calidad de Servicio mediante el modelo de Servicios Diferenciados, es el problema de decidir quien es el encargado de marcar el campo DS de los paquetes, ya que el usuario podría llegar a utilizar mal los códigos

DSCP, asignando estos códigos a flujos de trafico que no requieren un tratamiento diferenciado, puesto que el cliente podría asignar el código de un paquete de voz a un paquete de datos, lo que podría ocasionar problemas en el nodo del ISP.

- Para que el modelo de Servicios Diferenciados funcione adecuadamente, se requiere un correcto dimensionamiento de la red, para obtener el máximo provecho de los recursos con los que cuenta la misma.

5.2 RECOMENDACIONES

- Se debe tener en cuenta, que para el uso de las herramientas y comandos provistos por iproute2, se debe contar con un núcleo del Sistema Operativo Linux, que sea mayor o igual al kernel 2.4, ya que a partir de esta distribución de Linux, se tiene ya incorporado todas estas herramientas que facilitan el uso de Diffserv en Linux.

- El Servidor, que realice el tratamiento y marcado de los paquetes correspondientes a los flujos de tráfico estipulados en el SLA, debe ser manejado adecuadamente para evitar casos en los que, se asigne prioridad a agregados de flujo que no correspondan ó que no se encuentren enmarcados en el SLA.

GLOSARIO

ADSL (Asymmetric Digital Subscriber Line): Línea de Abonado Digital Asimétrica, Consiste en una línea digital de alta velocidad, apoyada en el par simétrico de cobre que lleva la línea telefónica convencional o línea de abonado.

Behavior Aggregate (BA, también llamado a veces "agregado de tráfico"): es una colección de paquetes con el mismo DSCP (Diffserv Code Point) atravesando un enlace en una dirección.

CODEC: es una abreviatura de Codificador-Decodificador. Describe una especificación implementada en software, hardware o una combinación de ambos, capaz de transformar un archivo con un flujo de datos o una señal.

DNS Domain Name Service): Servicio de resolución de nombres en direcciones IP con el mismo fin que el protocolo RAS pero a través de un servidor DNS

Driver: programa encargado de servir de intermediario entre el sistema operativo y los distintos dispositivos conectados en el computador.

GSM (Global System for Mobile communications): Sistema Global para las Comunicaciones Móviles, es un estándar mundial de telefonía móvil digital que utiliza la codificación GMSK.

Inet: significa Internet (IPv4).

Kernel: núcleo del Sistema Operativo Linux, parte principal del sistema operativo. Código fuente del propio sistema.

MTU (Maximum Transfer Unit - MTU): expresa el tamaño en bytes del datagrama más grande que puede pasar por una capa de un protocolo de comunicaciones.

PBX: es el acrónimo de Private Branch eXchange o Private Business eXchange. También mal llamada por los usuarios como planta o central telefónica, es un servicio ofrecido por una empresa de telecomunicaciones.

Precedencia de descarte: en el interior de los nodos los paquetes son almacenados en buffers de tamaño finito, como consecuencia de esto cuando se agota su capacidad hay que proceder al descarte de uno o más paquetes. La precedencia de descarte permite determinar cuales son los paquetes que se van a descartar cuando se produzca está situación.

PSTN (Public Switching Telefonic Network): Red Pública Telefónica Conmutada, es una red de comunicación diseñada primordialmente para la transmisión de voz, aunque pueda también transportar datos, por ejemplo en el caso del fax o de la conexión a Internet a través de un módem.

qdisc: significa Disciplina de Cola (Queueing Discipline).

RAS (Registration, Admision and Status): Protocolo de comunicaciones que permite a una estación H.323 localizar otra estación H.323 a través del Gatekeeper

RDSI (Red Digital de Servicios Integrados o ISDN en inglés): Es una red que procede por evolución de la Red Digital Integrada (RDI) y que facilita conexiones digitales extremo a extremo para proporcionar una amplia gama de servicios, tanto de voz como de otros tipos, y a la que los usuarios acceden a través de un conjunto de interfaces normalizados.

RTP (Real-time Transport Protocol): Protocolo de Transporte de Tiempo real, es un protocolo de nivel de aplicación (no de nivel de transporte, como su nombre podría hacer pensar) utilizado para la transmisión de información en tiempo real, como por ejemplo audio y video en una video-conferencia.

UDP (User Datagram Protocol): es un protocolo del nivel de transporte basado en el intercambio de datagramas.

VoIP: Voz sobre Protocolo de Internet, también llamado Voz sobre IP, Telefonía IP, Telefonía por Internet, es el enrutamiento de conversaciones de voz sobre Internet o a través de alguna otra red basada en IP.

Zona H.323: es el conjunto de terminales, pasarelas y MCU (Unidad de Control Multipunto) gestionadas por un único gatekeeper.

BIBLIOGRAFÍA:

- José M. Huidrobo y David Roldan, “Integración de Voz y Datos”.
- http://www.tecnovida.com.ve/category/telecomunicaciones/ Voz sobre IP.htm
- http://telematica.cicese.mx/i2/Informes/InformeQcudi.doc
- http://www.redes-linux.com/manuales/routing/Enrutamiento-avanzado-y-control-de-trafico-en-Linux.pdf
- http://fmc.axarnet.es/tcp_ip/tema-01/tema-01.htm
- http://iie.fing.edu.uy/ense/asign/perfredes/trabajos/trabajos_2003/Diffserv/Trabajo%20Final.pdf
- http://www.microsoft.com/latam/technet/articulos/windows2k/QoSmech/
- http://telematica.cicese.mx/i2/presentaciones/Primavera_2k1_CUDI_parte_2_files/frame.htm
- http://www.opalsoft.net/QoS/DS.htm
- http://www.ehas.org/cgi-bin/viewcvs.cgi/VoIP/QoS.xml?rev=1.5
- http://metroareanetworks.com/services/documentation/ExtremeWare6_1_spa.pdf
- http://redes-Sistema Operativo Linux.all-inone.net/manuales/routing/Enrutamiento-avanzado-y-control-de-trafico-en-Sistema Operativo Linux.pdf
- http://www.telefonica.es/sociedaddelainformacion/pdf/publicaciones/telecomunicacionesng/teleco_n_g.pdf
- http://usuarios.lycos.es/marsen/cap2-protocolos.htm
- www.ciiscg.org/Pdfs/articulo_telconet_metro_ethernet_ver2.pdf
- http://www.monografias.com/trabajos3/voip/voip.shtml
- http://www.monografias.com/trabajos10/tele/tele.shtml
- http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=219&mode=thread&order=0&thold=0
- http://www.voip-es.com/index.php?option=com_content&task=view&id=3&Itemid=61

- http://www.interec.com/productos/voIP/
- http://es.wikipedia.org/wiki/Ataque_de_denegaci%C3%B3n_de_servicio
- http://www.maestrosdelweb.com/editorial/ddos/
- http://www.eslared.org.ve/articulos/ermanno/voip.pdf
- http://es.wikipedia.org/wiki/Kernel
- http://www.monografias.com/trabajos18/teoria-colas/teoria-colas.shtml
- http://www.it.uc3m.es/~prometeo/rsc/problemas/Examenes_uc3m/feb98/teoria~1/teoria~1.html#SOLp3
- http://es.wikipedia.org/wiki/Address_Resolution_Protocol

ANEXOS

ANEXO A: RFC 2474 Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers

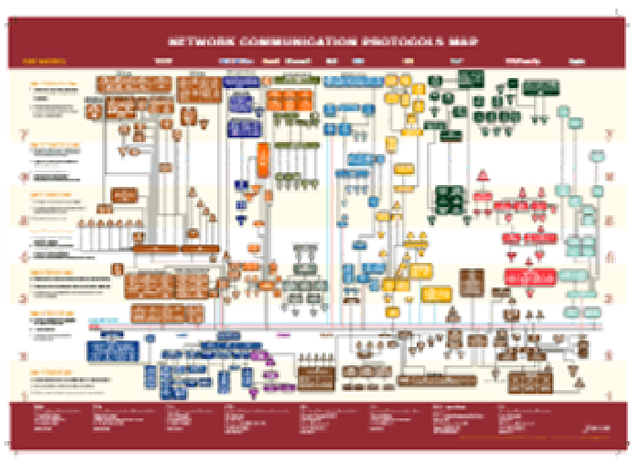
Network Working Group
Request for Comments: 2474
Obsoletes: 1455, 1349
Category: Standards Track

K. Nichols
Cisco Systems
S. Blake
Torrent Networking Technologies
F. Baker
Cisco Systems
D. Black
EMC Corporation
December 1998

Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.



Network Communication Protocol Map. To order: http://www.javvin.com/map.html
Easy to use network sniffing tool: http://www.javvin.com/packet.html

Abstract

Differentiated services enhancements to the Internet protocol are intended to enable scalable service discrimination in the Internet without the need for per-flow state and signaling at every hop. A variety of services may be built from a small, well-defined set of building blocks which are deployed in network nodes. The services may be either end-to-end or intra-domain; they include both those that can satisfy quantitative performance requirements (e.g., peak bandwidth) and those based on relative performance (e.g., "class" differentiation). Services can be constructed by a combination of:

- setting bits in an IP header field at network boundaries (autonomous system boundaries, internal administrative boundaries, or hosts),
- using those bits to determine how packets are forwarded by the nodes inside the network, and
- conditioning the marked packets at network boundaries in accordance with the requirements or rules of each service.

The requirements or rules of each service must be set through administrative policy mechanisms which are outside the scope of this document. A differentiated services-compliant network node includes a classifier that selects packets based on the value of the DS field, along with buffer management and packet scheduling mechanisms capable of delivering the specific packet forwarding treatment indicated by the DS field value. Setting of the DS field and conditioning of the temporal behavior of marked packets need only be performed at network boundaries and may vary in complexity.

This document defines the IP header field, called the DS (for differentiated services) field. In IPv4, it defines the layout of the TOS octet; in IPv6, the Traffic Class octet. In addition, a base set of packet forwarding treatments, or per-hop behaviors, is defined.

For a more complete understanding of differentiated services, see also the differentiated services architecture [ARCH].

Table of Contents

1. Introduction

Differentiated services are intended to provide a framework and building blocks to enable deployment of scalable service discrimination in the Internet. The differentiated services approach aims to speed deployment by separating the architecture into two major components, one of which is fairly well-understood and the other of which is just beginning to be understood. In this, we are guided by the original design of the Internet where the decision was made to separate the forwarding and routing components. Packet forwarding is the relatively simple task that needs to be performed on a per-packet basis as quickly as possible. Forwarding uses the packet header to find an entry in a routing table that determines the packet's output interface. Routing sets the entries in that table and may need to reflect a range of transit and other policies as well as to keep track of route failures. Routing tables are maintained as a background process to the forwarding task. Further, routing is the more complex task and it has continued to evolve over the past 20 years.

Analogously, the differentiated services architecture contains two main components. One is the fairly well-understood behavior in the forwarding path and the other is the more complex and still emerging background policy and allocation component that configures parameters used in the forwarding path. The forwarding path behaviors include the differential treatment an individual packet receives, as implemented by queue service disciplines and/or queue management disciplines. These per-hop behaviors are useful and required in network nodes to deliver differentiated treatment of packets no matter how we construct end-to-end or intra-domain services. Our focus is on the general semantics of the behaviors rather than the specific mechanisms used to implement them since these behaviors will evolve less rapidly than the mechanisms.

Per-hop behaviors and mechanisms to select them on a per-packet basis can be deployed in network nodes today and it is this aspect of the differentiated services architecture that is being addressed first. In addition, the forwarding path may require that some monitoring, policing, and shaping be done on the network traffic designated for "special" treatment in order to enforce requirements associated with the delivery of the special treatment. Mechanisms for this kind of traffic conditioning are also fairly well-understood. The wide deployment of such traffic conditioners is also important to enable the construction of services, though their actual use in constructing services may evolve over time.

The configuration of network elements with respect to which packets get special treatment and what kinds of rules are to be applied to the use of resources is much less well-understood. Nevertheless, it is possible to deploy useful differentiated services in networks by using simple policies and static configurations. As described in [ARCH], there are a number of ways to compose per-hop behaviors and traffic conditioners to create services. In the process, additional experience is gained that will guide more complex policies and allocations. The basic behaviors in the forwarding path can remain the same while this component of the architecture evolves. Experiences with the construction of such services will continue for some time, thus we avoid standardizing this construction as it is premature. Further, much of the details of service construction are covered by legal agreements between different business entities and we avoid this as it is very much outside the scope of the IETF.

This document concentrates on the forwarding path component. In the packet forwarding path, differentiated services are realized by mapping the codepoint contained in a field in the IP packet header to a particular forwarding treatment, or per-hop behavior (PHB), at each network node along its path. The codepoints may be chosen from a set of mandatory values defined later in this document, from a set of recommended values to be defined in future documents, or may have purely local meaning. PHBs are expected to be implemented by employing a range of queue service and/or queue management disciplines on a network node's output interface queue: for example weighted round-robin (WRR) queue servicing or drop-preference queue management.

Marking is performed by traffic conditioners at network boundaries, including the edges of the network (first-hop router or source host) and administrative boundaries. Traffic conditioners may include the primitives of marking, metering, policing and shaping (these mechanisms are described in [ARCH]). Services are realized by the use of particular packet classification and traffic conditioning mechanisms at boundaries coupled with the concatenation of per-hop behaviors along the transit path of the traffic. A goal of the differentiated services architecture is to specify these building blocks for future extensibility, both of the number and type of the building blocks and of the services built from them.

Terminology used in this memo is defined in Sec. 2. The differentiated services field definition (DS field) is given in Sec. 3. In Sec. 4, we discuss the desire for partial backwards compatibility with current use of the IPv4 Precedence field. As a solution, we introduce Class Selector Codepoints and Class Selector

Compliant PHBs. Sec. 5 presents guidelines for per-hop behavior standardization. Sec. 6 discusses guidelines for allocation of codepoints. Sec. 7 covers security considerations.

This document is a concise description of the DS field and its uses. It is intended to be read along with the differentiated services architecture [ARCH].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Terminology Used in This Document

Behavior Aggregate: a collection of packets with the same codepoint crossing a link in a particular direction. The terms "aggregate" and "behavior aggregate" are used interchangeably in this document.

Classifier: an entity which selects packets based on the content of packet headers according to defined rules.

Class Selector Codepoint: any of the eight codepoints in the range ' xxx000' (where 'x' may equal 'O' or '1'). Class Selector Codepoints are discussed in Sec. 4.2.2.

Class Selector Compliant PHB: a per-hop behavior satisfying the Class Selector PHB Requirements specified in Sec. 4.2.2.2.

Codepoint: a specific value of the DSCP portion of the DS field. Recommended codepoints SHOULD map to specific, standardized PHBs. Multiple codepoints MAY map to the same PHB.

Differentiated Services Boundary: the edge of a DS domain, where classifiers and traffic conditioners are likely to be deployed. A differentiated services boundary can be further sub-divided into ingress and egress nodes, where the ingress/egress nodes are the downstream/upstream nodes of a boundary link in a given traffic direction. A differentiated services boundary typically is found at the ingress to the first-hop differentiated services-compliant router (or network node) that a host's packets traverse, or at the egress of the last-hop differentiated services-compliant router or network node that packets traverse before arriving at a host. This is sometimes referred to as the boundary at a leaf router. A differentiated services boundary may be co-located with a host, subject to local policy. Also DS boundary.

Differentiated Services-Compliant: in compliance with the requirements specified in this document. Also DS-compliant.

Differentiated Services Domain: a contiguous portion of the Internet over which a consistent set of differentiated services policies are administered in a coordinated fashion. A differentiated services domain can represent different administrative domains or autonomous systems, different trust regions, different network technologies (e.g., cell/frame), hosts and routers, etc. Also DS domain.

Differentiated Services Field: the IPv4 header TOS octet or the IPv6 Traffic Class octet when interpreted in conformance with the definition given in this document. Also DS field.

Mechanism: The implementation of one or more per-hop behaviors according to a particular algorithm.

Microflow: a single instance of an application-to-application flow of packets which is identified by source address, destination address, protocol id, and source port, destination port (where applicable).

Per-hop Behavior (PHB): a description of the externally observable forwarding treatment applied at a differentiated services-compliant node to a behavior aggregate. The description of a PHB SHOULD be sufficiently detailed to allow the construction of predictable services, as documented in [ARCH].

Per-hop Behavior Group: a set of one or more PHBs that can only be meaningfully specified and implemented simultaneously, due to a common constraint applying to all PHBs in the set such as a queue servicing or queue management policy. Also PHB Group.

Traffic Conditioning: control functions that can be applied to a behavior aggregate, application flow, or other operationally useful subset of traffic, e.g., routing updates. These MAY include metering, policing, shaping, and packet marking. Traffic conditioning is used to enforce agreements between domains and to condition traffic to receive a differentiated service within a domain by marking packets with the appropriate codepoint in the DS field and by monitoring and altering the temporal characteristics of the aggregate where necessary. See [ARCH].

Traffic Conditioner: an entity that performs traffic conditioning functions and which MAY contain meters, policers, shapers, and markers. Traffic conditioners are typically deployed in DS boundary nodes (i.e., not in interior nodes of a DS domain).

Service: a description of the overall treatment of (a subset of) a customer's traffic across a particular domain, across a set of interconnected DS domains, or end-to-end. Service descriptions are covered by administrative policy and services are constructed by

applying traffic conditioning to create behavior aggregates which experience a known PHB at each node within the DS domain. Multiple services can be supported by a single per-hop behavior used in concert with a range of traffic conditioners.

To summarize, classifiers and traffic conditioners are used to select which packets are to be added to behavior aggregates. Aggregates receive differentiated treatment in a DS domain and traffic conditioners MAY alter the temporal characteristics of the aggregate to conform to some requirements. A packet's DS field is used to designate the packet's behavior aggregate and is subsequently used to determine which forwarding treatment the packet receives. A behavior aggregate classifier which can select a PHB, for example a differential output queue servicing discipline, based on the codepoint in the DS field SHOULD be included in all network nodes in a DS domain. The classifiers and traffic conditioners at DS boundaries are configured in accordance with some service specification, a matter of administrative policy outside the scope of this document.

Additional differentiated services definitions are given in [ARCH].

3. Differentiated Services Field Definition

A replacement header field, called the DS field, is defined, which is intended to supersede the existing definitions of the IPv4 TOS octet [RFC791] and the IPv6 Traffic Class octet [IPv6].

Six bits of the DS field are used as a codepoint (DSCP) to select the PHB a packet experiences at each node. A two-bit currently unused (CU) field is reserved and its definition and interpretation are outside the scope of this document. The value of the CU bits are ignored by differentiated services-compliant nodes when determining the per-hop behavior to apply to a received packet.

The DS field structure is presented below:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|          DSCP           |  CU   |
+---+---+---+---+---+---+---+---+
```

DSCP: differentiated services codepoint
CU: currently unused

In a DSCP value notation 'xxxxxx' (where 'x' may equal 'O' or '1') used in this document, the left-most bit signifies bit 0 of the DS field (as shown above), and the right-most bit signifies bit 5.

Implementors should note that the DSCP field is six bits wide. DS-compliant nodes MUST select PHBs by matching against the entire 6-bit DSCP field, e.g., by treating the value of the field as a table index which is used to select a particular packet handling mechanism which has been implemented in that device. The value of the CU field MUST be ignored by PHB selection. The DSCP field is defined as an unstructured field to facilitate the definition of future per-hop behaviors.

With some exceptions noted below, the mapping of codepoints to PHBs MUST be configurable. A DS-compliant node MUST support the logical equivalent of a configurable mapping table from codepoints to PHBs. PHB specifications MUST include a recommended default codepoint, which MUST be unique for codepoints in the standard space (see Sec. 6). Implementations should support the recommended codepoint-to-PHB mappings in their default configuration. Operators may choose to use different codepoints for a PHB, either in addition to or in place of the recommended default. Note that if operators do so choose, re-marking of DS fields may be necessary at administrative boundaries even if the same PHBs are implemented on both sides of the boundary.

See [ARCH] for further discussion of re-marking.

The exceptions to general configurability are for codepoints 'xxx000' and are noted in Secs. 4.2.2 and 4.3.

Packets received with an unrecognized codepoint SHOULD be forwarded as if they were marked for the Default behavior (see Sec. 4), and their codepoints should not be changed. Such packets MUST NOT cause the network node to malfunction.

The structure of the DS field shown above is incompatible with the existing definition of the IPv4 TOS octet in [RFC791]. The presumption is that DS domains protect themselves by deploying re-marking boundary nodes, as should networks using the RFC 791 Precedence designations. Correct operational procedure SHOULD follow [RFC791], which states: "If the actual use of these precedence designations is of concern to a particular network, it is the responsibility of that network to control the access to, and use of, those precedence designations." Validating the value of the DS field at DS boundaries is sensible in any case since an upstream node can easily set it to any arbitrary value. DS domains that are not isolated by suitably configured boundary nodes may deliver unpredictable service.

Nodes MAY rewrite the DS field as needed to provide a desired local or end-to-end service. Specifications of DS field translations at DS boundaries are the subject of service level agreements between providers and users, and are outside the scope of this document. Standardized PHBs allow providers to build their services from a well-known set of packet forwarding treatments that can be expected to be present in the equipment of many vendors.

4. Historical Codepoint Definitions and PHB Requirements

The DS field will have a limited backwards compatibility with current practice, as described in this section. Backwards compatibility is addressed in two ways. First, there are per-hop behaviors that are already in widespread use (e.g., those satisfying the IPv4 Precedence queueing requirements specified in [RFC1812]), and we wish to permit their continued use in DS-compliant nodes. In addition, there are some codepoints that correspond to historical use of the IP Precedence field and we reserve these codepoints to map to PHBs that meet the general requirements specified in Sec. 4.2.2.2, though the specific differentiated services PHBs mapped to by those codepoints MAY have additional specifications.

No attempt is made to maintain backwards compatibility with the "DTR" or TOS bits of the IPv4 TOS octet, as defined in [RFC791].

4.1 A Default PHB

A "default" PHB MUST be available in a DS-compliant node. This is the common, best-effort forwarding behavior available in existing routers as standardized in [RFC1812]. When no other agreements are in place, it is assumed that packets belong to this aggregate. Such packets MAY be sent into a network without adhering to any particular rules and the network will deliver as many of these packets as possible and as soon as possible, subject to other resource policy constraints. A reasonable implementation of this PHB would be a queueing discipline that sends packets of this aggregate whenever the output link is not required to satisfy another PHB. A reasonable policy for constructing services would ensure that the aggregate was not "starved". This could be enforced by a mechanism in each node that reserves some minimal resources (e.g, buffers, bandwidth) for Default behavior aggregates. This permits senders that are not differentiated services-aware to continue to use the network in the same manner as today. The impact of the introduction of differentiated services into a domain on the service expectations of its customers and peers is a complex matter involving policy decisions by the domain and is outside the scope of this document. The RECOMMENDED codepoint for the Default PHB is the bit pattern '000000'; the value '000000' MUST map to a PHB that meets these

specifications. The codepoint chosen for Default behavior is compatible with existing practice [RFC791]. Where a codepoint is not mapped to a standardized or local use PHB, it SHOULD be mapped to the Default PHB.

A packet initially marked for the Default behavior MAY be re-marked with another codepoint as it passes a boundary into a DS domain so that it will be forwarded using a different PHB within that domain, possibly subject to some negotiated agreement between the peering domains.

4.2 Once and Future IP Precedence Field Use

We wish to maintain some form of backward compatibility with present uses of the IP Precedence Field: bits 0-2 of the IPv4 TOS octet. Routers exist that use the IP Precedence field to select different per-hop forwarding treatments in a similar way to the use proposed here for the DSCP field. Thus, a simple prototype differentiated services architecture can be quickly deployed by appropriately configuring these routers. Further, IP systems today understand the location of the IP Precedence field, and thus if these bits are used in a similar manner as DS-compliant equipment is deployed, significant failures are not likely during early deployment. In other words, strict DS-compliance need not be ubiquitous even within a single service provider's network if bits 0-2 of the DSCP field are employed in a manner similar to, or subsuming, the deployed uses of the IP Precedence field.

4.2.1 IP Precedence History and Evolution in Brief

The IP Precedence field is something of a forerunner of the DS field. IP Precedence, and the IP Precedence Field, were first defined in [RFC791]. The values that the three-bit IP Precedence Field might take were assigned to various uses, including network control traffic, routing traffic, and various levels of privilege. The least level of privilege was deemed "routine traffic". In [RFC791], the notion of Precedence was defined broadly as "An independent measure of the importance of this datagram." Not all values of the IP Precedence field were assumed to have meaning across boundaries, for instance "The Network Control precedence designation is intended to be used within a network only. The actual use and control of that designation is up to each network." [RFC791]

Although early BBN IMPs implemented the Precedence feature, early commercial routers and UNIX IP forwarding code generally did not. As networks became more complex and customer requirements grew, commercial router vendors developed ways to implement various kinds of queueing services including priority queueing, which were

generally based on policies encoded in filters in the routers, which examined IP addresses, IP protocol numbers, TCP or UDP ports, and other header fields. IP Precedence was and is among the options such filters can examine.

In short, IP Precedence is widely deployed and widely used, if not in exactly the manner intended in [RFC791]. This was recognized in [RFC1122], which states that while the use of the IP Precedence field is valid, the specific assignment of the priorities in [RFC791] were merely historical.

4.2.2 Subsuming IP Precedence into Class Selector Codepoints

A specification of the packet forwarding treatments selected by the IP Precedence field today would have to be quite general; probably not specific enough to build predictable services from in the differentiated services framework. To preserve partial backwards compatibility with known current uses of the IP Precedence field without sacrificing future flexibility, we have taken the approach of describing minimum requirements on a set of PHBs that are compatible with most of the deployed forwarding treatments selected by the IP Precedence field. In addition, we give a set of codepoints that MUST map to PHBs meeting these minimum requirements. The PHBs mapped to by these codepoints MAY have a more detailed list of specifications in addition to the required ones stated here. Other codepoints MAY map to these same PHBs. We refer to this set of codepoints as the Class Selector Codepoints, and the minimum requirements for PHBs that these codepoints may map to are called the Class Selector PHB Requirements.

4.2.2.1 The Class Selector Codepoints

A specification of the packet forwarding treatments selected by the The DS field values of 'xxx000|xx', or DSCP = 'xxx000' and CU subfield unspecified, are reserved as a set of Class Selector Codepoints. PHBs which are mapped to by these codepoints MUST satisfy the Class Selector PHB requirements in addition to preserving the Default PHB requirement on codepoint '000000' (Sec. 4.1).

4.2.2.2 The Class Selector PHB Requirements

We refer to a Class Selector Codepoint with a larger numerical value than another Class Selector Codepoint as having a higher relative order while a Class Selector Codepoint with a smaller numerical value than another Class Selector Codepoint is said to have a lower relative order. The set of PHBs mapped to by the eight Class Selector Codepoints MUST yield at least two independently forwarded classes of traffic, and PHBs selected by a Class Selector Codepoint

SHOULD give packets a probability of timely forwarding that is not lower than that given to packets marked with a Class Selector codepoint of lower relative order, under reasonable operating conditions and traffic loads. A discarded packet is considered to be an extreme case of untimely forwarding. In addition, PHBs selected by codepoints '11x000' MUST give packets a preferential forwarding treatment by comparison to the PHB selected by codepoint '000000' to preserve the common usage of IP Precedence values '110' and '111' for routing traffic.

Further, PHBs selected by distinct Class Selector Codepoints SHOULD be independently forwarded; that is, packets marked with different Class Selector Codepoints MAY be re-ordered. A network node MAY enforce limits on the amount of the node's resources that can be utilized by each of these PHBs.

PHB groups whose specification satisfy these requirements are referred to as Class Selector Compliant PHBs.

The Class Selector PHB Requirements on codepoint '000000' are compatible with those listed for the Default PHB in Sec. 4.1.

4.2.2.3 Using the Class Selector PHB Requirements for IP Precedence Compatibility

A DS-compliant network node can be deployed with a set of one or more Class Selector Compliant PHB groups. This document states that the set of codepoints 'xxx000' MUST map to such a set of PHBs. As it is also possible to map multiple codepoints to the same PHB, the vendor or the network administrator MAY configure the network node to map codepoints to PHBs irrespective of bits 3-5 of the DSCP field to yield a network that is compatible with historical IP Precedence use. Thus, for example, codepoint '011010' would map to the same PHB as codepoint '011000'.

4.2.2.4 Example Mechanisms for Implementing Class Selector Compliant PHB Groups

Class Selector Compliant PHBs can be realized by a variety of mechanisms, including strict priority queueing, weighted fair queueing (WFQ), WRR, or variants [RPS, HPFQA, DRR], or Class-Based Queuing [CBQ]. The distinction between PHBs and mechanisms is described in more detail in Sec. 5.

It is important to note that these mechanisms might be available through other PHBs (standardized or not) that are available in a particular vendor's equipment. For example, future documents may standardize a Strict Priority Queueing PHB group for a set of

recommended codepoints. A network administrator might configure those routers to select the Strict Priority Queueing PHBs with codepoints 'xxx000' in conformance with the requirements of this document.

As a further example, another vendor might employ a CBQ mechanism in its routers. The CBQ mechanism could be used to implement the Strict Priority Queueing PHBs as well as a set of Class Selector Compliant PHBs with a wider range of features than would be available in a set of PHBs that did no more than meet the minimum Class Selector PHB requirements.

4.3 Summary

This document defines codepoints 'xxx000' as the Class Selector codepoints, where PHBs selected by these codepoints MUST meet the Class Selector PHB Requirements described in Sec. 4.2.2.2. This is done to preserve a useful level of backward compatibility with current uses of the IP Precedence field in the Internet without unduly limiting future flexibility. In addition, codepoint '000000' is used as the Default PHB value for the Internet and, as such, is not configurable. The remaining seven non-zero Class Selector codepoints are configurable only to the extent that they map to PHBs that meet the requirements in Sec. 4.2.2.2.

5. Per-Hop Behavior Standardization Guidelines

The behavioral characteristics of a PHB are to be standardized, and not the particular algorithms or the mechanisms used to implement them. A node may have a (possibly large) set of parameters that can be used to control how packets are scheduled onto an output interface (e.g., N separate queues with settable priorities, queue lengths, round-robin weights, drop algorithm, drop preference weights and thresholds, etc). To illustrate the distinction between a PHB and a mechanism, we point out that Class Selector Compliant PHBs might be implemented by several mechanisms, including: strict priority queueing, WFQ, WRR, or variants [HPFQA, RPS, DRR], or CBQ [CBQ], in isolation or in combination.

PHBs may be specified individually, or as a group (a single PHB is a special case of a PHB group). A PHB group usually consists of a set of two or more PHBs that can only be meaningfully specified and implemented simultaneously, due to a common constraint applying to each PHB within the group, such as a queue servicing or queue management policy. A PHB group specification SHOULD describe conditions under which a packet might be re-marked to select another PHB within the group. It is RECOMMENDED that PHB implementations do not introduce any packet re-ordering within a microflow. PHB group

specifications MUST identify any possible packet re-ordering implications which may occur for each individual PHB, and which may occur if different packets within a microflow are marked for different PHBs within the group.

Only those per-hop behaviors that are not described by an existing PHB standard, and have been implemented, deployed, and shown to be useful, SHOULD be standardized. Since current experience with differentiated services is quite limited, it is premature to hypothesize the exact specification of these per-hop behaviors.

Each standardized PHB MUST have an associated RECOMMENDED codepoint, allocated out of a space of 32 codepoints (see Sec. 6). This specification has left room in the codepoint space to allow for evolution, thus the defined space ('xxx000') is intentionally sparse.

Network equipment vendors are free to offer whatever parameters and capabilities are deemed useful or marketable. When a particular, standardized PHB is implemented in a node, a vendor MAY use any algorithm that satisfies the definition of the PHB according to the standard. The node's capabilities and its particular configuration determine the different ways that packets can be treated.

Service providers are not required to use the same node mechanisms or configurations to enable service differentiation within their networks, and are free to configure the node parameters in whatever way that is appropriate for their service offerings and traffic engineering objectives. Over time certain common per-hop behaviors are likely to evolve (i.e., ones that are particularly useful for implementing end-to-end services) and these MAY be associated with particular EXP/LU PHB codepoints in the DS field, allowing use across domain boundaries (see Sec. 6). These PHBs are candidates for future standardization.

It is RECOMMENDED that standardized PHBs be specified in accordance with the guidelines set out in [ARCH].

6. IANA Considerations

The DSCP field within the DS field is capable of conveying 64 distinct codepoints. The codepoint space is divided into three pools for the purpose of codepoint assignment and management: a pool of 32 RECOMMENDED codepoints (Pool 1) to be assigned by Standards Action as defined in [CONS], a pool of 16 codepoints (Pool 2) to be reserved for experimental or Local Use (EXP/LU) as defined in [CONS], and a pool of 16 codepoints (Pool 3) which are initially available for experimental or local use, but which should be preferentially

utilized for standardized assignments if Pool 1 is ever exhausted. The pools are defined in the following table (where 'x' refers to either '0' or '1'):

Pool	Codepoint space	Assignment Policy
1	xxxxx0	Standards Action
2	xxxx11	EXP/LU
3	xxxx01	EXP/LU (*)

(*) may be utilized for future Standards Action allocations as necessary

This document assigns eight RECOMMENDED codepoints ('xxx000') which are drawn from Pool 1 above. These codepoints MUST be mapped, not to specific PHBs, but to PHBs that meet "at least" the requirements set forth in Sec. 4.2.2.2 to provide a minimal level of backwards compatibility with IP Precedence as defined in [RFC791] and as deployed in some current equipment.

7. Security Considerations

This section considers security issues raised by the introduction of differentiated services, primarily the potential for denial-of-service attacks, and the related potential for theft of service by unauthorized traffic (Section 7.1). Section 7.2 addresses the operation of differentiated services in the presence of IPsec including its interaction with IPsec tunnel mode and other tunnelling protocols. See [ARCH] for more extensive treatment of the security concerns raised by the overall differentiated services architecture.

7.1 Theft and Denial of Service

The primary goal of differentiated services is to allow different levels of service to be provided for traffic streams on a common network infrastructure. A variety of techniques may be used to achieve this, but the end result will be that some packets receive different (e.g., better) service than others. The mapping of network traffic to the specific behaviors that result in different (e.g., better or worse) service is indicated primarily by the DS codepoint, and hence an adversary may be able to obtain better service by modifying the codepoint to values indicating behaviors used for enhanced services or by injecting packets with such codepoint values. Taken to its limits, such theft-of-service becomes a denial-of-service attack when the modified or injected traffic depletes the resources available to forward it and other traffic streams.

The defense against this class of theft- and denial-of-service attacks consists of the combination of traffic conditioning at DS domain boundaries with security and integrity of the network infrastructure within a DS domain. DS domain boundary nodes MUST ensure that all traffic entering the domain is marked with codepoint values appropriate to the traffic and the domain, remarking the traffic with new codepoint values if necessary. These DS boundary nodes are the primary line of defense against theft- and denial-of-service attacks based on modified codepoints, as success of any such attack indicates that the codepoints used by the attacking traffic were inappropriate. An important instance of a boundary node is that any traffic-originating node within a DS domain is the initial boundary node for that traffic. Interior nodes in a DS domain rely on DS codepoints to associate traffic with the forwarding PHBs, and are NOT REQUIRED to check codepoint values before using them. As a result, the interior nodes depend on the correct operation of the DS domain boundary nodes to prevent the arrival of traffic with inappropriate codepoints or in excess of provisioned levels that would disrupt operation of the domain.

7.2 IPsec and Tunnelling Interactions

The IPsec protocol, as defined in [ESP, AH], does not include the IP header's DS field in any of its cryptographic calculations (in the case of tunnel mode, it is the outer IP header's DS field that is not included). Hence modification of the DS field by a network node has no effect on IPsec's end-to-end security, because it cannot cause any IPsec integrity check to fail. As a consequence, IPsec does not provide any defense against an adversary's modification of the DS field (i.e., a man-in-the-middle attack), as the adversary's modification will also have no effect on IPsec's end-to-end security.

IPsec's tunnel mode provides security for the encapsulated IP header's DS field. A tunnel mode IPsec packet contains two IP headers: an outer header supplied by the tunnel ingress node and an encapsulated inner header supplied by the original source of the packet. When an IPsec tunnel is hosted (in whole or in part) on a differentiated services network, the intermediate network nodes operate on the DS field in the outer header. At the tunnel egress node, IPsec processing includes removing the outer header and forwarding the packet (if required) using the inner header. The IPsec protocol REQUIRES that the inner header's DS field not be changed by this decapsulation processing to ensure that modifications to the DS field cannot be used to launch theft- or denial-of-service attacks across an IPsec tunnel endpoint. This document makes no change to that requirement. If the inner IP header has not been processed by a DS boundary node for the tunnel egress node's DS

domain, the tunnel egress node is the boundary node for traffic exiting the tunnel, and hence MUST ensure that the resulting traffic has appropriate DS codepoints.

When IPsec tunnel egress decapsulation processing includes a sufficiently strong cryptographic integrity check of the encapsulated packet (where sufficiency is determined by local security policy), the tunnel egress node can safely assume that the DS field in the inner header has the same value as it had at the tunnel ingress node. An important consequence is that otherwise insecure links within a DS domain can be secured by a sufficiently strong IPsec tunnel. This analysis and its implications apply to any tunnelling protocol that performs integrity checks, but the level of assurance of the inner header's DS field depends on the strength of the integrity check performed by the tunnelling protocol. In the absence of sufficient assurance for a tunnel that may transit nodes outside the current DS domain (or is otherwise vulnerable), the encapsulated packet MUST be treated as if it had arrived at a boundary from outside the DS domain.

8. Acknowledgements

The authors would like to acknowledge the Differentiated Services Working Group for discussions which helped shape this document.

9. References

[AH] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[ARCH] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[CBQ] S. Floyd and V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks", IEEE/ACM Transactions on Networking, Vol. 3 no. 4, pp. 365-386, August 1995.

[CONS] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434, October 1998.

[DRR] M. Shreedhar and G. Varghese, Efficient Fair Queueing using Deficit Round Robin", Proc. ACM SIGCOMM 95, 1995.

[ESP] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[HPFQA] J. Bennett and Hui Zhang, "Hierarchical Packet Fair Queueing Algorithms", Proc. ACM SIGCOMM 96, August 1996.

[IPv6] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC791] Postel, J., Editor, "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC1122] Braden, R., "Requirements for Internet hosts - communication layers", STD 3, RFC 1122, October 1989.

[RFC1812] Baker, F., Editor, "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RPS] D. Stiliadis and A. Varma, "Rate-Proportional Servers: A Design Methodology for Fair Queueing Algorithms", IEEE/ACM Trans. on Networking, April 1998.

Authors' Addresses

Kathleen Nichols
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134-1706

Phone: +1-408-525-4857
EMail: kmn@cisco.com

Steven Blake
Torrent Networking Technologies
3000 Aerial Center, Suite 140
Morrisville, NC 27560

Phone: +1-919-468-8466 x232
EMail: slblake@torrentnet.com

Fred Baker
Cisco Systems
519 Lado Drive
Santa Barbara, CA 93111

Phone: +1-408-526-4257
EMail: fred@cisco.com

David L. Black
EMC Corporation
35 Parkwood Drive
Hopkinton, MA 01748

Phone: +1-508-435-1000 x76140
EMail: black_david@emc.com

Full Copyright Statement

ANEXO B: RFC 2475 An Architecture for Differentiated Services

Network Working Group
Request for Comments: 2475
Category: Informational

S. Blake
Torrent Networking Technologies
D. Black
EMC Corporation
M. Carlson
Sun Microsystems
E. Davies
Nortel UK
Z. Wang
Bell Labs Lucent Technologies
W. Weiss
Lucent Technologies

December 1998

An Architecture for Differentiated Services

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Network Communication Protocol Map. To order: http://www.javvin.com/map.html
Easy to use network sniffing tool: http://www.javvin.com/packet.html

Abstract

This document defines an architecture for implementing scalable service differentiation in the Internet. This architecture achieves scalability by aggregating traffic classification state which is conveyed by means of IP-layer packet marking using the DS field [DSFIELD]. Packets are classified and marked to receive a particular per-hop forwarding behavior on nodes along their path. Sophisticated classification, marking, policing, and shaping operations need only be implemented at network boundaries or hosts. Network resources are allocated to traffic streams by service provisioning policies which govern how traffic is marked and conditioned upon entry to a differentiated services-capable network, and how that traffic is forwarded within that network. A wide variety of services can be implemented on top of these building blocks.

Table of Contents

1. Introduction

1.1 Overview

This document defines an architecture for implementing scalable service differentiation in the Internet. A "Service" defines some significant characteristics of packet transmission in one direction across a set of one or more paths within a network. These

characteristics may be specified in quantitative or statistical terms of throughput, delay, jitter, and/or loss, or may otherwise be specified in terms of some relative priority of access to network resources. Service differentiation is desired to accommodate heterogeneous application requirements and user expectations, and to permit differentiated pricing of Internet service.

This architecture is composed of a number of functional elements implemented in network nodes, including a small set of per-hop forwarding behaviors, packet classification functions, and traffic conditioning functions including metering, marking, shaping, and policing. This architecture achieves scalability by implementing complex classification and conditioning functions only at network boundary nodes, and by applying per-hop behaviors to aggregates of traffic which have been appropriately marked using the DS field in the IPv4 or IPv6 headers [DSFIELD]. Per-hop behaviors are defined to permit a reasonably granular means of allocating buffer and bandwidth resources at each node among competing traffic streams. Per-application flow or per-customer forwarding state need not be maintained within the core of the network. A distinction is maintained between:

- o the service provided to a traffic aggregate,
- o the conditioning functions and per-hop behaviors used to realize services,
- o the DS field value (DS codepoint) used to mark packets to select a per-hop behavior, and
- o the particular node implementation mechanisms which realize a per-hop behavior.

Service provisioning and traffic conditioning policies are sufficiently decoupled from the forwarding behaviors within the network interior to permit implementation of a wide variety of service behaviors, with room for future expansion.

This architecture only provides service differentiation in one direction of traffic flow and is therefore asymmetric. Development of a complementary symmetric architecture is a topic of current research but is outside the scope of this document; see for example [EXPLICIT].

Sect. 1.2 is a glossary of terms used within this document. Sec. 1.3 lists requirements addressed by this architecture, and Sec. 1.4 provides a brief comparison to other approaches for service differentiation. Sec. 2 discusses the components of the architecture

in detail. Sec. 3 proposes guidelines for per-hop behavior specifications. Sec. 4 discusses interoperability issues with nodes and networks which do not implement differentiated services as defined in this document and in [DSFIELD]. Sec. 5 discusses issues with multicast service delivery. Sec. 6 addresses security and tunnel considerations.

1.2 Terminology

This section gives a general conceptual overview of the terms used in this document. Some of these terms are more precisely defined in later sections of this document.

Behavior Aggregate (BA)	a DS behavior aggregate.
BA classifier	a classifier that selects packets based only on the contents of the DS field.
Boundary link	a link connecting the edge nodes of two domains.
Classifier	an entity which selects packets based on the content of packet headers according to defined rules.
DS behavior aggregate	a collection of packets with the same DS codepoint crossing a link in a particular direction.
DS boundary node	a DS node that connects one DS domain to a node either in another DS domain or in a domain that is not DS-capable.
DS-capable	capable of implementing differentiated services as described in this architecture; usually used in reference to a domain consisting of DS-compliant nodes.
DS codepoint	a specific value of the DSCP portion of the DS field, used to select a PHB.
DS-compliant	enabled to support differentiated services functions and behaviors as defined in [DSFIELD], this document, and other differentiated services documents; usually used in reference to a node or device.

DS domain	a DS-capable domain; a contiguous set of nodes which operate with a common set of service provisioning policies and PHB definitions.
DS egress node	a DS boundary node in its role in handling traffic as it leaves a DS domain.
DS ingress node	a DS boundary node in its role in handling traffic as it enters a DS domain.
DS interior node	a DS node that is not a DS boundary node.
DS field	the IPv4 header TOS octet or the IPv6 Traffic Class octet when interpreted in conformance with the definition given in [DSFIELD]. The bits of the DSCP field encode the DS codepoint, while the remaining bits are currently unused.
DS node	a DS-compliant node.
DS region	a set of contiguous DS domains which can offer differentiated services over paths across those DS domains.
Downstream DS domain	the DS domain downstream of traffic flow on a boundary link.
Dropper	a device that performs dropping.
Dropping	the process of discarding packets based on specified rules; policing.
Legacy node	a node which implements IPv4 Precedence as defined in [RFC791,RFC1812] but which is otherwise not DS-compliant.
Marker	a device that performs marking.
Marking	the process of setting the DS codepoint in a packet based on defined rules; pre-marking, re-marking.
Mechanism	a specific algorithm or operation (e.g., queueing discipline) that is implemented in a node to realize a set of one or more per-hop behaviors.

Meter	a device that performs metering.
Metering	the process of measuring the temporal properties (e.g., rate) of a traffic stream selected by a classifier. The instantaneous state of this process may be used to affect the operation of a marker, shaper, or dropper, and/or may be used for accounting and measurement purposes.
Microflow	a single instance of an application-to-application flow of packets which is identified by source address, source port, destination address, destination port and protocol id.
MF Classifier	a multi-field (MF) classifier which selects packets based on the content of some arbitrary number of header fields; typically some combination of source address, destination address, DS field, protocol ID, source port and destination port.
Per-Hop-Behavior (PHB)	the externally observable forwarding behavior applied at a DS-compliant node to a DS behavior aggregate.
PHB group	a set of one or more PHBs that can only be meaningfully specified and implemented simultaneously, due to a common constraint applying to all PHBs in the set such as a queue servicing or queue management policy. A PHB group provides a service building block that allows a set of related forwarding behaviors to be specified together (e.g., four dropping priorities). A single PHB is a special case of a PHB group.
Policing	the process of discarding packets (by a dropper) within a traffic stream in accordance with the state of a corresponding meter enforcing a traffic profile.
Pre-mark	to set the DS codepoint of a packet prior to entry into a downstream DS domain.

Provider DS domain	the DS-capable provider of services to a source domain.
Re-mark	to change the DS codepoint of a packet, usually performed by a marker in accordance with a TCA.
Service	the overall treatment of a defined subset of a customer's traffic within a DS domain or end-to-end.
Service Level Agreement (SLA)	a service contract between a customer and a service provider that specifies the forwarding service a customer should receive. A customer may be a user organization (source domain) or another DS domain (upstream domain). A SLA may include traffic conditioning rules which constitute a TCA in whole or in part.
Service Provisioning Policy	a policy which defines how traffic conditioners are configured on DS boundary nodes and how traffic streams are mapped to DS behavior aggregates to achieve a range of services.
Shaper	a device that performs shaping.
Shaping	the process of delaying packets within a traffic stream to cause it to conform to some defined traffic profile.
Source domain	a domain which contains the node(s) originating the traffic receiving a particular service.
Traffic conditioner	an entity which performs traffic conditioning functions and which may contain meters, markers, droppers, and shapers. Traffic conditioners are typically deployed in DS boundary nodes only. A traffic conditioner may re-mark a traffic stream or may discard or shape packets to alter the temporal characteristics of the stream and bring it into compliance with a traffic profile.

Traffic conditioning	control functions performed to enforce rules specified in a TCA, including metering, marking, shaping, and policing.
Traffic Conditioning Agreement (TCA)	an agreement specifying classifier rules and any corresponding traffic profiles and metering, marking, discarding and/or shaping rules which are to apply to the traffic streams selected by the classifier. A TCA encompasses all of the traffic conditioning rules explicitly specified within a SLA along with all of the rules implicit from the relevant service requirements and/or from a DS domain's service provisioning policy.
Traffic profile	a description of the temporal properties of a traffic stream such as rate and burst size.
Traffic stream	an administratively significant set of one or more microflows which traverse a path segment. A traffic stream may consist of the set of active microflows which are selected by a particular classifier.
Upstream DS domain	the DS domain upstream of traffic flow on a boundary link.

1.3 Requirements

The history of the Internet has been one of continuous growth in the number of hosts, the number and variety of applications, and the capacity of the network infrastructure, and this growth is expected to continue for the foreseeable future. A scalable architecture for service differentiation must be able to accommodate this continued growth.

The following requirements were identified and are addressed in this architecture:

- should accommodate a wide variety of services and provisioning policies, extending end-to-end or within a particular (set of) network(s),

- should allow decoupling of the service from the particular application in use,

- should work with existing applications without the need for application programming interface changes or host software modifications (assuming suitable deployment of classifiers, markers, and other traffic conditioning functions),

- should decouple traffic conditioning and service provisioning functions from forwarding behaviors implemented within the core network nodes,

- should not depend on hop-by-hop application signaling,

- should require only a small set of forwarding behaviors whose implementation complexity does not dominate the cost of a network device, and which will not introduce bottlenecks for future high-speed system implementations,

- should avoid per-microflow or per-customer state within core network nodes,

- should utilize only aggregated classification state within the network core,

- should permit simple packet classification implementations in core network nodes (BA classifier),

- should permit reasonable interoperability with non-DS-compliant network nodes,

- should accommodate incremental deployment.

1.4 Comparisons with Other Approaches

The differentiated services architecture specified in this document can be contrasted with other existing models of service differentiation. We classify these alternative models into the following categories: relative priority marking, service marking, label switching, Integrated Services/RSVP, and static per-hop classification.

Examples of the relative priority marking model include IPv4 Precedence marking as defined in [RFC791], 802.5 Token Ring priority [TR], and the default interpretation of 802.1p traffic classes [802.1p]. In this model the application, host, or proxy node selects a relative priority or "precedence" for a packet (e.g., delay or discard priority), and the network nodes along the transit path apply the appropriate priority forwarding behavior corresponding to the priority value within the packet's header. Our architecture can be considered as a refinement to this model, since we more clearly

specify the role and importance of boundary nodes and traffic conditioners, and since our per-hop behavior model permits more general forwarding behaviors than relative delay or discard priority.

An example of a service marking model is IPv4 TOS as defined in [RFC1349]. In this example each packet is marked with a request for a "type of service", which may include "minimize delay", "maximize throughput", "maximize reliability", or "minimize cost". Network nodes may select routing paths or forwarding behaviors which are suitably engineered to satisfy the service request. This model is subtly different from our architecture. Note that we do not describe the use of the DS field as an input to route selection. The TOS markings defined in [RFC1349] are very generic and do not span the range of possible service semantics. Furthermore, the service request is associated with each individual packet, whereas some service semantics may depend on the aggregate forwarding behavior of a sequence of packets. The service marking model does not easily accommodate growth in the number and range of future services (since the codepoint space is small) and involves configuration of the "TOS->forwarding behavior" association in each core network node. Standardizing service markings implies standardizing service offerings, which is outside the scope of the IETF. Note that provisions are made in the allocation of the DS codepoint space to allow for locally significant codepoints which may be used by a provider to support service marking semantics [DSFIELD].

Examples of the label switching (or virtual circuit) model include Frame Relay, ATM, and MPLS [FRELAY, ATM]. In this model path forwarding state and traffic management or QoS state is established for traffic streams on each hop along a network path. Traffic aggregates of varying granularity are associated with a label switched path at an ingress node, and packets/cells within each label switched path are marked with a forwarding label that is used to lookup the next-hop node, the per-hop forwarding behavior, and the replacement label at each hop. This model permits finer granularity resource allocation to traffic streams, since label values are not globally significant but are only significant on a single link; therefore resources can be reserved for the aggregate of packets/ cells received on a link with a particular label, and the label switching semantics govern the next-hop selection, allowing a traffic stream to follow a specially engineered path through the network. This improved granularity comes at the cost of additional management and configuration requirements to establish and maintain the label switched paths. In addition, the amount of forwarding state maintained at each node scales in proportion to the number of edge nodes of the network in the best case (assuming multipoint-to-point

label switched paths), and it scales in proportion with the square of the number of edge nodes in the worst case, when edge-edge label switched paths with provisioned resources are employed.

The Integrated Services/RSVP model relies upon traditional datagram forwarding in the default case, but allows sources and receivers to exchange signaling messages which establish additional packet classification and forwarding state on each node along the path between them [RFC1633, RSVP]. In the absence of state aggregation, the amount of state on each node scales in proportion to the number of concurrent reservations, which can be potentially large on high-speed links. This model also requires application support for the RSVP signaling protocol. Differentiated services mechanisms can be utilized to aggregate Integrated Services/RSVP state in the core of the network [Bernet].

A variant of the Integrated Services/RSVP model eliminates the requirement for hop-by-hop signaling by utilizing only "static" classification and forwarding policies which are implemented in each node along a network path. These policies are updated on administrative timescales and not in response to the instantaneous mix of microflows active in the network. The state requirements for this variant are potentially worse than those encountered when RSVP is used, especially in backbone nodes, since the number of static policies that might be applicable at a node over time may be larger than the number of active sender-receiver sessions that might have installed reservation state on a node. Although the support of large numbers of classifier rules and forwarding policies may be computationally feasible, the management burden associated with installing and maintaining these rules on each node within a backbone network which might be traversed by a traffic stream is substantial.

Although we contrast our architecture with these alternative models of service differentiation, it should be noted that links and nodes employing these techniques may be utilized to extend differentiated services behaviors and semantics across a layer-2 switched infrastructure (e.g., 802.1p LANs, Frame Relay/ATM backbones) interconnecting DS nodes, and in the case of MPLS may be used as an alternative intra-domain implementation technology. The constraints imposed by the use of a specific link-layer technology in particular regions of a DS domain (or in a network providing access to DS domains) may imply the differentiation of traffic on a coarser grain basis. Depending on the mapping of PHBs to different link-layer services and the way in which packets are scheduled over a restricted set of priority classes (or virtual circuits of different category and capacity), all or a subset of the PHBs in use may be supportable (or may be indistinguishable).

2. Differentiated Services Architectural Model

The differentiated services architecture is based on a simple model where traffic entering a network is classified and possibly conditioned at the boundaries of the network, and assigned to different behavior aggregates. Each behavior aggregate is identified by a single DS codepoint. Within the core of the network, packets are forwarded according to the per-hop behavior associated with the DS codepoint. In this section, we discuss the key components within a differentiated services region, traffic classification and conditioning functions, and how differentiated services are achieved through the combination of traffic conditioning and PHB-based forwarding.

2.1 Differentiated Services Domain

A DS domain is a contiguous set of DS nodes which operate with a common service provisioning policy and set of PHB groups implemented on each node. A DS domain has a well-defined boundary consisting of DS boundary nodes which classify and possibly condition ingress traffic to ensure that packets which transit the domain are appropriately marked to select a PHB from one of the PHB groups supported within the domain. Nodes within the DS domain select the forwarding behavior for packets based on their DS codepoint, mapping that value to one of the supported PHBs using either the recommended codepoint->PHB mapping or a locally customized mapping [DSFIELD]. Inclusion of non-DS-compliant nodes within a DS domain may result in unpredictable performance and may impede the ability to satisfy service level agreements (SLAs).

A DS domain normally consists of one or more networks under the same administration; for example, an organization's intranet or an ISP. The administration of the domain is responsible for ensuring that adequate resources are provisioned and/or reserved to support the SLAs offered by the domain.

2.1.1 DS Boundary Nodes and Interior Nodes

A DS domain consists of DS boundary nodes and DS interior nodes. DS boundary nodes interconnect the DS domain to other DS or non-DS-capable domains, whilst DS interior nodes only connect to other DS interior or boundary nodes within the same DS domain.

Both DS boundary nodes and interior nodes must be able to apply the appropriate PHB to packets based on the DS codepoint; otherwise unpredictable behavior may result. In addition, DS boundary nodes may be required to perform traffic conditioning functions as defined by a traffic conditioning agreement (TCA) between their DS domain and

the peering domain which they connect to (see Sec. 2.3.3).

Interior nodes may be able to perform limited traffic conditioning functions such as DS codepoint re-marking. Interior nodes which implement more complex classification and traffic conditioning functions are analogous to DS boundary nodes (see Sec. 2.3.4.4).

A host in a network containing a DS domain may act as a DS boundary node for traffic from applications running on that host; we therefore say that the host is within the DS domain. If a host does not act as a boundary node, then the DS node topologically closest to that host acts as the DS boundary node for that host's traffic.

2.1.2 DS Ingress Node and Egress Node

DS boundary nodes act both as a DS ingress node and as a DS egress node for different directions of traffic. Traffic enters a DS domain at a DS ingress node and leaves a DS domain at a DS egress node. A DS ingress node is responsible for ensuring that the traffic entering the DS domain conforms to any TCA between it and the other domain to which the ingress node is connected. A DS egress node may perform traffic conditioning functions on traffic forwarded to a directly connected peering domain, depending on the details of the TCA between the two domains. Note that a DS boundary node may act as a DS interior node for some set of interfaces.

2.2 Differentiated Services Region

A differentiated services region (DS Region) is a set of one or more contiguous DS domains. DS regions are capable of supporting differentiated services along paths which span the domains within the region.

The DS domains in a DS region may support different PHB groups internally and different codepoint->PHB mappings. However, to permit services which span across the domains, the peering DS domains must each establish a peering SLA which defines (either explicitly or implicitly) a TCA which specifies how transit traffic from one DS domain to another is conditioned at the boundary between the two DS domains.

It is possible that several DS domains within a DS region may adopt a common service provisioning policy and may support a common set of PHB groups and codepoint mappings, thus eliminating the need for traffic conditioning between those DS domains.

2.3 Traffic Classification and Conditioning

Differentiated services are extended across a DS domain boundary by establishing a SLA between an upstream network and a downstream DS domain. The SLA may specify packet classification and re-marking rules and may also specify traffic profiles and actions to traffic streams which are in- or out-of-profile (see Sec. 2.3.2). The TCA between the domains is derived (explicitly or implicitly) from this SLA.

The packet classification policy identifies the subset of traffic which may receive a differentiated service by being conditioned and/or mapped to one or more behavior aggregates (by DS codepoint re-marking) within the DS domain.

Traffic conditioning performs metering, shaping, policing and/or re-marking to ensure that the traffic entering the DS domain conforms to the rules specified in the TCA, in accordance with the domain's service provisioning policy. The extent of traffic conditioning required is dependent on the specifics of the service offering, and may range from simple codepoint re-marking to complex policing and shaping operations. The details of traffic conditioning policies which are negotiated between networks is outside the scope of this document.

2.3.1 Classifiers

Packet classifiers select packets in a traffic stream based on the content of some portion of the packet header. We define two types of classifiers. The BA (Behavior Aggregate) Classifier classifies packets based on the DS codepoint only. The MF (Multi-Field) classifier selects packets based on the value of a combination of one or more header fields, such as source address, destination address, DS field, protocol ID, source port and destination port numbers, and other information such as incoming interface.

Classifiers are used to "steer" packets matching some specified rule to an element of a traffic conditioner for further processing. Classifiers must be configured by some management procedure in accordance with the appropriate TCA.

The classifier should authenticate the information which it uses to classify the packet (see Sec. 6).

Note that in the event of upstream packet fragmentation, MF classifiers which examine the contents of transport-layer header fields may incorrectly classify packet fragments subsequent to the first. A possible solution to this problem is to maintain

fragmentation state; however, this is not a general solution due to the possibility of upstream fragment re-ordering or divergent routing paths. The policy to apply to packet fragments is outside the scope of this document.

2.3.2 Traffic Profiles

A traffic profile specifies the temporal properties of a traffic stream selected by a classifier. It provides rules for determining whether a particular packet is in-profile or out-of-profile. For example, a profile based on a token bucket may look like:

codepoint=X, use token-bucket r, b

The above profile indicates that all packets marked with DS codepoint X should be measured against a token bucket meter with rate r and burst size b. In this example out-of-profile packets are those packets in the traffic stream which arrive when insufficient tokens are available in the bucket. The concept of in- and out-of-profile can be extended to more than two levels, e.g., multiple levels of conformance with a profile may be defined and enforced.

Different conditioning actions may be applied to the in-profile packets and out-of-profile packets, or different accounting actions may be triggered. In-profile packets may be allowed to enter the DS domain without further conditioning; or, alternatively, their DS codepoint may be changed. The latter happens when the DS codepoint is set to a non-Default value for the first time [DSFIELD], or when the packets enter a DS domain that uses a different PHB group or codepoint->PHB mapping policy for this traffic stream. Out-of-profile packets may be queued until they are in-profile (shaped), discarded (policed), marked with a new codepoint (re-marked), or forwarded unchanged while triggering some accounting procedure. Out-of-profile packets may be mapped to one or more behavior aggregates that are "inferior" in some dimension of forwarding performance to the BA into which in-profile packets are mapped.

Note that a traffic profile is an optional component of a TCA and its use is dependent on the specifics of the service offering and the domain's service provisioning policy.

2.3.3 Traffic Conditioners

A traffic conditioner may contain the following elements: meter, marker, shaper, and dropper. A traffic stream is selected by a classifier, which steers the packets to a logical instance of a traffic conditioner. A meter is used (where appropriate) to measure the traffic stream against a traffic profile. The state of the meter

with respect to a particular packet (e.g., whether it is in- or out-of-profile) may be used to affect a marking, dropping, or shaping action.

When packets exit the traffic conditioner of a DS boundary node the DS codepoint of each packet must be set to an appropriate value.

Fig. 1 shows the block diagram of a classifier and traffic conditioner. Note that a traffic conditioner may not necessarily contain all four elements. For example, in the case where no traffic profile is in effect, packets may only pass through a classifier and a marker.

```
                          +-------+
                          |       |-------------------+
                    +----->| Meter |                   |
                    |     |       |--+                |
                    |     +-------+  |                |
                    |                V                V
              +------------+      +--------+      +---------+
              |            |      |        |      | Shaper/ |
 packets =====>| Classifier |=====>| Marker |=====>| Dropper |=====>
              |            |      |        |      |         |
              +------------+      +--------+      +---------+
```

Fig. 1: Logical View of a Packet Classifier and Traffic Conditioner

2.3.3.1 Meters

Traffic meters measure the temporal properties of the stream of packets selected by a classifier against a traffic profile specified in a TCA. A meter passes state information to other conditioning functions to trigger a particular action for each packet which is either in- or out-of-profile (to some extent).

2.3.3.2 Markers

Packet markers set the DS field of a packet to a particular codepoint, adding the marked packet to a particular DS behavior aggregate. The marker may be configured to mark all packets which are steered to it to a single codepoint, or may be configured to mark a packet to one of a set of codepoints used to select a PHB in a PHB group, according to the state of a meter. When the marker changes the codepoint in a packet it is said to have "re-marked" the packet.

2.3.3.3 Shapers

Shapers delay some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. A shaper usually has a finite-size buffer, and packets may be discarded if there is not sufficient buffer space to hold the delayed packets.

2.3.3.4 Droppers

Droppers discard some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. This process is know as "policing" the stream. Note that a dropper can be implemented as a special case of a shaper by setting the shaper buffer size to zero (or a few) packets.

2.3.4 Location of Traffic Conditioners and MF Classifiers

Traffic conditioners are usually located within DS ingress and egress boundary nodes, but may also be located in nodes within the interior of a DS domain, or within a non-DS-capable domain.

2.3.4.1 Within the Source Domain

We define the source domain as the domain containing the node(s) which originate the traffic receiving a particular service. Traffic sources and intermediate nodes within a source domain may perform traffic classification and conditioning functions. The traffic originating from the source domain across a boundary may be marked by the traffic sources directly or by intermediate nodes before leaving the source domain. This is referred to as initial marking or "pre-marking".

Consider the example of a company that has the policy that its CEO's packets should have higher priority. The CEO's host may mark the DS field of all outgoing packets with a DS codepoint that indicates "higher priority". Alternatively, the first-hop router directly connected to the CEO's host may classify the traffic and mark the CEO's packets with the correct DS codepoint. Such high priority traffic may also be conditioned near the source so that there is a limit on the amount of high priority traffic forwarded from a particular source.

There are some advantages to marking packets close to the traffic source. First, a traffic source can more easily take an application's preferences into account when deciding which packets should receive better forwarding treatment. Also, classification of

packets is much simpler before the traffic has been aggregated with packets from other sources, since the number of classification rules which need to be applied within a single node is reduced.

Since packet marking may be distributed across multiple nodes, the source DS domain is responsible for ensuring that the aggregated traffic towards its provider DS domain conforms to the appropriate TCA. Additional allocation mechanisms such as bandwidth brokers or RSVP may be used to dynamically allocate resources for a particular DS behavior aggregate within the provider's network [2BIT, Bernet]. The boundary node of the source domain should also monitor conformance to the TCA, and may police, shape, or re-mark packets as necessary.

2.3.4.2 At the Boundary of a DS Domain

Traffic streams may be classified, marked, and otherwise conditioned on either end of a boundary link (the DS egress node of the upstream domain or the DS ingress node of the downstream domain). The SLA between the domains should specify which domain has responsibility for mapping traffic streams to DS behavior aggregates and conditioning those aggregates in conformance with the appropriate TCA. However, a DS ingress node must assume that the incoming traffic may not conform to the TCA and must be prepared to enforce the TCA in accordance with local policy.

When packets are pre-marked and conditioned in the upstream domain, potentially fewer classification and traffic conditioning rules need to be supported in the downstream DS domain. In this circumstance the downstream DS domain may only need to re-mark or police the incoming behavior aggregates to enforce the TCA. However, more sophisticated services which are path- or source-dependent may require MF classification in the downstream DS domain's ingress nodes.

If a DS ingress node is connected to an upstream non-DS-capable domain, the DS ingress node must be able to perform all necessary traffic conditioning functions on the incoming traffic.

2.3.4.3 In non-DS-Capable Domains

Traffic sources or intermediate nodes in a non-DS-capable domain may employ traffic conditioners to pre-mark traffic before it reaches the ingress of a downstream DS domain. In this way the local policies for classification and marking may be concealed.

2.3.4.4 In Interior DS Nodes

Although the basic architecture assumes that complex classification and traffic conditioning functions are located only in a network's ingress and egress boundary nodes, deployment of these functions in the interior of the network is not precluded. For example, more restrictive access policies may be enforced on a transoceanic link, requiring MF classification and traffic conditioning functionality in the upstream node on the link. This approach may have scaling limits, due to the potentially large number of classification and conditioning rules that might need to be maintained.

2.4 Per-Hop Behaviors

A per-hop behavior (PHB) is a description of the externally observable forwarding behavior of a DS node applied to a particular DS behavior aggregate. "Forwarding behavior" is a general concept in this context. For example, in the event that only one behavior aggregate occupies a link, the observable forwarding behavior (i.e., loss, delay, jitter) will often depend only on the relative loading of the link (i.e., in the event that the behavior assumes a work-conserving scheduling discipline). Useful behavioral distinctions are mainly observed when multiple behavior aggregates compete for buffer and bandwidth resources on a node. The PHB is the means by which a node allocates resources to behavior aggregates, and it is on top of this basic hop-by-hop resource allocation mechanism that useful differentiated services may be constructed.

The most simple example of a PHB is one which guarantees a minimal bandwidth allocation of X% of a link (over some reasonable time interval) to a behavior aggregate. This PHB can be fairly easily measured under a variety of competing traffic conditions. A slightly more complex PHB would guarantee a minimal bandwidth allocation of X% of a link, with proportional fair sharing of any excess link capacity. In general, the observable behavior of a PHB may depend on certain constraints on the traffic characteristics of the associated behavior aggregate, or the characteristics of other behavior aggregates.

PHBs may be specified in terms of their resource (e.g., buffer, bandwidth) priority relative to other PHBs, or in terms of their relative observable traffic characteristics (e.g., delay, loss). These PHBs may be used as building blocks to allocate resources and should be specified as a group (PHB group) for consistency. PHB groups will usually share a common constraint applying to each PHB within the group, such as a packet scheduling or buffer management policy. The relationship between PHBs in a group may be in terms of absolute or relative priority (e.g., discard priority by means of

deterministic or stochastic thresholds), but this is not required (e.g., N equal link shares). A single PHB defined in isolation is a special case of a PHB group.

PHBs are implemented in nodes by means of some buffer management and packet scheduling mechanisms. PHBs are defined in terms of behavior characteristics relevant to service provisioning policies, and not in terms of particular implementation mechanisms. In general, a variety of implementation mechanisms may be suitable for implementing a particular PHB group. Furthermore, it is likely that more than one PHB group may be implemented on a node and utilized within a domain. PHB groups should be defined such that the proper resource allocation between groups can be inferred, and integrated mechanisms can be implemented which can simultaneously support two or more groups. A PHB group definition should indicate possible conflicts with previously documented PHB groups which might prevent simultaneous operation.

As described in [DSFIELD], a PHB is selected at a node by a mapping of the DS codepoint in a received packet. Standardized PHBs have a recommended codepoint. However, the total space of codepoints is larger than the space available for recommended codepoints for standardized PHBs, and [DSFIELD] leaves provisions for locally configurable mappings. A codepoint->PHB mapping table may contain both 1->1 and N->1 mappings. All codepoints must be mapped to some PHB; in the absence of some local policy, codepoints which are not mapped to a standardized PHB in accordance with that PHB's specification should be mapped to the Default PHB.

2.5 Network Resource Allocation

The implementation, configuration, operation and administration of the supported PHB groups in the nodes of a DS Domain should effectively partition the resources of those nodes and the inter-node links between behavior aggregates, in accordance with the domain's service provisioning policy. Traffic conditioners can further control the usage of these resources through enforcement of TCAs and possibly through operational feedback from the nodes and traffic conditioners in the domain. Although a range of services can be deployed in the absence of complex traffic conditioning functions (e.g., using only static marking policies), functions such as policing, shaping, and dynamic re-marking enable the deployment of services providing quantitative performance metrics.

The configuration of and interaction between traffic conditioners and interior nodes should be managed by the administrative control of the domain and may require operational control through protocols and a control entity. There is a wide range of possible control models.

The precise nature and implementation of the interaction between these components is outside the scope of this architecture. However, scalability requires that the control of the domain does not require micro-management of the network resources. The most scalable control model would operate nodes in open-loop in the operational timeframe, and would only require administrative-timescale management as SLAs are varied. This simple model may be unsuitable in some circumstances, and some automated but slowly varying operational control (minutes rather than seconds) may be desirable to balance the utilization of the network against the recent load profile.

3. Per-Hop Behavior Specification Guidelines

Basic requirements for per-hop behavior standardization are given in [DSFIELD]. This section elaborates on that text by describing additional guidelines for PHB (group) specifications. This is intended to help foster implementation consistency. Before a PHB group is proposed for standardization it should satisfy these guidelines, as appropriate, to preserve the integrity of this architecture.

G.1: A PHB standard must specify a recommended DS codepoint selected from the codepoint space reserved for standard mappings [DSFIELD]. Recommended codepoints will be assigned by the IANA. A PHB proposal may recommend a temporary codepoint from the EXP/LU space to facilitate inter-domain experimentation. Determination of a packet's PHB must not require inspection of additional packet header fields beyond the DS field.

G.2: The specification of each newly proposed PHB group should include an overview of the behavior and the purpose of the behavior being proposed. The overview should include a problem or problems statement for which the PHB group is targeted. The overview should include the basic concepts behind the PHB group.These concepts should include, but are not restricted to, queueing behavior, discard behavior, and output link selection behavior. Lastly, the overview should specify the method by which the PHB group solves the problem or problems specified in the problem statement.

G.3: A PHB group specification should indicate the number of individual PHBs specified. In the event that multiple PHBs are specified, the interactions between these PHBs and constraints that must be respected globally by all the PHBs within the group should be clearly specified. As an example, the specification must indicate whether the probability of packet reordering within a microflow is increased if different packets in that microflow are marked for different PHBs within the group.

G.4: When proper functioning of a PHB group is dependent on constraints such as a provisioning restriction, then the PHB definition should describe the behavior when these constraints are violated. Further, if actions such as packet discard or re-marking are required when these constraints are violated, then these actions should be specifically stipulated.

G.5: A PHB group may be specified for local use within a domain in order to provide some domain-specific functionality or domain-specific services. In this event, the PHB specification is useful for providing vendors with a consistent definition of the PHB group. However, any PHB group which is defined for local use should not be considered for standardization, but may be published as an Informational RFC. In contrast, a PHB group which is intended for general use will follow a stricter standardization process. Therefore all PHB proposals should specifically state whether they are to be considered for general or local use.

It is recognized that PHB groups can be designed with the intent of providing host-to-host, WAN edge-to-WAN edge, and/or domain edge-to-domain edge services. Use of the term "end-to-end" in a PHB definition should be interpreted to mean "host-to-host" for consistency.

Other PHB groups may be defined and deployed locally within domains, for experimental or operational purposes. There is no requirement that these PHB groups must be publicly documented, but they should utilize DS codepoints from one of the EXP/LU pools as defined in [DSFIELD].

G.6: It may be possible or appropriate for a packet marked for a PHB within a PHB group to be re-marked to select another PHB within the group; either within a domain or across a domain boundary. Typically there are three reasons for such PHB modification:

a. The codepoints associated with the PHB group are collectively intended to carry state about the network,
b. Conditions exist which require PHB promotion or demotion of a packet (this assumes that PHBs within the group can be ranked in some order),
c. The boundary between two domains is not covered by a SLA. In this case the codepoint/PHB to select when crossing the boundary link will be determined by the local policy of the upstream domain.

A PHB specification should clearly state the circumstances under which packets marked for a PHB within a PHB group may, or should be modified (e.g., promoted or demoted) to another PHB within the group. If it is undesirable for a packet's PHB to be modified, the

specification should clearly state the consequent risks when the PHB is modified. A possible risk to changing a packet's PHB, either within or outside a PHB group, is a higher probability of packet re-ordering within a microflow. PHBs within a group may carry some host-to-host, WAN edge-to-WAN edge, and/or domain edge-to-domain edge semantics which may be difficult to duplicate if packets are re-marked to select another PHB from the group (or otherwise).

For certain PHB groups, it may be appropriate to reflect a state change in the node by re-marking packets to specify another PHB from within the group. If a PHB group is designed to reflect the state of a network, the PHB definition must adequately describe the relationship between the PHBs and the states they reflect. Further, if these PHBs limit the forwarding actions a node can perform in some way, these constraints may be specified as actions the node should, or must perform.

G.7: A PHB group specification should include a section defining the implications of tunneling on the utility of the PHB group. This section should specify the implications for the utility of the PHB group of a newly created outer header when the original DS field of the inner header is encapsulated in a tunnel. This section should also discuss what possible changes should be applied to the inner header at the egress of the tunnel, when both the codepoints from the inner header and the outer header are accessible (see Sec. 6.2).

G.8: The process of specifying PHB groups is likely to be incremental in nature. When new PHB groups are proposed, their known interactions with previously specified PHB groups should be documented. When a new PHB group is created, it can be entirely new in scope or it can be an extension to an existing PHB group. If the PHB group is entirely independent of some or all of the existing PHB specifications, a section should be included in the PHB specification which details how the new PHB group can co-exist with those PHB groups already standardized. For example, this section might indicate the possibility of packet re-ordering within a microflow for packets marked by codepoints associated with two separate PHB groups. If concurrent operation of two (or more) different PHB groups in the same node is impossible or detrimental this should be stated. If the concurrent operation of two (or more) different PHB groups requires some specific behaviors by the node when packets marked for PHBs from these different PHB groups are being processed by the node at the same time, these behaviors should be stated.

Care should be taken to avoid circularity in the definitions of PHB groups.

If the proposed PHB group is an extension to an existing PHB group, a section should be included in the PHB group specification which details how this extension interoperates with the behavior being extended. Further, if the extension alters or more narrowly defines the existing behavior in some way, this should also be clearly indicated.

G.9: Each PHB specification should include a section specifying minimal conformance requirements for implementations of the PHB group. This conformance section is intended to provide a means for specifying the details of a behavior while allowing for implementation variation to the extent permitted by the PHB specification. This conformance section can take the form of rules, tables, pseudo-code, or tests.

G.10: A PHB specification should include a section detailing the security implications of the behavior. This section should include a discussion of the re-marking of the inner header's codepoint at the egress of a tunnel and its effect on the desired forwarding behavior.

Further, this section should also discuss how the proposed PHB group could be used in denial-of-service attacks, reduction of service contract attacks, and service contract violation attacks. Lastly, this section should discuss possible means for detecting such attacks as they are relevant to the proposed behavior.

G.11: A PHB specification should include a section detailing configuration and management issues which may affect the operation of the PHB and which may impact candidate services that might utilize the PHB.

G.12: It is strongly recommended that an appendix be provided with each PHB specification that considers the implications of the proposed behavior on current and potential services. These services could include but are not restricted to be user-specific, device-specific, domain-specific or end-to-end services. It is also strongly recommended that the appendix include a section describing how the services are verified by users, devices, and/or domains.

G.13: It is recommended that an appendix be provided with each PHB specification that is targeted for local use within a domain, providing guidance for PHB selection for packets which are forwarded into a peer domain which does not support the PHB group.

G.14: It is recommended that an appendix be provided with each PHB specification which considers the impact of the proposed PHB group on existing higher-layer protocols. Under some circumstances PHBs may allow for possible changes to higher-layer protocols which may increase or decrease the utility of the proposed PHB group.

G.15: It is recommended that an appendix be provided with each PHB specification which recommends mappings to link-layer QoS mechanisms to support the intended behavior of the PHB across a shared-medium or switched link-layer. The determination of the most appropriate mapping between a PHB and a link-layer QoS mechanism is dependent on many factors and is outside the scope of this document; however, the specification should attempt to offer some guidance.

4. Interoperability with Non-Differentiated Services-Compliant Nodes

We define a non-differentiated services-compliant node (non-DS-compliant node) as any node which does not interpret the DS field as specified in [DSFIELD] and/or does not implement some or all of the standardized PHBs (or those in use within a particular DS domain). This may be due to the capabilities or configuration of the node. We define a legacy node as a special case of a non-DS-compliant node which implements IPv4 Precedence classification and forwarding as defined in [RFC791, RFC1812], but which is otherwise not DS-compliant. The precedence values in the IPv4 TOS octet are compatible by intention with the Class Selector Codepoints defined in [DSFIELD], and the precedence forwarding behaviors defined in [RFC791, RFC1812] comply with the Class Selector PHB Requirements also defined in [DSFIELD]. A key distinction between a legacy node and a DS-compliant node is that the legacy node may or may not interpret bits 3-6 of the TOS octet as defined in [RFC1349] (the "DTRC" bits); in practice it will not interpret these bit as specified in [DSFIELD]. We assume that the use of the TOS markings defined in [RFC1349] is deprecated. Nodes which are non-DS-compliant and which are not legacy nodes may exhibit unpredictable forwarding behaviors for packets with non-zero DS codepoints.

Differentiated services depend on the resource allocation mechanisms provided by per-hop behavior implementations in nodes. The quality or statistical assurance level of a service may break down in the event that traffic transits a non-DS-compliant node, or a non-DS-capable domain.

We will examine two separate cases. The first case concerns the use of non-DS-compliant nodes within a DS domain. Note that PHB forwarding is primarily useful for allocating scarce node and link resources in a controlled manner. On high-speed, lightly loaded links, the worst-case packet delay, jitter, and loss may be

negligible, and the use of a non-DS-compliant node on the upstream end of such a link may not result in service degradation. In more realistic circumstances, the lack of PHB forwarding in a node may make it impossible to offer low-delay, low-loss, or provisioned bandwidth services across paths which traverse the node. However, use of a legacy node may be an acceptable alternative, assuming that the DS domain restricts itself to using only the Class Selector Codepoints defined in [DSFIELD], and assuming that the particular precedence implementation in the legacy node provides forwarding behaviors which are compatible with the services offered along paths which traverse that node. Note that it is important to restrict the codepoints in use to the Class Selector Codepoints, since the legacy node may or may not interpret bits 3-5 in accordance with [RFC1349], thereby resulting in unpredictable forwarding results.

The second case concerns the behavior of services which traverse non-DS-capable domains. We assume for the sake of argument that a non-DS-capable domain does not deploy traffic conditioning functions on domain boundary nodes; therefore, even in the event that the domain consists of legacy or DS-compliant interior nodes, the lack of traffic enforcement at the boundaries will limit the ability to consistently deliver some types of services across the domain. A DS domain and a non-DS-capable domain may negotiate an agreement which governs how egress traffic from the DS-domain should be marked before entry into the non-DS-capable domain. This agreement might be monitored for compliance by traffic sampling instead of by rigorous traffic conditioning. Alternatively, where there is knowledge that the non-DS-capable domain consists of legacy nodes, the upstream DS domain may opportunistically re-mark differentiated services traffic to one or more of the Class Selector Codepoints. Where there is no knowledge of the traffic management capabilities of the downstream domain, and no agreement in place, a DS domain egress node may choose to re-mark DS codepoints to zero, under the assumption that the non-DS-capable domain will treat the traffic uniformly with best-effort service.

In the event that a non-DS-capable domain peers with a DS domain, traffic flowing from the non-DS-capable domain should be conditioned at the DS ingress node of the DS domain according to the appropriate SLA or policy.

5. Multicast Considerations

Use of differentiated services by multicast traffic introduces a number of issues for service provisioning. First, multicast packets which enter a DS domain at an ingress node may simultaneously take multiple paths through some segments of the domain due to multicast packet replication. In this way they consume more network resources

than unicast packets. Where multicast group membership is dynamic, it is difficult to predict in advance the amount of network resources that may be consumed by multicast traffic originating from an upstream network for a particular group. A consequence of this uncertainty is that it may be difficult to provide quantitative service guarantees to multicast senders. Further, it may be necessary to reserve codepoints and PHBs for exclusive use by unicast traffic, to provide resource isolation from multicast traffic.

The second issue is the selection of the DS codepoint for a multicast packet arriving at a DS ingress node. Because that packet may exit the DS domain at multiple DS egress nodes which peer with multiple downstream domains, the DS codepoint used should not result in the request for a service from a downstream DS domain which is in violation of a peering SLA. When establishing classifier and traffic conditioner state at an DS ingress node for an aggregate of traffic receiving a differentiated service which spans across the egress boundary of the domain, the identity of the adjacent downstream transit domain and the specifics of the corresponding peering SLA can be factored into the configuration decision (subject to routing policy and the stability of the routing infrastructure). In this way peering SLAs with downstream DS domains can be partially enforced at the ingress of the upstream domain, reducing the classification and traffic conditioning burden at the egress node of the upstream domain. This is not so easily performed in the case of multicast traffic, due to the possibility of dynamic group membership. The result is that the service guarantees for unicast traffic may be impacted. One means of addressing this problem is to establish a separate peering SLA for multicast traffic, and to either utilize a particular set of codepoints for multicast packets, or to implement the necessary classification and traffic conditioning mechanisms in the DS egress nodes to provide preferential isolation for unicast traffic in conformance with the peering SLA with the downstream domain.

6. Security and Tunneling Considerations

This section addresses security issues raised by the introduction of differentiated services, primarily the potential for denial-of-service attacks, and the related potential for theft of service by unauthorized traffic (Sec. 6.1). In addition, the operation of differentiated services in the presence of IPsec and its interaction with IPsec are also discussed (Sec. 6.2), as well as auditing requirements (Sec. 6.3). This section considers issues introduced by the use of both IPsec and non-IPsec tunnels.

6.1 Theft and Denial of Service

The primary goal of differentiated services is to allow different levels of service to be provided for traffic streams on a common network infrastructure. A variety of resource management techniques may be used to achieve this, but the end result will be that some packets receive different (e.g., better) service than others. The mapping of network traffic to the specific behaviors that result in different (e.g., better or worse) service is indicated primarily by the DS field, and hence an adversary may be able to obtain better service by modifying the DS field to codepoints indicating behaviors used for enhanced services or by injecting packets with the DS field set to such codepoints. Taken to its limits, this theft of service becomes a denial-of-service attack when the modified or injected traffic depletes the resources available to forward it and other traffic streams. The defense against such theft- and denial-of-service attacks consists of the combination of traffic conditioning at DS boundary nodes along with security and integrity of the network infrastructure within a DS domain.

As described in Sec. 2, DS ingress nodes must condition all traffic entering a DS domain to ensure that it has acceptable DS codepoints. This means that the codepoints must conform to the applicable TCA(s) and the domain's service provisioning policy. Hence, the ingress nodes are the primary line of defense against theft- and denial-of-service attacks based on modified DS codepoints (e.g., codepoints to which the traffic is not entitled), as success of any such attack constitutes a violation of the applicable TCA(s) and/or service provisioning policy. An important instance of an ingress node is that any traffic-originating node in a DS domain is the ingress node for that traffic, and must ensure that all originated traffic carries acceptable DS codepoints.

Both a domain's service provisioning policy and TCAs may require the ingress nodes to change the DS codepoint on some entering packets (e.g., an ingress router may set the DS codepoint of a customer's traffic in accordance with the appropriate SLA). Ingress nodes must condition all other inbound traffic to ensure that the DS codepoints are acceptable; packets found to have unacceptable codepoints must either be discarded or must have their DS codepoints modified to acceptable values before being forwarded. For example, an ingress node receiving traffic from a domain with which no enhanced service agreement exists may reset the DS codepoint to the Default PHB codepoint [DSFIELD]. Traffic authentication may be required to validate the use of some DS codepoints (e.g., those corresponding to enhanced services), and such authentication may be performed by technical means (e.g., IPsec) and/or non-technical means (e.g., the inbound link is known to be connected to exactly one customer site).

An inter-domain agreement may reduce or eliminate the need for ingress node traffic conditioning by making the upstream domain partly or completely responsible for ensuring that traffic has DS codepoints acceptable to the downstream domain. In this case, the ingress node may still perform redundant traffic conditioning checks to reduce the dependence on the upstream domain (e.g., such checks can prevent theft-of-service attacks from propagating across the domain boundary). If such a check fails because the upstream domain is not fulfilling its responsibilities, that failure is an auditable event; the generated audit log entry should include the date/time the packet was received, the source and destination IP addresses, and the DS codepoint that caused the failure. In practice, the limited gains from such checks need to be weighed against their potential performance impact in determining what, if any, checks to perform under these circumstances.

Interior nodes in a DS domain may rely on the DS field to associate differentiated services traffic with the behaviors used to implement enhanced services. Any node doing so depends on the correct operation of the DS domain to prevent the arrival of traffic with unacceptable DS codepoints. Robustness concerns dictate that the arrival of packets with unacceptable DS codepoints must not cause the failure (e.g., crash) of network nodes. Interior nodes are not responsible for enforcing the service provisioning policy (or individual SLAs) and hence are not required to check DS codepoints before using them. Interior nodes may perform some traffic conditioning checks on DS codepoints (e.g., check for DS codepoints that are never used for traffic on a specific link) to improve security and robustness (e.g., resistance to theft-of-service attacks based on DS codepoint modifications). Any detected failure of such a check is an auditable event and the generated audit log entry should include the date/time the packet was received, the source and destination IP addresses, and the DS codepoint that caused the failure. In practice, the limited gains from such checks need to be weighed against their potential performance impact in determining what, if any, checks to perform at interior nodes.

Any link that cannot be adequately secured against modification of DS codepoints or traffic injection by adversaries should be treated as a boundary link (and hence any arriving traffic on that link is treated as if it were entering the domain at an ingress node). Local security policy provides the definition of "adequately secured," and such a definition may include a determination that the risks and consequences of DS codepoint modification and/or traffic injection do not justify any additional security measures for a link. Link security can be enhanced via physical access controls and/or software means such as tunnels that ensure packet integrity.

6.2 IPsec and Tunneling Interactions

The IPsec protocol, as defined in [ESP, AH], does not include the IP header's DS field in any of its cryptographic calculations (in the case of tunnel mode, it is the outer IP header's DS field that is not included). Hence modification of the DS field by a network node has no effect on IPsec's end-to-end security, because it cannot cause any IPsec integrity check to fail. As a consequence, IPsec does not provide any defense against an adversary's modification of the DS field (i.e., a man-in-the-middle attack), as the adversary's modification will also have no effect on IPsec's end-to-end security. In some environments, the ability to modify the DS field without affecting IPsec integrity checks may constitute a covert channel; if it is necessary to eliminate such a channel or reduce its bandwidth, the DS domains should be configured so that the required processing (e.g., set all DS fields on sensitive traffic to a single value) can be performed at DS egress nodes where traffic exits higher security domains.

IPsec's tunnel mode provides security for the encapsulated IP header's DS field. A tunnel mode IPsec packet contains two IP headers: an outer header supplied by the tunnel ingress node and an encapsulated inner header supplied by the original source of the packet. When an IPsec tunnel is hosted (in whole or in part) on a differentiated services network, the intermediate network nodes operate on the DS field in the outer header. At the tunnel egress node, IPsec processing includes stripping the outer header and forwarding the packet (if required) using the inner header. If the inner IP header has not been processed by a DS ingress node for the tunnel egress node's DS domain, the tunnel egress node is the DS ingress node for traffic exiting the tunnel, and hence must carry out the corresponding traffic conditioning responsibilities (see Sec. 6.1). If the IPsec processing includes a sufficiently strong cryptographic integrity check of the encapsulated packet (where sufficiency is determined by local security policy), the tunnel egress node can safely assume that the DS field in the inner header has the same value as it had at the tunnel ingress node. This allows a tunnel egress node in the same DS domain as the tunnel ingress node, to safely treat a packet passing such an integrity check as if it had arrived from another node within the same DS domain, omitting the DS ingress node traffic conditioning that would otherwise be required. An important consequence is that otherwise insecure links internal to a DS domain can be secured by a sufficiently strong IPsec tunnel.

This analysis and its implications apply to any tunneling protocol that performs integrity checks, but the level of assurance of the inner header's DS field depends on the strength of the integrity

check performed by the tunneling protocol. In the absence of sufficient assurance for a tunnel that may transit nodes outside the current DS domain (or is otherwise vulnerable), the encapsulated packet must be treated as if it had arrived at a DS ingress node from outside the domain.

The IPsec protocol currently requires that the inner header's DS field not be changed by IPsec decapsulation processing at a tunnel egress node. This ensures that an adversary's modifications to the DS field cannot be used to launch theft- or denial-of-service attacks across an IPsec tunnel endpoint, as any such modifications will be discarded at the tunnel endpoint. This document makes no change to that IPsec requirement.

If the IPsec specifications are modified in the future to permit a tunnel egress node to modify the DS field in an inner IP header based on the DS field value in the outer header (e.g., copying part or all of the outer DS field to the inner DS field), then additional considerations would apply. For a tunnel contained entirely within a single DS domain and for which the links are adequately secured against modifications of the outer DS field, the only limits on inner DS field modifications would be those imposed by the domain's service provisioning policy. Otherwise, the tunnel egress node performing such modifications would be acting as a DS ingress node for traffic exiting the tunnel and must carry out the traffic conditioning responsibilities of an ingress node, including defense against theft- and denial-of-service attacks (See Sec. 6.1). If the tunnel enters the DS domain at a node different from the tunnel egress node, the tunnel egress node may depend on the upstream DS ingress node having ensured that the outer DS field values are acceptable. Even in this case, there are some checks that can only be performed by the tunnel egress node (e.g., a consistency check between the inner and outer DS codepoints for an encrypted tunnel). Any detected failure of such a check is an auditable event and the generated audit log entry should include the date/time the packet was received, the source and destination IP addresses, and the DS codepoint that was unacceptable.

An IPsec tunnel can be viewed in at least two different ways from an architectural perspective. If the tunnel is viewed as a logical single hop "virtual wire", the actions of intermediate nodes in forwarding the tunneled traffic should not be visible beyond the ends of the tunnel and hence the DS field should not be modified as part of decapsulation processing. In contrast, if the tunnel is viewed as a multi-hop participant in forwarding traffic, then modification of the DS field as part of tunnel decapsulation processing may be desirable. A specific example of the latter situation occurs when a tunnel terminates at an interior node of a DS domain at which the domain administrator does not wish to deploy traffic conditioning

logic (e.g., to simplify traffic management). This could be supported by using the DS codepoint in the outer IP header (which was subject to traffic conditioning at the DS ingress node) to reset the DS codepoint in the inner IP header, effectively moving DS ingress traffic conditioning responsibilities from the IPsec tunnel egress node to the appropriate upstream DS ingress node (which must already perform that function for unencapsulated traffic).

6.3 Auditing

Not all systems that support differentiated services will implement auditing. However, if differentiated services support is incorporated into a system that supports auditing, then the differentiated services implementation should also support auditing. If such support is present the implementation must allow a system administrator to enable or disable auditing for differentiated services as a whole, and may allow such auditing to be enabled or disabled in part.

For the most part, the granularity of auditing is a local matter. However, several auditable events are identified in this document and for each of these events a minimum set of information that should be included in an audit log is defined. Additional information (e.g., packets related to the one that triggered the auditable event) may also be included in the audit log for each of these events, and additional events, not explicitly called out in this specification, also may result in audit log entries. There is no requirement for the receiver to transmit any message to the purported sender in response to the detection of an auditable event, because of the potential to induce denial of service via such action.

7. Acknowledgements

This document has benefitted from earlier drafts by Steven Blake, David Clark, Ed Ellesson, Paul Ferguson, Juha Heinanen, Van Jacobson, Kalevi Kilkki, Kathleen Nichols, Walter Weiss, John Wroclawski, and Lixia Zhang.

The authors would like to acknowledge the following individuals for their helpful comments and suggestions: Kathleen Nichols, Brian Carpenter, Konstantinos Dovrolis, Shivkumar Kalyana, Wu-chang Feng, Marty Borden, Yoram Bernet, Ronald Bonica, James Binder, Borje Ohlman, Alessio Casati, Scott Brim, Curtis Villamizar, Hamid Ould-Brahi, Andrew Smith, John Renwick, Werner Almesberger, Alan O'Neill, James Fu, and Bob Braden.

8. References

[802.1p] ISO/IEC Final CD 15802-3 Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Common specifications - Part 3: Media Access Control (MAC) bridges, (current draft available as IEEE P802.1D/D15).

[AH] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[ATM] ATM Traffic Management Specification Version 4.0 <af-tm-0056.000>, ATM Forum, April 1996.

[Bernet] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, K. Nichols, and M. Speer, "A Framework for Use of RSVP with Diff-serv Networks", Work in Progress.

[DSFIELD] Nichols, K., Blake, S., Baker, F. and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[EXPLICIT] D. Clark and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service", IEEE/ACM Trans. on Networking, vol. 6, no. 4, August 1998, pp. 362-373.

[ESP] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[FRELAY] ANSI T1S1, "DSSI Core Aspects of Frame Rely", March 1990.

[RFC791] Postel, J., Editor, "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC1349] Almquist, P., "Type of Service in the Internet Protocol Suite", RFC 1349, July 1992.

[RFC1633] Braden, R., Clark, D. and S. Shenker, "Integrated Services in the Internet Architecture: An Overview", RFC 1633, July 1994.

[RFC1812] Baker, F., Editor, "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RSVP] Braden, B., Zhang, L., Berson S., Herzog, S. and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[2BIT] K. Nichols, V. Jacobson, and L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet", ftp://ftp.ee.lbl.gov/papers/dsarch.pdf, November 1997.

[TR] ISO/IEC 8802-5 Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Common specifications - Part 5: Token Ring Access Method and Physical Layer Specifications, (also ANSI/IEEE Std 802.5-1995), 1995.

Authors' Addresses

Steven Blake
Torrent Networking Technologies
3000 Aerial Center, Suite 140
Morrisville, NC 27560

Phone: +1-919-468-8466 x232
EMail: slblake@torrentnet.com

David L. Black
EMC Corporation
35 Parkwood Drive
Hopkinton, MA 01748

Phone: +1-508-435-1000 x76140
EMail: black_david@emc.com

Mark A. Carlson
Sun Microsystems, Inc.
2990 Center Green Court South
Boulder, CO 80301

Phone: +1-303-448-0048 x115
EMail: mark.carlson@sun.com

Elwyn Davies
Nortel UK
London Road
Harlow, Essex CM17 9NA, UK

Phone: +44-1279-405498
EMail: elwynd@nortel.co.uk

Zheng Wang
Bell Labs Lucent Technologies
101 Crawfords Corner Road
Holmdel, NJ 07733

EMail: zhwang@bell-labs.com

Walter Weiss
Lucent Technologies
300 Baker Avenue, Suite 100
Concord, MA 01742-2168

EMail: wweiss@lucent.com

Full Copyright Statement

ANEXO C: RFC 2597 Assured Forwarding PHB Group

Network Working Group
Request for Comments: 2597
Category: Standards Track

J. Heinanen
Telia Finland
F. Baker
Cisco Systems
W. Weiss
Lucent Technologies
J. Wroclawski
MIT LCS
June 1999

Assured Forwarding PHB Group

Status of this Memo

Copyright Notice

Abstract

This document defines a general use Differentiated Services (DS) [Blake] Per-Hop-Behavior (PHB) Group called Assured Forwarding (AF). The AF PHB group provides delivery of IP packets in four independently forwarded AF classes. Within each AF class, an IP packet can be assigned one of three different levels of drop precedence. A DS node does not reorder IP packets of the same microflow if they belong to the same AF class.

1. Purpose and Overview

There is a demand to provide assured forwarding of IP packets over the Internet. In a typical application, a company uses the Internet to interconnect its geographically distributed sites and wants an assurance that IP packets within this intranet are forwarded with high probability as long as the aggregate traffic from each site does not exceed the subscribed information rate (profile). It is desirable that a site may exceed the subscribed profile with the understanding that the excess traffic is not delivered with as high probability as the traffic that is within the profile. It is also

important that the network does not reorder packets that belong to the same microflow, as defined in [Nichols], no matter if they are in or out of the profile.

Assured Forwarding (AF) PHB group is a means for a provider DS domain to offer different levels of forwarding assurances for IP packets received from a customer DS domain. Four AF classes are defined, where each AF class is in each DS node allocated a certain amount of forwarding resources (buffer space and bandwidth). IP packets that wish to use the services provided by the AF PHB group are assigned by the customer or the provider DS domain into one or more of these AF classes according to the services that the customer has subscribed to. Further background about this capability and some ways to use it may be found in [Clark].

Within each AF class IP packets are marked (again by the customer or the provider DS domain) with one of three possible drop precedence values. In case of congestion, the drop precedence of a packet determines the relative importance of the packet within the AF class. A congested DS node tries to protect packets with a lower drop precedence value from being lost by preferably discarding packets with a higher drop precedence value.

In a DS node, the level of forwarding assurance of an IP packet thus depends on (1) how much forwarding resources has been allocated to the AF class that the packet belongs to, (2) what is the current load of the AF class, and, in case of congestion within the class, (3) what is the drop precedence of the packet.

For example, if traffic conditioning actions at the ingress of the provider DS domain make sure that an AF class in the DS nodes is only moderately loaded by packets with the lowest drop precedence value and is not overloaded by packets with the two lowest drop precedence values, then the AF class can offer a high level of forwarding assurance for packets that are within the subscribed profile (i.e., marked with the lowest drop precedence value) and offer up to two lower levels of forwarding assurance for the excess traffic.

This document describes the AF PHB group. An otherwise DS-compliant node is not required to implement this PHB group in order to be considered DS-compliant, but when a DS-compliant node is said to implement an AF PHB group, it must conform to the specification in this document.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [Bradner].

2. The AF PHB Group

Assured Forwarding (AF) PHB group provides forwarding of IP packets in N independent AF classes. Within each AF class, an IP packet is assigned one of M different levels of drop precedence. An IP packet that belongs to an AF class i and has drop precedence j is marked with the AF codepoint AFij, where 1 <= i <= N and 1 <= j <= M. Currently, four classes (N=4) with three levels of drop precedence in each class (M=3) are defined for general use. More AF classes or levels of drop precedence MAY be defined for local use.

A DS node SHOULD implement all four general use AF classes. Packets in one AF class MUST be forwarded independently from packets in another AF class, i.e., a DS node MUST NOT aggregate two or more AF classes together.

A DS node MUST allocate a configurable, minimum amount of forwarding resources (buffer space and bandwidth) to each implemented AF class. Each class SHOULD be serviced in a manner to achieve the configured service rate (bandwidth) over both small and large time scales.

An AF class MAY also be configurable to receive more forwarding resources than the minimum when excess resources are available either from other AF classes or from other PHB groups. This memo does not specify how the excess resources should be allocated, but implementations MUST specify what algorithms are actually supported and how they can be parameterized.

Within an AF class, a DS node MUST NOT forward an IP packet with smaller probability if it contains a drop precedence value p than if it contains a drop precedence value q when p < q. Note that this requirement can be fulfilled without needing to dequeue and discard already-queued packets.

Within each AF class, a DS node MUST accept all three drop precedence codepoints and they MUST yield at least two different levels of loss probability. In some networks, particularly in enterprise networks, where transient congestion is a rare and brief occurrence, it may be reasonable for a DS node to implement only two different levels of loss probability per AF class. While this may suffice for some networks, three different levels of loss probability SHOULD be supported in DS domains where congestion is a common occurrence.

If a DS node only implements two different levels of loss probability for an AF class x, the codepoint AFx1 MUST yield the lower loss probability and the codepoints AFx2 and AFx3 MUST yield the higher loss probability.

A DS node MUST NOT reorder AF packets of the same microflow when they belong to the same AF class regardless of their drop precedence. There are no quantifiable timing requirements (delay or delay variation) associated with the forwarding of AF packets.

The relationship between AF classes and other PHBs is described in Section 7 of this memo.

The AF PHB group MAY be used to implement both end-to-end and domain edge-to-domain edge services.

3. Traffic Conditioning Actions

A DS domain MAY at the edge of a domain control the amount of AF traffic that enters or exits the domain at various levels of drop precedence. Such traffic conditioning actions MAY include traffic shaping, discarding of packets, increasing or decreasing the drop precedence of packets, and reassigning of packets to other AF classes. However, the traffic conditioning actions MUST NOT cause reordering of packets of the same microflow.

4. Queueing and Discard Behavior

This section defines the queueing and discard behavior of the AF PHB group. Other aspects of the PHB group's behavior are defined in Section 2.

An AF implementation MUST attempt to minimize long-term congestion within each class, while allowing short-term congestion resulting from bursts. This requires an active queue management algorithm. An example of such an algorithm is Random Early Drop (RED) [Floyd]. This memo does not specify the use of a particular algorithm, but does require that several properties hold.

An AF implementation MUST detect and respond to long-term congestion within each class by dropping packets, while handling short-term congestion (packet bursts) by queueing packets. This implies the presence of a smoothing or filtering function that monitors the instantaneous congestion level and computes a smoothed congestion level. The dropping algorithm uses this smoothed congestion level to determine when packets should be discarded.

The dropping algorithm MUST be insensitive to the short-term traffic characteristics of the microflows using an AF class. That is, flows with different short-term burst shapes but identical longer-term packet rates should have packets discarded with essentially equal probability. One way to achieve this is to use randomness within the dropping function.

The dropping algorithm MUST treat all packets within a single class and precedence level identically. This implies that for any given smoothed congestion level, the discard rate of a particular microflow's packets within a single precedence level will be proportional to that flow's percentage of the total amount of traffic passing through that precedence level.

The congestion indication feedback to the end nodes, and thus the level of packet discard at each drop precedence in relation to congestion, MUST be gradual rather than abrupt, to allow the overall system to reach a stable operating point. One way to do this (RED) uses two (configurable) smoothed congestion level thresholds. When the smoothed congestion level is below the first threshold, no packets of the relevant precedence are discarded. When the smoothed congestion level is between the first and the second threshold, packets are discarded with linearly increasing probability, ranging from zero to a configurable value reached just prior to the second threshold. When the smoothed congestion level is above the second threshold, packets of the relevant precedence are discarded with 100% probability.

To allow the AF PHB to be used in many different operating environments, the dropping algorithm control parameters MUST be independently configurable for each packet drop precedence and for each AF class.

Within the limits above, this specification allows for a range of packet discard behaviors. Inconsistent discard behaviors lead to inconsistent end-to-end service semantics and limit the range of possible uses of the AF PHB in a multi-vendor environment. As experience is gained, future versions of this document may more tightly define specific aspects of the desirable behavior.

5. Tunneling

When AF packets are tunneled, the PHB of the tunneling packet MUST NOT reduce the forwarding assurance of the tunneled AF packet nor cause reordering of AF packets belonging to the same microflow.

6. Recommended Codepoints

Recommended codepoints for the four general use AF classes are given below. These codepoints do not overlap with any other general use PHB groups.

The RECOMMENDED values of the AF codepoints are as follows: AF11 = '001010', AF12 = '001100', AF13 = '001110', AF21 = '010010', AF22 = '010100', AF23 = '010110', AF31 = '011010', AF32 = '011100', AF33 = '011110', AF41 = '100010', AF42 = '100100', and AF43 = '100110'. The table below summarizes the recommended AF codepoint values.

	Class 1	Class 2	Class 3	Class 4
Low Drop Prec	001010	010010	011010	100010
Medium Drop Prec	001100	010100	011100	100100
High Drop Prec	001110	010110	011110	100110

7. Interactions with Other PHB Groups

The AF codepoint mappings recommended above do not interfere with the local use spaces nor the Class Selector codepoints recommended in [Nichols]. The PHBs selected by those Class Selector codepoints may thus coexist with the AF PHB group and retain the forwarding behavior and relationships that was defined for them. In particular, the Default PHB codepoint of '000000' may remain to be used for conventional best effort traffic. Similarly, the codepoints '11x000' may remain to be used for network control traffic.

The AF PHB group, in conjunction with edge traffic conditioning actions that limit the amount of traffic in each AF class to a (generally different) percentage of the class's allocated resources, can be used to obtain the overall behavior implied by the Class Selector PHBs. In this case it may be appropriate within a DS domain to use some or all of the Class Selector codepoints as aliases of AF codepoints.

In addition to the Class Selector PHBs, any other PHB groups may co-exist with the AF PHB group within the same DS domain. However, any AF PHB group implementation should document the following:

(a) Which, if any, other PHB groups may preempt the forwarding resources specifically allocated to each AF PHB class. This preemption MUST NOT happen in normal network operation, but may be appropriate in certain unusual situations - for example, the '11x000' codepoint may preempt AF forwarding resources, to give precedence to unexpectedly high levels of network control traffic when required.

(b) How "excess" resources are allocated between the AF PHB group and other implemented PHB groups. For example, once the minimum allocations are given to each AF class, any remaining resources could be allocated evenly between the AF classes and the Default PHB. In an alternative example, any remaining resources could be allocated to forwarding excess AF traffic, with resources devoted to the Default PHB only when all AF demand is met.

This memo does not specify that any particular relationship hold between AF PHB groups and other implemented PHB groups; it requires only that whatever relationship is chosen be documented. Implementations MAY allow either or both of these relationships to be configurable. It is expected that this level of configuration flexibility will prove valuable to many network administrators.

8. Security Implications

In order to protect itself against denial of service attacks, a provider DS domain SHOULD limit the traffic entering the domain to the subscribed profiles. Also, in order to protect a link to a customer DS domain from denial of service attacks, the provider DS domain SHOULD allow the customer DS domain to specify how the resources of the link are allocated to AF packets. If a service offering requires that traffic marked with an AF codepoint be limited by such attributes as source or destination address, it is the responsibility of the ingress node in a network to verify validity of such attributes.

Other security considerations are covered in [Blake] and [Nichols].

9. Intellectual Property Rights

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information, consult the online list of claimed rights.

10. IANA Considerations

This document allocates twelve codepoints, listed in section 6, in Pool 1 of the code space defined by [Nichols].

Appendix: Example Services

The AF PHB group could be used to implement, for example, the so-called Olympic service, which consists of three service classes: bronze, silver, and gold. Packets are assigned to these three classes so that packets in the gold class experience lighter load (and thus have greater probability for timely forwarding) than packets assigned to the silver class. Same kind of relationship exists between the silver class and the bronze class. If desired, packets within each class may be further separated by giving them either low, medium, or high drop precedence.

The bronze, silver, and gold service classes could in the network be mapped to the AF classes 1, 2, and 3. Similarly, low, medium, and high drop precedence may be mapped to AF drop precedence levels 1, 2, or 3.

The drop precedence level of a packet could be assigned, for example, by using a leaky bucket traffic policer, which has as its parameters a rate and a size, which is the sum of two burst values: a committed burst size and an excess burst size. A packet is assigned low drop precedence if the number of tokens in the bucket is greater than the excess burst size, medium drop precedence if the number of tokens in the bucket is greater than zero, but at most the excess burst size, and high drop precedence if the bucket is empty. It may also be necessary to set an upper limit to the amount of high drop precedence traffic from a customer DS domain in order to avoid the situation where an avalanche of undeliverable high drop precedence packets from one customer DS domain can deny service to possibly deliverable high drop precedence packets from other domains.

Another way to assign the drop precedence level of a packet could be to limit the user traffic of an Olympic service class to a given peak rate and distribute it evenly across each level of drop precedence. This would yield a proportional bandwidth service, which equally apportions available capacity during times of congestion under the assumption that customers with high bandwidth microflows have subscribed to higher peak rates than customers with low bandwidth microflows.

The AF PHB group could also be used to implement a loss and low latency service using an over provisioned AF class, if the maximum arrival rate to that class is known a priori in each DS node. Specification of the required admission control services, however, is beyond the scope of this document. If low loss is not an objective, a low latency service could be implemented without over provisioning by setting a low maximum limit to the buffer space available for an AF class.

References

[Blake] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[Bradner] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[Clark] Clark, D. and Fang, W., Explicit Allocation of Best Effort Packet Delivery Service. IEEE/ACM Transactions on Networking, Volume 6, Number 4, August 1998, pp. 362-373.

[Floyd] Floyd, S., and Jacobson, V., Random Early Detection gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking, Volume 1, Number 4, August 1993, pp. 397-413.

[Nichols] Nichols, K., Blake, S., Baker, F. and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

Authors' Addresses

Juha Heinanen
Telia Finland
Myyrmaentie 2
01600 Vantaa, Finland

EMail: jh@telia.fi

Fred Baker
Cisco Systems
519 Lado Drive
Santa Barbara, California 93111

EMail: fred@cisco.com

Walter Weiss
Lucent Technologies
300 Baker Avenue, Suite 100,
Concord, MA 01742-2168

EMail: wweiss@lucent.com

John Wroclawski
MIT Laboratory for Computer Science
545 Technology Sq.
Cambridge, MA 02139

EMail: jtw@lcs.mit.edu

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

ANEXO D: RFC 2598 Expedited Forwarding PHB Group

Network Working Group
Request for Comments: 2598
Category: Standards Track

V. Jacobson
K. Nichols
Cisco Systems
K. Poduri
Bay Networks
June 1999

An Expedited Forwarding PHB

Status of this Memo

Copyright Notice

Abstract

The definition of PHBs (per-hop forwarding behaviors) is a critical part of the work of the Diffserv Working Group. This document describes a PHB called Expedited Forwarding. We show the generality of this PHB by noting that it can be produced by more than one mechanism and give an example of its use to produce at least one service, a Virtual Leased Line. A recommended codepoint for this PHB is given.

A pdf version of this document is available at ftp://ftp.ee.lbl.gov/papers/ef_phb.pdf

1. Introduction

Network nodes that implement the differentiated services enhancements to IP use a codepoint in the IP header to select a per-hop behavior (PHB) as the specific forwarding treatment for that packet [RFC2474, RFC2475]. This memo describes a particular PHB called expedited forwarding (EF). The EF PHB can be used to build a low loss, low latency, low jitter, assured bandwidth, end-to-end service through DS domains. Such a service appears to the endpoints like a point-to-point connection or a "virtual leased line". This service has also been described as Premium service [2BIT].

Loss, latency and jitter are all due to the queues traffic experiences while transiting the network. Therefore providing low loss, latency and jitter for some traffic aggregate means ensuring that the aggregate sees no (or very small) queues. Queues arise when (short-term) traffic arrival rate exceeds departure rate at some node. Thus a service that ensures no queues for some aggregate is equivalent to bounding rates such that, at every transit node, the aggregate's maximum arrival rate is less than that aggregate's minimum departure rate.

Creating such a service has two parts:

1) Configuring nodes so that the aggregate has a well-defined minimum departure rate. ("Well-defined" means independent of the dynamic state of the node. In particular, independent of the intensity of other traffic at the node.)

2) Conditioning the aggregate (via policing and shaping) so that its arrival rate at any node is always less than that node's configured minimum departure rate.

The EF PHB provides the first part of the service. The network boundary traffic conditioners described in [RFC2475] provide the second part.

The EF PHB is not a mandatory part of the Differentiated Services architecture, i.e., a node is not required to implement the EF PHB in order to be considered DS-compliant. However, when a DS-compliant node claims to implement the EF PHB, the implementation must conform to the specification given in this document.

The next sections describe the EF PHB in detail and give examples of how it might be implemented. The keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", and "MAY" that appear in this document are to be interpreted as described in [Bradner97].

2. Description of EF per-hop behavior

The EF PHB is defined as a forwarding treatment for a particular diffserv aggregate where the departure rate of the aggregate's packets from any diffserv node must equal or exceed a configurable rate. The EF traffic SHOULD receive this rate independent of the intensity of any other traffic attempting to transit the node. It SHOULD average at least the configured rate when measured over any time interval equal to or longer than the time it takes to send an output link MTU sized packet at the configured rate. (Behavior at time scales shorter than a packet time at the configured rate is

deliberately not specified.) The configured minimum rate MUST be settable by a network administrator (using whatever mechanism the node supports for non-volatile configuration).

If the EF PHB is implemented by a mechanism that allows unlimited preemption of other traffic (e.g., a priority queue), the implementation MUST include some means to limit the damage EF traffic could inflict on other traffic (e.g., a token bucket rate limiter). Traffic that exceeds this limit MUST be discarded. This maximum EF rate, and burst size if appropriate, MUST be settable by a network administrator (using whatever mechanism the node supports for non-volatile configuration). The minimum and maximum rates may be the same and configured by a single parameter.

The Appendix describes how this PHB can be used to construct end-to-end services.

2.2 Example Mechanisms to Implement the EF PHB

Several types of queue scheduling mechanisms may be employed to deliver the forwarding behavior described in section 2.1 and thus implement the EF PHB. A simple priority queue will give the appropriate behavior as long as there is no higher priority queue that could preempt the EF for more than a packet time at the configured rate. (This could be accomplished by having a rate policer such as a token bucket associated with each priority queue to bound how much the queue can starve other traffic.)

It’s also possible to use a single queue in a group of queues serviced by a weighted round robin scheduler where the share of the output bandwidth assigned to the EF queue is equal to the configured rate. This could be implemented, for example, using one PHB of a Class Selector Compliant set of PHBs [RFC2474].

Another possible implementation is a CBQ [CBQ] scheduler that gives the EF queue priority up to the configured rate.

All of these mechanisms have the basic properties required for the EF PHB though different choices result in different ancillary behavior such as jitter seen by individual microflows. See Appendix A.3 for simulations that quantify some of these differences.

2.3 Recommended codepoint for this PHB

Codepoint 101110 is recommended for the EF PHB.

2.4 Mutability

Packets marked for EF PHB MAY be remarked at a DS domain boundary only to other codepoints that satisfy the EF PHB. Packets marked for EF PHBs SHOULD NOT be demoted or promoted to another PHB by a DS domain.

2.5 Tunneling

When EF packets are tunneled, the tunneling packets must be marked as EF.

2.6 Interaction with other PHBs

Other PHBs and PHB groups may be deployed in the same DS node or domain with the EF PHB as long as the requirement of section 2.1 is met.

3. Security Considerations

To protect itself against denial of service attacks, the edge of a DS domain MUST strictly police all EF marked packets to a rate negotiated with the adjacent upstream domain. (This rate must be <= the EF PHB configured rate.) Packets in excess of the negotiated rate MUST be dropped. If two adjacent domains have not negotiated an EF rate, the downstream domain MUST use 0 as the rate (i.e., drop all EF marked packets).

Since the end-to-end premium service constructed from the EF PHB requires that the upstream domain police and shape EF marked traffic to meet the rate negotiated with the downstream domain, the downstream domain’s policer should never have to drop packets. Thus these drops SHOULD be noted (e.g., via SNMP traps) as possible security violations or serious misconfiguration. Similarly, since the aggregate EF traffic rate is constrained at every interior node, the EF queue should never overflow so if it does the drops SHOULD be noted as possible attacks or serious misconfiguration.

4. IANA Considerations

This document allocates one codepoint, 101110, in Pool 1 of the code space defined by [RFC2474].

5. References

[Bradner97] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2474] Nichols, K., Blake, S., Baker, F. and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2475] Black, D., Blake, S., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[2BIT] K. Nichols, V. Jacobson, and L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet", Work in Progress, ftp://ftp.ee.lbl.gov/papers/dsarch.pdf

[CBQ] S. Floyd and V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks", IEEE/ACM Transactions on Networking, Vol. 3 no. 4, pp. 365-386, August 1995.

[RFC2415] Poduri, K. and K. Nichols, "Simulation Studies of Increased Initial TCP Window Size", RFC 2415, September 1998.

[LCN] K. Nichols, "Improving Network Simulation with Feedback", Proceedings of LCN ’98, October 1998.

6. Authors' Addresses

Van Jacobson
Cisco Systems, Inc
170 W. Tasman Drive
San Jose, CA 95134-1706

EMail: van@cisco.com

Kathleen Nichols
Cisco Systems, Inc
170 W. Tasman Drive
San Jose, CA 95134-1706

EMail: kmn@cisco.com

Kedarnath Poduri
Bay Networks, Inc.
4401 Great America Parkway
Santa Clara, CA 95052-8185

EMail: kpoduri@baynetworks.com

Appendix A: Example use of and experiences with the EF PHB

A.1 Virtual Leased Line Service

A VLL Service, also known as Premium service [2BIT], is quantified by a peak bandwidth.

A.2 Experiences with its use in ESNET

A prototype of the VLL service has been deployed on DOE’s ESNet backbone. This uses weighted-round-robin queuing features of Cisco 75xx series routers to implement the EF PHB. The early tests have been very successful and work is in progress to make the service available on a routine production basis (see ftp://ftp.ee.lbl.gov/talks/vj-doeqos.pdf and ftp://ftp.ee.lbl.gov/talks/vj-i2qos-may98.pdf for details).

A.3 Simulation Results

A.3.1 Jitter variation

In section 2.2, we pointed out that a number of mechanisms might be used to implement the EF PHB. The simplest of these is a priority queue (PQ) where the arrival rate of the queue is strictly less than its service rate. As jitter comes from the queuing delay along the path, a feature of this implementation is that EF-marked microflows will see very little jitter at their subscribed rate since packets spend little time in queues. The EF PHB does not have an explicit jitter requirement but it is clear from the definition that the expected jitter in a packet stream that uses a service based on the EF PHB will be less with PQ than with best-effort delivery. We used simulation to explore how weighted round-robin (WRR) compares to PQ in jitter. We chose these two since they"re the best and worst cases, respectively, for jitter and we wanted to supply rough guidelines for EF implementers choosing to use WRR or similar mechanisms.

Our simulation model is implemented in a modified ns-2 described in [RFC2415] and [LCN]. We used the CBQ modules included with ns-2 as a basis to implement priority queuing and WRR. Our topology has six hops with decreasing bandwidth in the direction of a single 1.5 Mbps bottleneck link (see figure 6). Sources produce EF-marked packets at an average bit rate equal to their subscribed packet rate. Packets are produced with a variation of +-10% from the interpacket spacing at the subscribed packet rate. The individual source rates were picked aggregate to 30% of the bottleneck link or 450 Kbps. A mixture of FTPs and HTTPs is then used to fill the link. Individual EF packet sources produce either all 160 byte packets or all 1500 byte packets.

Though we present the statistics of flows with one size of packet, all of the experiments used a mixture of short and long packet EF sources so the EF queues had a mix of both packet lengths.

We defined jitter as the absolute value of the difference between the arrival times of two adjacent packets minus their departure times, |(aj-dj) - (ai-di)|. For the target flow of each experiment, we record the median and 90th percentile values of jitter (expressed as % of the subscribed EF rate) in a table. The pdf version of this document contains graphs of the jitter percentiles.

Our experiments compared the jitter of WRR and PQ implementations of the EF PHB. We assessed the effect of different choices of WRR queue weight and number of queues on jitter. For WRR, we define the service-to-arrival rate ratio as the service rate of the EF queue (or the queue"s minimum share of the output link) times the output link bandwidth divided by the peak arrival rate of EF-marked packets at the queue. Results will not be stable if the WRR weight is chosen to exactly balance arrival and departure rates thus we used a minimum service-to-arrival ratio of 1.03. In our simulations this means that the EF queue gets at least 31% of the output links. In WRR simulations we kept the link full with other traffic as described above, splitting the non-EF-marked traffic among the non-EF queues. (It should be clear from the experiment description that we are attempting to induce worst-case jitter and do not expect these settings or traffic to represent a "normal" operating point.)

Our first set of experiments uses the minimal service-to-arrival ratio of 1.06 and we vary the number of individual microflows composing the EF aggregate from 2 to 36. We compare these to a PQ implementation with 24 flows. First, we examine a microflow at a subscribed rate of 56 Kbps sending 1500 byte packets, then one at the same rate but sending 160 byte packets. Table 1 shows the 50th and 90th percentile jitter in percent of a packet time at the subscribed rate. Figure 1 plots the 1500 byte flows and figure 2 the 160 byte flows. Note that a packet-time for a 1500 byte packet at 56 Kbps is 214 ms, for a 160 byte packet 23 ms. The jitter for the large packets rarely exceeds half a subscribed rate packet-time, though most jitters for the small packets are at least one subscribed rate packet-time. Keep in mind that the EF aggregate is a mixture of small and large packets in all cases so short packets can wait for long packets in the EF queue. PQ gives a very low jitter.

Table 1: Variation in jitter with number of EF flows: Service/arrival ratio of 1.06 and subscription rate of 56 Kbps (all values given as % of subscribed rate)

	1500 byte pack.		160 byte packet	
# EF flows	50th %	90th %	50th %	90th %
PQ (24)	1	5	17	43
2	11	47	96	513
4	12	35	100	278
8	10	25	96	126
24	18	47	96	143

Next we look at the effects of increasing the service-to-arrival ratio. This means that EF packets should remain enqueued for less time though the bandwidth available to the other queues remains the same. In this set of experiments the number of flows in the EF aggregate was fixed at eight and the total number of queues at five (four non-EF queues). Table 2 shows the results for 1500 and 160 byte flows. Figures 3 plots the 1500 byte results and figure 4 the 160 byte results. Performance gains leveled off at service-to-arrival ratios of 1.5. Note that the higher service-to-arrival ratios do not give the same performance as PQ, but now 90% of packets experience less than a subscribed packet-time of jitter even for the small packets.

Table 2: Variation in Jitter of EF flows: service/arrival ratio varies, 8 flow aggregate, 56 Kbps subscribed rate

WRR	1500 byte pack.		160 byte packet	
Ser/Arr	50th %	90th %	50th %	90th %
PQ	1	3	17	43
1.03	14	27	100	178
1.30	7	21	65	113
1.50	5	13	57	104
1.70	5	13	57	100
2.00	5	13	57	104
3.00	5	13	57	100

Increasing the number of queues at the output interfaces can lead to more variability in the service time for EF packets so we carried out an experiment varying the number of queues at each output port. We fixed the number of flows in the aggregate to eight and used the minimal 1.03 service-to-arrival ratio. Results are shown in figure 5 and table 3. Figure 5 includes PQ with 8 flows as a baseline.

Table 3: Variation in Jitter with Number of Queues at Output Interface: Service-to-arrival ratio is 1.03, 8 flow aggregate

# EF flows	1500 byte packet 50th %	90th %
PQ (8)	1	3
2	7	21
4	7	21
6	8	22
8	10	23

It appears that most jitter for WRR is low and can be reduced by a proper choice of the EF queue's WRR share of the output link with respect to its subscribed rate. As noted, WRR is a worst case while PQ is the best case. Other possibilities include WFQ or CBQ with a fixed rate limit for the EF queue but giving it priority over other queues. We expect the latter to have performance nearly identical with PQ though future simulations are needed to verify this. We have not yet systematically explored effects of hop count, EF allocations other than 30% of the link bandwidth, or more complex topologies. The information in this section is not part of the EF PHB definition but provided simply as background to guide implementers.

A.3.2 VLL service

We used simulation to see how well a VLL service built from the EF PHB behaved, that is, does it look like a 'leased line' at the subscribed rate. In the simulations of the last section, none of the EF packets were dropped in the network and the target rate was always achieved for those CBR sources. However, we wanted to see if VLL really looks like a 'wire' to a TCP using it. So we simulated long-lived FTPs using a VLL service. Table 4 gives the percentage of each link allocated to EF traffic (bandwidths are lower on the links with fewer EF microflows), the subscribed VLL rate, the average rate for the same type of sender-receiver pair connected by a full duplex dedicated link at the subscribed rate and the average of the VLL flows for each simulation (all sender-receiver pairs had the same value). Losses only occur when the input shaping buffer overflows but not in the network. The target rate is not achieved due to the well-known TCP behavior.

Table 4: Performance of FTPs using a VLL service

% link to EF	Average delivered rate (Kbps) Subscribed	Dedicated	VLL
20	100	90	90
40	150	143	143
60	225	213	215

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.