Nuevas técnicas de control y gestión de tráfico en Internet para proporcionar Calidad de Servicio extremo a extremo

Universidad Politécnica de Cartagena

Departamento de Tecnologías de la Información y las Comunicaciones

María Dolores Cano Baños

Director

Dr. D. José Fernando Cerdán Cartagena

2004

UMI Number: 3162783

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

UNIVERSIDAD POLITÉCNICA DE CARTAGENA
Comisión de Doctorado

AUTORIZACIÓN DEL DIRECTOR/A DE LA TESIS

D. JOSÉ FERNANDO CERDÁN CARTAGENA, Profesor Doctor del Área de INGENIERÍA TELEMÁTICA en el Departamento de TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES

A U T O R I Z A:

La presentación de la Tesis Doctoral titulada "Nuevas técnicas de control y gestión de tráfico en Internet para proporcionar Calidad de Servicio extremo a extremo", realizada por Dª. MARÍA DOLORES CANO BAÑOS, bajo mi dirección y supervisión, en el Departamento de TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES, y que presenta para la obtención del grado de Doctor por la Universidad Politécnica de Cartagena.

En Cartagena, a _____ de _____ de 2004

EL DIRECTORA/A DE TESIS

Fdo.: JOSÉ FERNANDO CERDÁN CARTAGENA

UNIVERSIDAD POLITÉCNICA DE CARTAGENA
Comisión de Doctorado

AUTORIZACIÓN DEL DEPARTAMENTO RESPONSABLE

Dª. BÁRBARA ÁLVAREZ TORRES, Directora del Departamento TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES

I N F O R M A:

Que la Tesis Doctoral titulada "Nuevas técnicas de control y gestión de tráfico en Internet para proporcionar Calidad de Servicio extremo a extremo", ha sido realizada por Dª. MARÍA DOLORES CANO BAÑOS, bajo la dirección y supervisión de D. JOSÉ FERNANDO CERDÁN CARTAGENA y que el Departamento ha dado su conformidad para que sea presentada ante la Comisión de Doctorado.

En Cartagena, a _____ de _____ de _____

LA DIRECTORA DEL DEPARTAMENTO

Fdo.: BÁRBARA ÁLVAREZ TORRES

En memoria de mi padre

A mi madre

Agradecimientos

Quiero expresar mi gratitud a todas las personas que me han acompañado a lo largo de estos años.

Me gustaría destacar muy especialmente la labor desempeñada por el Director de esta Tesis Doctoral, Fernando Cerdán Cartagena. He sido afortunada por contar con su orientación y consejos. Para mi se ha convertido en un referente como investigador. Gracias Fernando por guiarme durante estos años.

Quisiera agradecer también su ayuda y colaboración a todos los compañeros del Área de Ingeniería Telemática y del Departamento de Tecnologías de la Información y las Comunicaciones. A todos gracias por vuestro apoyo.

De todo corazón, gracias a los buenos amigos por los momentos que hemos pasado. En especial y con mucho cariño a Cristina, Ginés, Juan y Pablo. Ojalá que las tradiciones del *core* perduren muchos años.

Por encima de todo, gracias a mi familia, por el cariño, por el ánimo y por la comprensión. Sin vosotros, esto habría no habría sido posible. Y gracias a ti, Javi, por el cariño y apoyo en todo momento.

Índice de contenidos

ÍNDICE DE CONTENIDOS I

ÍNDICE DE FIGURAS IV

ÍNDICE DE TABLAS IX

OBJETIVOS Y CONTRIBUCIONES XI

CAPÍTULO 1 INTRODUCCIÓN 1

1.1. Introducción 1
1.1.1. Conceptos de DiffServ 3
1.1.1.1. Dominio DiffServ 3
1.1.1.2. Acondicionador de tráfico 4
1.1.1.3. Comportamiento por saltos 6
1.1.2. Gestión y control activo de la congestión 8

1.2. Estado de la técnica 12

1.3. Herramienta de simulación 15

CAPÍTULO 2 PROPUESTAS DE ACONDICIONADORES DE TRÁFICO EN DIFFSERV 17

2.1. Introducción 17

2.2. Configuración de las simulaciones 18

2.3. Acondicionadores de tráfico clásicos 19
2.3.1. Algoritmo del cubo licuante o *Leaky Bucket* 19
2.3.1.1 Tasa de velocidad de paquetes *in* 20
2.3.1.2 Distribución del ancho de banda excedente 21
2.3.2. Algoritmo de ventana deslizante temporal o *Time Sliding Window* 24
2.3.2.1 Dependencia entre los valores de β y *winlength* 24
2.3.2.2 Tasa de velocidad de paquetes *in* 28
2.3.2.3 Distribución del ancho de banda excedente 30

2.4. Algoritmo basado en contadores *Counters Based* 30
2.4.1. Tasa de velocidad de paquetes *in* 31
2.4.2. Distribución del ancho de banda excedente 33

2.5. Conclusiones 35

CAPÍTULO 3 *COUNTERS BASED DUAL QUEUING*: PROPUESTA ALTERNATIVA PARA LA DIFERENCIACIÓN DE SERVICIOS 37

3.1. Introducción 37

3.2. El esquema de gestión de doble cola CBDQ 38

3.3. Escenarios de simulación 39

3.4. Resultados 40
3.4.1. Distribución del ancho de banda no contratado 40
3.4.2. Garantías de contrato 41
3.4.3. Efecto de los tamaños de paquete 43

3.5. Conclusiones 44

CAPÍTULO 4 *COUNTERS BASED MODIFIED*: NUEVO MÉTODO DE ACONDICIONADO DE TRÁFICO BASADO EN EL DESCARTE ANTICIPADO DE PAQUETES 47

4.1. Introducción 47

4.2. El acondicionador de tráfico CBM 47
4.2.1. Cómo establecer los límites *max* y *min* 50
4.2.2. Estimación del tiempo de ida y vuelta 50
4.2.3. Probabilidad de descarte de los paquetes *out* 50
4.2.4. El control de la generación de paquetes *out* 51

4.3. Topologías y escenarios de simulación 55

4.4. Resultados 56
4.4.1. Resultados en la topología de un solo nodo 56
4.4.1.1. Simulaciones con tráfico asegurado 57
4.4.1.2. Simulaciones con tráfico asegurado y *best-effort* 59
4.4.2. Topología de tres nodos 63
4.4.2.1. Simulaciones con tráfico asegurado 63
4.4.2.2. Simulaciones con tráfico asegurado y un nodo sin diferenciación de servicios 64
4.4.2.3. Simulaciones con tráfico asegurado y tráfico *best-effort* 67
4.4.2.4. Reconfiguración de los parámetros de RIO 70

4.5. Conclusiones 72

CAPÍTULO 5 NUEVAS TÉCNICAS PARA EL REPARTO PROPORCIONAL DE ANCHO DE BANDA EN EXCESO 75

5.1. Introducción 75

5.2. La función policía PETER 76

5.3. Descripción de los escenarios de simulación 80

5.4. Resultados 81
5.4.1. Caso 1: igualdad en los contratos y en los tiempos de ida y vuelta 81
5.4.2. Caso 2: diversidad de contratos e igualdad en los tiempos de ida y vuelta 83
5.4.3. Caso 3: igualdad de contratos y variación en los tiempos de ida y vuelta..... 84
5.4.4. Caso 4: diversidad de contratos y de tiempos de ida y vuelta..... 85
5.4.5. Caso 5: incremento en el número de fuentes 85
5.4.6. Diferencias entre PETER-RIO y PETER-DQ 87
5.4.6.1. Caso 1: igualdad de contratos y retardos 87
5.4.6.2. Caso 2: variación de contratos e igualdad de retardos..... 88
5.4.6.3. Caso 3: igualdad de contratos y variación de retardos..... 89
5.4.6.4. Caso 4: variedad de contratos y de retardos..... 90
5.4.6.5. Caso 5: incremento en el número de fuentes 91

5.5. Implementación 92

5.6. Conclusiones..... 93

CONCLUSIONES 95

ANEXO A ESTUDIO EXPERIMENTAL DE LAS SOLUCIONES DISPONIBLES EN ENRUTADORES COMERCIALES PARA IMPLEMENTAR DIFFSERV 99

A.1. Introducción..... 99

A.2. Técnicas de gestión de colas..... 99

A.3. Topología Experimental..... 100

A.4. Resultados 104
A.4.1. Caso 1: Una única cola FIFO con WRED 104
A.4.2 Caso 2: CBWFQ sin RED 106
A.4.3 CBWFQ con WRED 109

A.5. Conclusiones..... 111

GLOSARIO DE TÉRMINOS 113

REFERENCIAS BIBLIOGRÁFICAS 117

Índice de figuras

Fig. 1.1 Dominio de Servicios Diferenciados 3
Fig. 1.2 Acondicionador de tráfico y sus componentes lógicos 5
Fig. 1.3 Localización de los acondicionadores de tráfico 5
Fig. 1.4 Cabecera del protocolo de Internet IP 7
Fig. 1.5 Cola por prioridad (PQ) 9
Fig. 1.6 Asignación de cola justa (FQ) 10
Fig. 1.7 Asignación justa de cola por pesos (WFQ) 10
Fig. 1.8 Ejemplo RED. Perfil de descarte 11
Fig. 1.9 Ejemplo RIO. Perfil de descarte 12

Fig. 2.1 Topología de red para las simulaciones 18
Fig. 2.2 Pseudo-código del algoritmo *Leaky Bucket*. 19
Fig. 2.3 Diagrama de flujo del algoritmo *Leaky Bucket* 20
Fig. 2.4 τ vs. RTT 21
Fig. 2.5 % del ancho de banda excedente que obtiene cada fuente en función del RTT con a) ζ=0 y b) ζ=10 22
Fig. 2.6 Implementación de TSW 24
Fig. 2.7 β vs. *winlength* 25
Fig. 2.8 β vs. *winlength* 26
Fig. 2.9 Guía de configuración de valores de β 26
Fig. 2.10 Ejemplo de una configuración errónea 27
Fig. 2.11 Ejemplo de una configuración correcta 27
Fig. 2.12 Caudal de paquetes *in* de la fuente s_1 29
Fig. 2.13 Caudal de paquetes *in* de la fuente s_6 29
Fig. 2.14 Pseudo-código del algoritmo *Counters Based* 31
Fig. 2.15 Diagrama de flujo del algoritmo *Counters Based* 31
Fig. 2.16 Caudal alcanzado de paquetes *in* por la fuente s_1 (contrato de 1 Mbps) 32
Fig. 2.17 Caudal alcanzado de paquetes *in* por la fuente s_4 (contrato de 3 Mbps) 32
Fig. 2.18 Caudal alcanzado de paquetes *in* por la fuente s_6 (contrato de 4 Mbps) 33
Fig. 2.19 Distribución del ancho de banda en exceso con el algoritmo *Counters Based* 34

Fig. 3.1 Esquema de gestión del cola de doble cola CBDQ 38
Fig. 3.2 Topología para las simulaciones 39
Fig. 3.3 Caudal de paquetes *in* en el escenario A. 41
Fig. 3.4 Caudal de paquetes *in* en el escenario B. 42
Fig. 3.5 Caudal de paquetes *in* en el escenario C. 42
Fig. 3.6 Caudal de paquetes *in* en el escenario D. 43
Fig. 3.7 Caudal de paquetes *in* en el escenario E. 43

Fig. 4.1 Paquetes *out* generados por la fuente s_1 entre paquetes *in* consecutivos 48
Fig. 4.2 Paquetes *out* generados por la fuente s_7 entre paquetes *in* consecutivos 49
Fig. 4.3 Descarte de paquetes *out* en el acondicionador de tráfico CBM 49
Fig. 4.4 Ecuaciones examinadas para encontrar la curva de probabilidad de descarte de paquetes *out* en el acondicionador de tráfico CBM 51
Fig. 4.5 Pseudo-código simplificado del acondicionador de tráfico CBM 52

Fig. 4.6 Paquetes *out* entre paquetes *in* consecutivos en la fuente s_1 *sin* descarte de paquetes *out* en el acondicionador de tráfico CB 53
Fig. 4.7 Paquetes *out* entre paquetes *in* consecutivos en la fuente s_1 *con* descarte de paquetes *out* en el acondicionador de tráfico CBM 53
Fig. 4.8 Paquetes *out* entre paquetes *in* consecutivos en la fuente s_7 *sin* descarte de paquetes *out* en el acondicionador de tráfico CB 54
Fig. 4.9 Paquetes *out* entre paquetes *in* consecutivos en la fuente s_7 *con* descarte de paquetes *out* en el acondicionador de tráfico CBM 54
Fig. 4.10 Topología general de un único cuello de botella 55
Fig. 4.11 Topología general de tres nodos 55
Fig. 4.12 Índices de justicia alcanzados en los escenarios A, B, C, D y E dentro de la topología de un único cuello de botella con los acondicionadores de tráfico *Counters-Based Modified* (CBM), *Counters-Based* (CB), *Time Sliding Window* (TSW) y *Leaky Bucket* (LB) 58
Fig. 4.13 Caudal de paquetes *in* en el escenario B en la topología de un solo cuello de botella. 58
Fig. 4.14 Caudal de paquetes *in* en el escenario D en la topología de un solo cuello de botella 58
Fig. 4.15 *Goodput* (bps) de las fuentes *best-effort* (s_4 a s_7) en el escenario A 60
Fig. 4.16 *Goodput* (bps) de las fuentes *best-effort* (s_4 a s_7) en el escenario B 60
Fig. 4.17 *Goodput* (bps) de las fuentes *best-effort* (s_4 a s_7) en el escenario C 61
Fig. 4.18 *Goodput* (bps) de las fuentes *best-effort* (s_4 a s_7) en el escenario D 61
Fig. 4.19 *Goodput* (bps) de las fuentes *best-effort* (s_4 a s_7) en el escenario E 61
Fig. 4.20 Índice de justicia en los escenarios A, B, C, D y E donde coexisten fuentes aseguradas y *best-effort* 62
Fig. 4.21 Caudal de paquetes *in* en el escenario D con fuentes aseguradas y *best-effort* 62
Fig. 4.22 Topología de tres nodos en el caso 1 63
Fig. 4.23 Topología de tres nodos en el caso 2 con el nodo E_2 sin diferenciación de servicios (caso 2a) 65
Fig. 4.24 Caudal de paquetes *in* obtenido en el caso 2a utilizando CBM en el escenario C. 65
Fig. 4.25 Caudal de paquetes *in* obtenido en el caso 2a utilizando CBM en el escenario D 66
Fig. 4.26 Topología de tres nodos RIO (caso 3a) 67
Fig. 4.27 Topología de tres nodos en el caso 3b (RED en nodo E_2) empleando CBM 69
Fig. 4.28 Cuadro comparativo de los índices de justicia de todos los escenarios para cada uno de los casos de estudio analizados en la topología de tres nodos 70
Fig. 4.29 Topología de tres nodos en la que E_1 sólo recibe tráfico asegurado, E_2 sólo gestiona tráfico *best-effort* y los parámetros de RIO de E_3 han sido reconfigurados 70

Fig. 5.1 Funcionamiento general de PETER 78
Fig. 5.2 Topología de ejemplo. 79
Fig. 5.3 Pseudo-código del algoritmo PETER aplicado en los acondicionadores de tráfico 79
Fig. 5.4 Topología para el estudio mediante simulación 81
Fig. 5.5 *Goodputs* en el caso 1 con PETER y una carga de red del 60% 82
Fig. 5.6 Índice de justicia vs. carga de red en el caso 1. Comparativa entre PETER y TSW 82
Fig. 5.7 Índice de justicia vs. carga de red en el caso 2. Comparativa entre PETER y TSW 83
Fig. 5.8 Índice de justicia vs. carga de red en el caso 3. Comparativa entre PETER y TSW 84
Fig. 5.9 Índice de justicia vs. carga de red en el caso 4. Comparativa entre PETER y TSW 85
Fig. 5.10 Índice de justicia vs. carga de red en el caso 5. Comparativa entre PETER y TSW ... 86
Fig. 5.11 Un nodo DQ 87
Fig. 5.12 Índice de justicia vs. carga de red en el caso 1 utilizando DQ. Comparativa entre PETER y TSW 88
Fig. 5.13 Índice de justicia vs. carga de red en el caso 2 utilizando DQ. Comparativa entre PETER y TSW 89
Fig. 5.14 Índice de justicia vs. carga de red en el caso 3 utilizando DQ. Comparativa entre PETER y TSW 90

Fig. 5.15 Índice de justicia vs. carga de red en el caso 4 utilizando DQ. Comparativa entre PETER y TSW 91
Fig. 5.16 Índice de justicia vs. carga de red en el caso 5 utilizando DQ. Comparativa entre PETER y TSW 92

Fig. A.1. Topología experimental 100
Fig. A.2 Cola FIFO con WRED para todo el tráfico AF 101
Fig. A.3 CBWFQ sin WRED para el tráfico AF 101
Fig. A.4 CBWFQ con WRED para todo el tráfico AF 102
Fig. A.5 Caudal de paquetes *in* en el escenario A. FIFO con WRED 104
Fig. A.6 Ancho de banda en exceso en el escenario A. FIFO con WRED 105
Fig. A.7 Caudal de paquetes *in* en el escenario B. FIFO con WRED 105
Fig. A.8 Ancho de banda en exceso en el escenario B. FIFO con WRED 105
Fig. A.9 Caudal de paquetes *in* en el escenario A. CBWFQ sin WRED 106
Fig. A.10 Ancho de banda en exceso en el escenario A. CBWFQ sin RED 106
Fig. A.11 Caudal de paquetes *in* en el escenario B. CBWFQ sin RED 107
Fig. A.12 Ancho de banda en exceso en el escenario B. CBWFQ sin RED 107
Fig. A.13 Caudal de paquetes *in* para una carga de red del 62,5% en el escenario C. CBWFQ sin RED 108
Fig. A.14 Ancho de banda en exceso que obtiene cada red LAN en el escenario C. CBWFQ sin RED 108
Fig. A.15 Caudal de paquetes *in* en el escenario A. CBWFQ con WRED 109
Fig. A.16 Ancho de banda en exceso en el escenario A. CBWFQ con WRED 110
Fig. A.17 Caudal de paquetes *in* en el escenario B. CBWFQ con WRED.. 110
Fig. A.18 Ancho de banda en exceso en el escenario B. CBWFQ con WRED. 110
Fig. A.19 Ancho de banda en exceso en el escenario C. CBWFQ con WRED. 111

Índice de tablas

Tabla 1.1 Valores recomendados del campo DSCP para el servicio asegurado AF y sus correspondientes valores de precedencia de descarte..... 7

Tabla 2.1 Simulaciones con *Leaky Bucket* con el mismo RTT para cada fuente. 23
Tabla 2.2 Simulaciones con *Leaky Bucket* con distintos RTT cada fuente. 23
Tabla 2.3 TSW con los valores β según la Fig. 2.9 y *winlength* = 200 ms 26
Tabla 2.4 Necesidad del factor de corrección de β por la influencia del RTT. 28
Tabla 2.5 Comportamiento de TSW con el mismo RTT todas las fuentes 29
Tabla 2.6 Comportamiento de TSW cuando cada fuente presenta un RTT diferente 30
Tabla 2.7 Resultados de simulaciones con el algoritmo *Counters Based* con RTT similares para cada fuente 34
Tabla 2.8 Resultados de simulaciones con el algoritmo *Counters Based* con RTT distintos para cada fuente 34

Tabla 3.1 Tabla comparativa del índice de justicia 40
Tabla 3.2 Índice de justicia aplicando CBDQ con diferentes tamaños de paquete (no simultáneos) 44
Tabla 3.3 Tabla comparativa del índice de justicia con diferentes esquemas de acondicionado y gestión de cola con tamaños de paquete variables (simultáneos) 44

Tabla 4.1 Valores de los umbrales *max* y *min* que se utilizan para CBM en cada escenario sobre la topología de la Fig. 4.10..... 57
Tabla 4.2 Contratos, tiempos de ida y vuelta y límites *max-min* de las fuentes TCP Reno para los casos 1 y 2 (los datos se corresponden con las fuentes s_0 a s_7 respectivamente)... 64
Tabla 4.3 Caudal alcanzado de paquetes *in* (Mbps) e índices de justicia en los cinco escenarios del caso 1 64
Tabla 4.4 Caudal de paquetes *in* (Mbps) e índices de justicia en los cinco escenarios del caso 2a (RED en el nodo E_2) 66
Tabla 4.5 Caudal de paquetes *in* (Mbps) e índices de justicia en los cinco escenarios del caso 2b (RED en el nodo E_1)..... 67
Tabla 4.6 Contratos, tiempos de ida y vuelta y límites *max-min*, si aplica, de las fuentes TCP Reno para el caso 3 (los datos se corresponden con las fuentes s_0 a s_{11} respectivamente) 68
Tabla 4.7 Caudal de paquetes *in* (Mbps) e índices de justicia en los cinco escenarios del caso 3a (todos los nodos son RIO) 68
Tabla 4.8 Caudal de paquetes *in* (Mbps) e índices de justicia en los cinco escenarios del caso 3b (RED en el nodo E_2)..... 69
Tabla 4.9 Contratos, tiempos de ida y vuelta y límites max-min, si aplica, de las fuentes TCP Reno para la topología de la Fig. 4.29 (los datos se corresponden con las fuentes s0 a s11 respectivamente) 71
Tabla 4.10 Caudal de paquetes *in* (Mbps) e índices de justicia en los cinco escenarios correspondiente a la topología de la Fig. 4.29 con actualización de parámetros RIO en el nodo E_3..... 71

Tabla 4.11 Caudal de paquetes *in* (Mbps) e índices de justicia en los cinco escenarios correspondiente a la topología de la Fig. 4.29 sin actualización de parámetros RIO en el nodo E_3 71

Tabla 5.1 *Goodputs* obtenidos en el caso 1 con ocho fuentes TCP Reno con contratos de 2,5 Mbps (carga de la red del 60%). El tiempo de ida y vuelta queda fijado a 50 ms para todas las fuentes 82
Tabla 5.2 *Goodputs* obtenidos en el caso 2 con ocho fuentes TCP Reno con contratos de 1-1-2-2-3-3-4 y 4 Mbps (carga de la red del 60%). El tiempo de ida y vuelta queda fijado a 50 ms para todas las fuentes 83
Tabla 5.3 *Goodputs* obtenidos en el caso 3 con ocho fuentes TCP Reno con contratos de 2,5 Mbps (carga de la red del 60%). El tiempo de ida y vuelta queda fijado en 10-20-30-40-50-60-70 y 80 ms para las conexiones s_1 a s_8 respectivamente 84
Tabla 5.4 *Goodputs* obtenidos en el caso 4 con ocho fuentes TCP Reno con contratos de 4-4-3-3-2-2-1 y 1 Mbps (carga de red del 60%). El tiempo de ida y vuelta se establece a 10-20-30-40-50-60-70 y 80 ms para las conexiones s_1 a s_8 respectivamente 85
Tabla 5.5 *Goodputs* obtenidos en el caso 5 con dieciséis fuentes TCP Reno con contratos de 1 Mbps (s_1 a s_4) y 1,32 Mbps (s_5 a s_{16}) para una carga de red del 60%. El tiempo de ida y vuelta se establece a 50 ms para todas las conexiones 86
Tabla 5.6 *Goodputs* obtenidos en el caso 1 utilizando DQ. Las ocho fuentes TCP Reno tienen contratos de 2,5 Mbps (carga de la red del 60%). El tiempo de ida y vuelta queda fijado a 50 ms para todas las fuentes 87
Tabla 5.7 *Goodputs* obtenidos en el caso 1 utilizando DQ. Las ocho fuentes TCP Reno tienen contratos de 1-1-2-2-3-3-4 y 4 Mbps (carga de la red del 60%). El tiempo de ida y vuelta queda fijado a 50 ms para todas las fuentes 88
Tabla 5.8 *Goodputs* obtenidos en el caso 3 utilizando DQ. Las ocho fuentes TCP Reno tienen contratos de 2,5 Mbps (carga de la red del 60%). El tiempo de ida y vuelta queda fijado a 10-20-30-40-50-60-70 y 80 ms para las fuentes s_1 a s_8 89
Tabla 5.9 *Goodputs* obtenidos en el caso 4 utilizando DQ. Las ocho fuentes TCP Reno tienen contratos de 4-4-3-3-2-2-1 y 1 Mbps (carga de la red del 60%). El tiempo de ida y vuelta queda fijado a 10-20-30-40-50-60-70 y 80 ms para las fuentes s_1 a s_8 90
Tabla 5.10 *Goodputs* obtenidos en el caso 5 utilizando DQ. Las dieciséis fuentes TCP Reno tienen contratos de 1 Mbps (s_1 a s_4) y 1,32 Mbps (s_5 a s_{16}) (carga de la red del 60%). El tiempo de ida y vuelta queda fijado a 50 ms 91

Tabla A.1 Contratos para el escenario A 103
Tabla A.2 Contratos para el escenario B 103
Tabla A.3 Contratos para el escenario C 103

Objetivos y contribuciones

La principal motivación que nos ha llevado a realizar este trabajo ha sido la constatación de que en la actualidad, la mayoría de redes de comunicación requieren Calidad de Servicio. En el momento de comenzar esta tesis, parecía claro que la principal apuesta para la consecución de una Internet con Calidad de Servicio era la arquitectura de Servicios Diferenciados. No obstante, la tendencia actual parece ser mixta. Dentro de la arquitectura de Servicios Diferenciados son varios los niveles de servicio que se pueden ofrecer al usuario. Desde nuestro punto de vista, uno de los servicios más interesantes es el denominado servicio asegurado. Según indica la RFC 2597, el servicio asegurado pretende ofrecer al usuario final la garantía de un ancho de banda mínimo, normalmente el contrato establecido con el proveedor de servicios, y además, si existe ancho de banda disponible no contratado distribuirlo de modo justo entre los distintos usuarios. Se entiende por una distribución justa el reparto o bien equitativo entre los usuarios o bien proporcional al contrato de cada uno de ellos.

Los Servicios Diferenciados basan su funcionamiento en la separación de funciones. Los nodos frontera de los dominios llevarán a cabo tareas de acondicionado de tráfico, desempeñando un papel clave en las prestaciones finales [CANO04a]. Mientras que todos los nodos, frontera e interiores, realizarán labores de planificación y gestión de la congestión. Así, el primer paso de este trabajo ha sido conocer el estado de la técnica respecto a métodos de acondicionado de tráfico y métodos de planificación y gestión de colas utilizados para ofrecer un servicio asegurado dentro de la arquitectura de Servicios Diferenciados. El resultado de este estado de la técnica se ha ido incorporando en los diferentes trabajos que se han publicado derivados de los resultados de esta tesis.

El siguiente paso ha sido realizar un estudio comparativo de las prestaciones de algunos algoritmos de acondicionado existentes, a los que podríamos designar como clásicos por ser referencia clave en la literatura especializada. En [CANO01a] se ha realizado un análisis de las prestaciones de estos clásicos junto con las pautas a seguir para su correcta configuración. De los resultados de simulación obtenidos en [CANO01a] se resuelve que estos algoritmos carecen de la eficiencia necesaria para ofrecer un servicio asegurado con todas las garantías. Efectivamente, es necesario un algoritmo de acondicionado que no sea tan dependiente de los parámetros de configuración y de las características de la red. Intentando solucionar este problema, hemos presentado un nuevo mecanismo de acondicionado basado en contadores denominado *Counters Based* [CANO01a]. Este mismo trabajo incluye el correspondiente estudio de prestaciones mediante simulación para fuentes de tráfico TCP Reno. De los resultados obtenidos se ha de destacar que mejora notablemente a sus predecesores pero sigue presentando deficiencias en cuanto a la distribución equitativa del ancho de banda en exceso.

Sin olvidar la importancia de los mecanismos de planificación y gestión de las colas que todos los nodos de un dominio de Servicios Diferenciados deben incorporar, hemos presentado en [CANO02] [CANO03b] un nuevo mecanismo para la implementación del comportamiento por saltos del servicio asegurado denominado *Counters Based Dual Queuing*. Este esquema se basa en separar los paquetes pertenecientes al servicio asegurado pero con diferentes niveles de precedencia en colas distintas, evitando así cualquier tipo de interferencia entre ellos. Como algoritmo de marcado se mantiene la propuesta *Counters Based*. El estudio de prestaciones se ha realizado mediante simulación para fuentes de tráfico TCP Reno y se ha publicado en el mismo trabajo. De los resultados se puede extraer que se consiguen garantizar los contratos en diferentes escenarios de características heterogéneas y además se reparte el ancho de banda en exceso de modo equitativo.

Con el objetivo de encontrar un acondicionador que, independientemente del esquema de gestión de las colas, permitiera completar las expectativas del servicio asegurado, hemos presentado una modificación al acondicionador *Counters*-Based. La idea básica es poder proporcionar justicia en el reparto del ancho de banda en exceso. Así nace el acondicionador de tráfico *Counters Based Modified*. Este acondicionador se basa en el descarte probabilístico de paquetes con menor nivel de precedencia. Hemos realizado un estudio de prestaciones mediante simulación para fuentes de tráfico TCP Reno. Las simulaciones se han realizado en distintos escenarios con tiempos de ida y vuelta y contratos variables, además de estudiar la interacción entre tráfico *best-effort* y tráfico asegurado. De los resultados se observa que es posible garantizar un servicio asegurado que cumpla con sus dos objetivos. Este trabajo se publicó en [CANO02a].

Es de vital importancia demostrar la aplicabilidad de los acondicionadores de tráfico en contextos reales y su posible uso en entornos como Internet. Con este fin, hemos analizado el comportamiento del acondicionador de tráfico *Counters Based Modified* en un entorno más realista [CANO03] [CANO03a] [CANO04] [CANO04b]. Las simulaciones se realizaron en escenarios heterogéneos en los que además de variabilidad de tiempos de ida y vuelta y contratos se ha contemplado la posibilidad de que alguno de los nodos del dominio no pueda ofrecer diferenciación de servicios, así como la mezcla de tráficos con el fin de estudiar la robustez del sistema. Los resultados muestran que con este acondicionador de tráfico se mantienen los contratos con gran exactitud y se consigue distribuir el ancho de banda no contratado de modo equitativo ente las fuentes que componen el agregado.

Debido al desacuerdo existente entre los autores de literatura relacionada sobre el método en el que el ancho de banda no contratado debe distribuirse entre las fuentes de un agregado, hemos buscado una solución que permitiese repartir este ancho de banda en exceso de modo proporcional al contrato de cada fuente. Con este nuevo acondicionador de tráfico al que hemos denominado *Proporcional Excess Traffic conditionER* (PETER), las fuentes de tráfico se adaptan a la situación real de la red, disminuyendo su tasa de emisión si fuese necesario. Los buenos resultados obtenidos en el estudio mediante simulación demuestran que es posible conseguir los objetivos del servicio asegurado con una distribución proporcional del ancho de banda en exceso [CANO04c].

Conocida la tendencia actual a emplear la arquitectura de Servicios Diferenciados para garantizar Calidad de Servicio a los usuarios de redes IP, nos hemos propuesto conocer de qué mecanismos dispone un operador o proveedor de servicios hoy en día para poder implementar esta arquitectura en equipos comerciales. Así, de forma paralela al desarrollo de nuevos acondicionadores de tráfico y algoritmos de planificación de colas para el servicio asegurado, hemos realizado un estudio experimental de prestaciones del único método de acondicionado disponible en los equipos comerciales, conocido como *token bucket*, y de tres métodos de implementación del comportamiento por saltos del servicio asegurado [CANO04d]. La principal contribución de este trabajo es la constatación de que las herramientas disponibles en la actualidad no proporcionan de un modo escalable un servicio asegurado con todas las garantías a los usuarios finales, siendo imprescindible incorporar nuevas propuestas como las planteadas en esta tesis.

[CANO01a] Maria-Dolores Cano, Fernando Cerdan, Joan Garcia-Haro, Josemaria Malgosa-Sanahuja, “Performance Evaluation of Traffic Conditioner Mechanisms for the Internet Assured Service”, *Proceedings* de SPIE Quality of Service over Next-Generation Data Networks, Vol. 4524, pp. 182-193, Denver, USA, Agosto 2001.

[CANO02] Maria-Dolores Cano, Fernando Cerdan, Joan Garcia-Haro, Josemaria Malgosa-Sanahuja, “A New Proposal for Assuring Services in Internet”, *Proceedings* de Internet Computing IC’02, CSREA Press, Vol. II, pp. 379-384, Las Vegas, USA, Junio 2002.

[CANO02a] Maria-Dolores Cano, Fernando Cerdan, Joan Garcia-Haro, Josemaria Malgosa-Sanahuja, "Counters-Based Modified Traffic Conditioner", *Lecture Notes in Computer Science* (QofIS 2002), Vol. 2511, pp. 57-67, Springer-Verlag, 2002.

[CANO03] Maria-Dolores Cano, Fernando Cerdan, Joan Garcia-Haro, Josemaria Malgosa-Sanahuja, "Performance Evaluation of the Counters-Based Modified Traffic Conditioner in a DiffServ Network", *Proceedings* de International Symposium on Computers and Communications ISCC'03, Vol. I, pp. 305-311, Antalya, Turquia, Julio 2003.

[CANO03b] Maria-Dolores Cano, Juan Jose Alcaraz, Pablo Lopez-Matencio, Fernando Cerdan, "CBDQ: Garantía de Calidad de Servicio en Internet", *Proceeding*s de XIII Jornadas Telecom. I+D 2003, Madrid, Noviembre 2003.

[CANO03a] Maria-Dolores Cano, Fernando Cerdan, Joan Garcia-Haro, Josemaria Malgosa-Sanahuja, "Análisis de las Prestaciones del Acondicionador de Tráfico Counters-Based Modified en un Dominio DiffServ", *Proceedings* de Jornadas de Ingeniería Telemática JITEL'03, Las Palmas de Gran Canaria, España, Septiembre 2003.

[CANO04] Maria-Dolores Cano y Fernando Cerdan "End-to-End TCP Performance of the Couple CBM Traffic Conditioner and RIO Buffer Management in a Three-Node Topology", *Proceedings* de International Conference on Communications and Computer Networks CCN'04, Cambridge, USA, Noviembre 2004.

[CANO04a] Maria-Dolores Cano, Pablo Lopez-Matencio, Juan Jose Alcaraz, Fernando Cerdan, "El papel de los acondicionadores de tráfico para ofrecer Calidad de Servicio extremo a extremo", Revista II Teleco-Forum, Universidad Politécnica de Cartagena, pp. 80-82, Cartagena, España, Abril 2004.

[CANO04b] Maria-Dolores Cano, Fernando Cerdan, Joan Garcia-Haro "CBM: Guarantees for a Complete Internet Assured Service", en proceso de revision en *Journal of Communications and Networks*.

[CANO04c] Maria-Dolores Cano, Fernando Cerdan, Joan Garcia-Haro, "Traffic Control for an Effective Internet Service Differentiation", en proceso de revisión en *International Journal of Electronics and Communications*.

[CANO04d] Maria-Dolores Cano, Fernando Cerdan "An Experimental Study of Bandwidth Assurance in a DiffServ Network", en proceso de revisión en International Conference on Internet and Multimedia Systems and Applications EuroIMSA'05.

Capítulo 1
Los Servicios Diferenciados

1.1. Introducción

Hace algunos años, el uso de Internet estaba limitado a aplicaciones como el correo electrónico o la transferencia de archivos. En estas circunstancias, el modelo convencional de servicios *best-effort*, en el que todos los paquetes compiten de igual manera por los recursos de la red era suficientemente bueno como para proporcionar un servicio aceptable al usuario final. Sin embargo, se ha producido un extraordinario crecimiento de Internet y se han desarrollado infinidad de nuevas aplicaciones de telecomunicación, con diversas necesidades de retardo, varianza del retardo (conocido en inglés como *jitter*), ancho de banda, pérdidas de paquetes y disponibilidad. El resultado es un incremento exponencial en la utilización de los recursos. Claramente, este hecho puede provocar situaciones de congestión en las redes de comunicación. Para evitar este efecto indeseable se han concebido diferentes propuestas con el objetivo de proporcionar a cada aplicación exactamente aquellos recursos de red que necesite. En otras palabras, capaces de ofrecer calidad de servicio (*Quality of Service*, QoS).

La respuesta más sencilla para solventar esta demanda sería añadir más recursos, más capacidad, a los enlaces. Esta técnica se conoce como sobredimensionado de redes. Es decir, se emplean redes con una capacidad muy superior a los requisitos de las redes actuales, lo que representa una alternativa a la QoS. No obstante, esta forma de actuar tiene algunos problemas adheridos que justifican la investigación de mecanismos más complejos capaces de proporcionar QoS. En primer lugar la tendencia actual es que las redes IP sean cada vez más heterogéneas, abarcando desde troncales (*backbone*) de fibra óptica hasta tecnologías inalámbricas (*wireless*) o cableadas en los puntos finales. Esta tendencia hace de Internet una red ubicua y multiservicio, donde diferentes proveedores de servicios (*Internet Service Provider*, ISP), operadores y distribuidores de contenidos compiten e interactúan ofreciendo nuevas aplicaciones y servicios. La presencia de diferentes *actores* en este escenario hace difícil encontrar una estrategia de ingeniería de tráfico para aumentar la capacidad de los enlaces. En segundo lugar, el coste y el desaprovechamiento de la tecnología empleada, además de que el rápido aumento del volumen de tráfico de la red podría dejar desbordados los sistemas sobredimensionados en muy poco tiempo, hacen del sobredimensionado una solución que por sí sola no se puede considerar efectiva. En consecuencia, existen suficientes razones de todo tipo (comerciales, etc.) para proporcionar QoS en las redes IP [OPER01].

Una primera aproximación al soporte de QoS en redes IP fue la arquitectura de Servicios Integrados (*Integrated Services*, IntServ) [RFC1633]. Según los planteamientos de esta arquitectura, el ancho de banda de la red se debe gestionar de manera implícita. Lo que conlleva el uso de mecanismos de reserva de recursos y control de acceso. El punto final de una aplicación deberá solicitar ser admitido en la red informando de sus requisitos de QoS empleando algún tipo de señalización como el protocolo de reserva de recursos RSVP (*Resource Reservation Protocol*) [RFC2205] definido por el IETF (*Internet Engineering Task Force*). De este modo, todos los nodos de la red, tales como dispositivos de enrutado o enrutadores (*routers*), deben ser informados de la clase de servicio que se solicita con sus correspondientes parámetros. Si hay recursos disponibles para atender las necesidades de esa

aplicación, la conexión será aceptada y se reservan los recursos hasta el cierre de la misma. De lo contrario, la conexión se rechaza. La arquitectura IntServ es capaz de garantizar estrictamente las necesidades de QoS. No obstante, esta arquitectura no se ha implantado en gran medida por considerarse demasiado compleja y por la falta de escalabilidad que presenta [SEDD98].

En este contexto, la necesidad de encontrar una solución escalable capaz de hacer frente a la necesidad de QoS de las redes actuales llevó a la aparición de los Servicios Diferenciados (*Differentiated Services*, DiffServ) [RFC2475] [RFC3260]. La arquitectura DiffServ intenta crear un esquema simple que ofrece una variedad de niveles de QoS, donde la complejidad se traslada a la frontera de la red. Entendemos por frontera, aquellos nodos que reciben o envían directamente el tráfico de o a los usuarios finales. DiffServ intenta a su vez, que los mecanismos empleados en el interior de la red sean tan sencillos como sea posible.

La implementación de DiffServ se basa en el uso del byte DSCP (*DiffServ Code Point*) de la cabecera IP [RFC2474] [RFC2475] [RFC3260]. El DSCP permite clasificar los paquetes y definir qué tipo de mecanismo se les ha de aplicar en los nodos interiores. El tráfico procedente de distintos flujos con requisitos similares de QoS se marca con el mismo DSCP, y así los flujos se agregan a una cola común o reciben el mismo trato. De este modo, son los propios paquetes los que llevan información sobre el tratamiento particular que deben recibir, evitando así que los nodos deban mantener información sobre el estado de los flujos de tráfico. En los nodos frontera (*Edge Router*, ER), los paquetes serán clasificados y acondicionados (funciones de marcado, espaciado, funciones policía, etc.). Mientras, en los nodos interiores (*Core Router*, CR) se crean un grupo de mecanismos que tratarán a los paquetes con diferentes prioridades en función de la información transportada en el DSCP. A ese trato especial se le conoce con el nombre de comportamiento por salto (*Per-Hop Behavior*, PHB). La arquitectura DiffServ depende en gran medida de las funciones policía y de marcado de los nodos frontera para poder asegurar que se proporcionan los servicios propuestos.

En la actualidad, el IETF ha estandarizado dos PHB: el EF PHB (*Expedited Forwarding PHB*) [RFC3246] y el AF PHB (*Assured Forwarding PHB*) [RFC2597] [RFC3260], también conocido este último como Servicio Asegurado. Tal y como se describe en [RFC3246], el servicio EF PHB se puede emplear para construir un servicio extremo a extremo que garantiza pocas pérdidas, bajos retardos, poco *jitter* y un ancho de banda a través de diferentes dominios. Es decir, que aparece ante los puntos finales de una conexión como una línea virtual alquilada (*virtual leased line*). Por otro lado, la idea que subyace en el servicio AF es asegurar un caudal (*throughput*) mínimo a cada fuente, que normalmente es la tasa contratada, también denominada CIR (*Committed Information Rate*); y además, permitir a las fuentes consumir más ancho de banda del contratado si la carga de la red es baja. Para lograr estos objetivos, los paquetes de cada flujo se marcan como pertenecientes a una de las cuatro clases o instancias AF. Como se especifica en [RFC2597] [RFC3260], dentro de cada instancia AF un paquete IP puede tener un nivel de precedencia de entre tres posibles. En caso de congestión, el nivel de precedencia de un paquete determinará su importancia dentro de la clase AF a la que pertenece. Un nodo DiffServ en congestión protegerá aquellos paquetes con menor nivel de precedencia de descarte, preferiblemente eliminando los que tengan un nivel de precedencia de descarte mayor.

El servicio EF se usa mayoritariamente en aplicaciones críticas, con tráfico que requiere la más alta prioridad, por ejemplo para aplicaciones como voz sobre IP (*Voice over IP*, VoIP). En tanto que el servicio AF congrega a un mayor número de aplicaciones y en consecuencia más tráfico, pues habrá infinidad de empresas y usuarios particulares que harán uso de Internet con el deseo de que sus paquetes IP lleguen al destino siempre y cuando la cantidad de tráfico no supere una tasa contratada. Además, teniendo la convicción de que cualquier tráfico en exceso, por encima del contrato, se transmita también si hay recursos disponibles (si sobra ancho de banda) aunque con menor prioridad que el tráfico dentro del perfil contratado. Es por este motivo por el que la tesis aquí planteada se centrará en el servicio AF. Como se dijo anteriormente, el servicio AF depende fuertemente de las técnicas empleadas para acondicionar el tráfico en los nodos frontera. Por ello, es necesario un estudio del estado de la técnica de las mismas, siendo este proceso parte fundamental de la tesis doctoral aquí iniciada. Es evidente

que se producirá cierta interacción entre las técnicas seleccionadas para acondicionar el tráfico y los mecanismos de planificación (*scheduling*) aplicados en los nodos para implementar los PHB. En consecuencia, siendo también ineludible incluir en la tesis un estudio del estado del arte de estos mecanismos.

1.1.1. Conceptos de DiffServ

Antes de pasar a conocer el estado de la técnica asentaremos los conceptos que emplearemos en éste y posteriores capítulos.

1.1.1.1. Dominio DiffServ

Un dominio de Servicios Diferenciados es un conjunto de nodos contiguos que operan con una política de provisión de servicio común y con un grupo de PHB implementados en cada nodo. Los límites del dominio DiffServ quedan claramente establecidos por los nodos frontera, que clasifican y acondicionan el tráfico de entrada al dominio, asegurando así que los paquetes que circulan por el dominio van marcados con el apropiado comportamiento de salto (PHB) de alguno de los PHB implementados en ese dominio. Los nodos dentro del dominio seleccionan el trato a dar a los paquetes basándose en el DSCP, puesto que los distintos valores del DSCP se corresponden con los PHB implementados en el dominio.

La introducción de nodos que no implementan los Servicios Diferenciados dentro de un dominio DiffServ provocará resultados impredecibles como se indica en [RFC2475], pudiendo impedir incluso la consecución del acuerdo de nivel de servicio (*Service Level Agreement*, SLA). Comúnmente, un dominio DiffServ está formado por una o más redes bajo la misma administración, por ejemplo un proveedor de servicios de Internet. Quedará bajo la responsabilidad del administrador de ese dominio garantizar que los recursos adecuados estarán disponibles para sostener el SLA que el dominio ofrece.

Como ya hemos mencionado, un dominio DiffServ consta de nodos frontera y nodos interiores (véase Fig. 1.1). Los nodos frontera interconectan un dominio DiffServ con otros dominios (DiffServ o no), mientras que los nodos interiores sólo están conectados a otros nodos interiores o a nodos frontera. Ambos tipos de nodos deben ser capaces de aplicar el PHB apropiado a un paquete. Además, los nodos frontera realizan tareas de clasificación y acondicionado del tráfico si así se requiere.

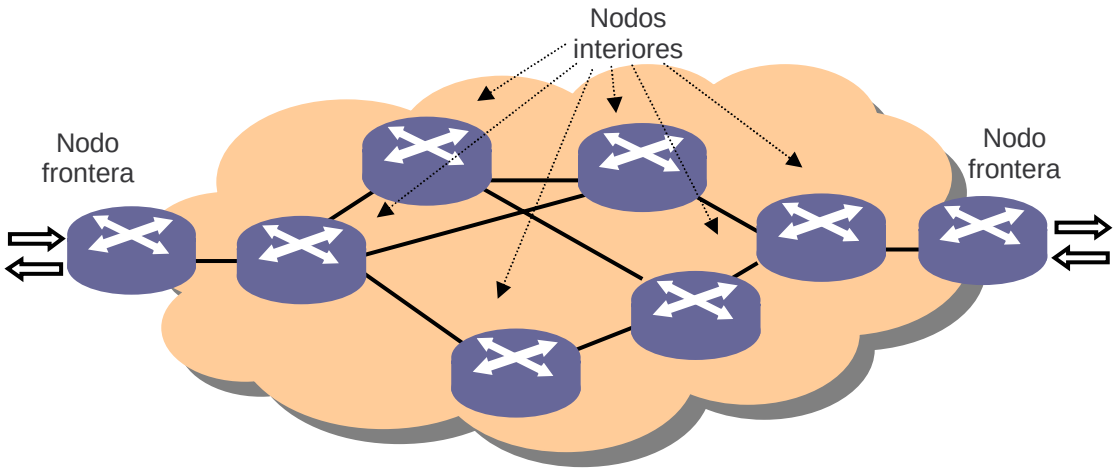


Fig. 1.1 Dominio de Servicios Diferenciados

Una región de Servicios Diferenciados es un conjunto de uno o más dominios DiffServ contiguos. Los dominios DiffServ que coexisten en una región DiffServ pueden utilizar PHB diferentes y tener correspondencias distintas entre los DSCP y los PHB a aplicar. Sin embargo, para permitir que los servicios puedan extenderse por distintos dominios, éstos deben establecer un SLA que especifique cómo el tráfico de un dominio se acondiciona en la frontera para pasar a otro dominio manteniendo la misma QoS. Es posible encontrarnos con varios dominios dentro de una región que adoptan la misma política de servicio con el mismo grupo de PHB definidos y con las mismas correspondencias entre DSCP y PHB. En este caso es por tanto innecesario el acondicionado de tráfico entre dominios.

1.1.1.2. Acondicionador de tráfico

El administrador del dominio DiffServ es el encargado de decidir cómo realizar el acondicionado del tráfico para consumar un acuerdo de nivel de servicio (SLA). Efectivamente, el SLA describe el servicio contratado e identifica el servicio que debe recibir el cliente. Dado que el SLA incluye en la mayoría de ocasiones consideraciones de naturaleza económica que caen fuera de la funcionalidad de los Servicios Diferenciados (como por ejemplo el coste del servicio), se ha acuñado un nuevo término: la especificación de nivel de servicio (*Service Level Specification*, SLS). El SLS define el servicio que recibe un flujo de tráfico dentro de un dominio DiffServ. Para poder proporcionar un SLS, se han de especificar los perfiles de tráfico, la clasificación de paquetes, las reglas de marcado (o remarcado) y en general, las tareas que se han de llevar a cabo en el caso de que los flujos de tráfico no se acomoden con los perfiles dados. A este conjunto de especificaciones se le denomina especificación de acondicionado del tráfico (*Traffic Conditioning Specification*, TCS). Es obvio que un TCS siempre deriva de un SLS.

Como resultado, el TCS incluye reglas de acondicionado de tráfico con el objetivo de que el cliente reciba el servicio que contrató. El acondicionado de tráfico incluye tareas como medición, marcado, espaciado, funciones policía, etc. La complejidad del acondicionador de tráfico (*Traffic Conditioner*, TC) dependerá del servicio que se oferta, pudiendo ir desde un simple remarcado del DSCP a elaboradas operaciones de funciones policía. Según [RFC2475] un acondicionador de tráfico se define como "una entidad que realiza funciones de acondicionado de tráfico y que puede contener medidores, marcadores, descartadores y espaciadores [...]", siendo por tanto capaz de alterar las características temporales del flujo para adaptarlo a un perfil de tráfico.

La Fig. 1.2 muestra una visión lógica de un acondicionador de tráfico. Observe que previamente a cualquier operación, el tráfico se debe clasificar. Los clasificadores seleccionan paquetes del flujo de tráfico basándose en el contenido de alguna porción de la cabecera del paquete. Básicamente existen dos tipos de clasificadores, uno se fundamenta en el uso del DSCP y recibe el nombre de clasificador de comportamiento agregado (*Behaviour Aggregate Classifier*). El segundo se basa en la combinación de uno o más campos de la cabecera (dirección IP origen, dirección IP destino, puerto origen, puerto destino, interfaz de entrada, etc.) y se denomina clasificador multicampo (*Multi-Field Classifier*). Una vez marcados los paquetes se dirigirán a alguno de los componentes lógicos del acondicionador.

Los tres elementos que forman el acondicionador de tráfico son: medidor, marcador y espaciador/descartador. Desde luego, no siempre se implementan todos ellos. Por ejemplo, si las fuentes de tráfico pudiesen generar tráfico de modo ilimitado sin comprometer las prestaciones de la red, el espaciador/descartador no aparecería. Un medidor, como su nombre indica, mide la velocidad o tasa a la que llegan los paquetes al acondicionador para así determinar la conformidad con los parámetros de tráfico y disparar una acción particular. Por ejemplo, si un usuario final tiene un contrato de 128 Kbps, el medidor estima la velocidad y si ésta sobrepasa el contrato entonces se llevará a cabo una acción diferente de si la velocidad está por debajo del contrato. Acciones comunes son: no hacer nada, marcar (o remarcar) el paquete con un determinado DSCP u otra operación como descartar el paquete o espaciarlo.

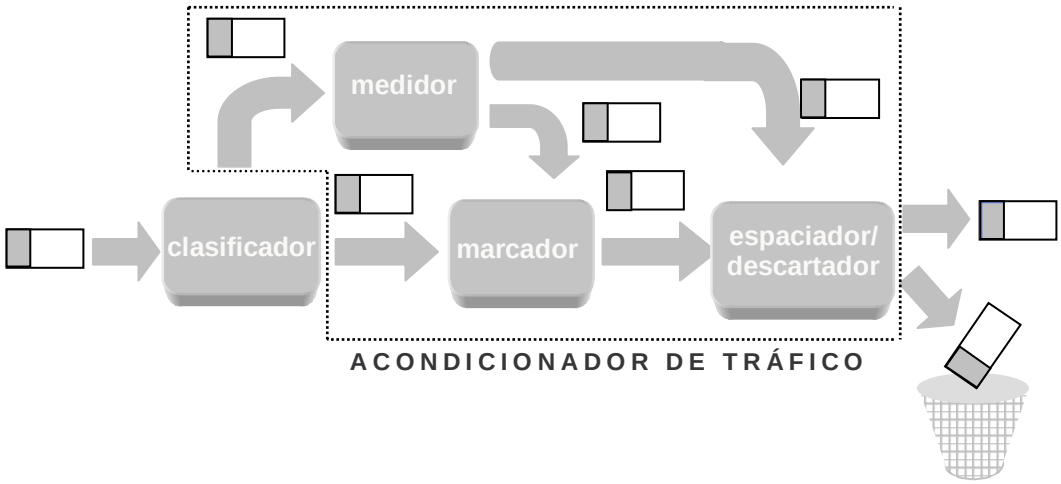


Fig. 1.2 Acondicionador de tráfico y sus componentes lógicos (medidor, marcador y espaciador/descartador)

El marcador escribe (o re-escribe) en el campo Servicios Diferenciados de la cabecera IP un DSCP determinado. Es decir, haciendo que el paquete pertenezca a un PHB específico. Por su parte, el espaciador retarda algunos o todos los paquetes del flujo para forzarlo a cumplir el perfil de tráfico. Puesto que el espaciador tendrá una cola finita, algunos de los paquetes puede que sean eliminados si la cola se llena. Finalmente, el descartador descarta algunos o todos los paquetes que le llegan con el mismo objetivo de hacer cumplir el perfil de tráfico.

Los acondicionadores de tráfico se encuentran habitualmente en los nodos frontera. En ocasiones excepcionales se pueden encontrar también en los nodos interiores, aunque en este caso estaríamos incumpliendo uno de los criterios de diseño de DiffServ puesto que estamos introduciendo complejidad en el interior de la red. Nótese además, que un nodo frontera no tiene por qué ser obligatoriamente un dispositivo de enrutamiento, podría tratarse del último sistema hardware o software (PC, controladores, etc.) sobre el que el administrador del dominio DiffServ tiene poder. Véase el ejemplo de la Fig. 1.3. Así, según el ejemplo de la figura, las tareas de acondicionado del tráfico se pueden realizar en el propio computador del usuario a través de controladores proporcionados por el administrador o ISP (Fig. 1.3. a)), el enrutador de salida de una empresa u organización (Fig. 1.3 b)) o en el enrutador de entrada al *backbone*. Como se indica en [RFC2475] existen algunas ventajas al realizar el acondicionado cerca de las propias fuentes de tráfico. En primer lugar, resulta sencillo para las fuentes de tráfico tener en cuenta las preferencias de cada aplicación a la hora de marcar un paquete para que posteriormente reciba un tratamiento mejor. Además, también es mucho más sencillo clasificar el tráfico cuando tenemos sólo uno o varios flujos de tráfico que cuando tenemos un agregado con quizá cientos de flujos.

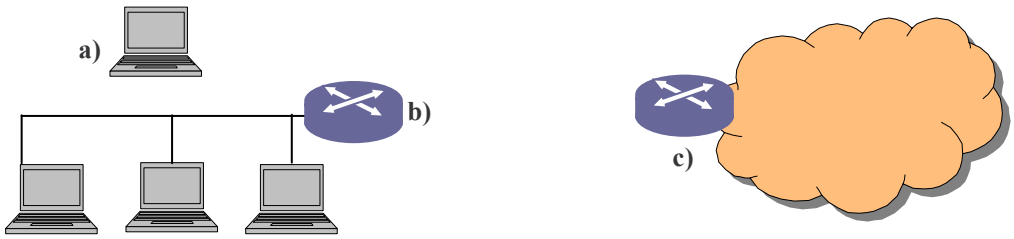


Fig. 1.3 Localización de los acondicionadores de tráfico: a) en la propia fuente de tráfico mediante controladores, b) en el enrutador de salida de una organización u empresa; c) en el nodo de entrada a la red

1.1.1.3. Comportamiento por saltos

Formalmente, un comportamiento por salto (PHB) es una descripción del comportamiento de encaminado observable externamente que un nodo DiffServ aplica a un conjunto de paquetes marcados con el mismo DSCP. En términos más concretos, un PHB hará referencia a mecanismos de planificación y gestión de las colas que permitan obtener los recursos especificados para una colección de paquetes en concreto. Un ejemplo sencillo sería un PHB que garantizase el x% del ancho de banda para los paquetes que pertenezcan a él. Hasta la fecha existen cuatro PHB estandarizados que son: el PHB por defecto, el PHB selector de clase, el PHB de encaminamiento asegurado (AF PHB) y el PHB de encaminamiento rápido (EF PHB). Éstos últimos ya comentados en la introducción. En la cabecera IP no se indica un PHB como tal sino que se emplea el DSCP para hacer corresponder determinados DSCP con ciertos PHB. El campo DiffServ de la cabecera IP (denominado en IP v.4 como *Type of Service*) consta de 8 bits (vea la Fig. 1.4), donde los 6 primeros forman el DSCP y los 2 últimos no se utilizan (actualmente en proceso de experimentación para redes ECN (*Explicit Congestion Notification*)). Esto da como resultado un total de 2^6 posibles valores de DSCP. No obstante, según las recomendaciones de [RFC2474] sólo 21 de ellos están en uso, permitiendo de este modo asignar DSCP sin uso actual a futuros PHB. En la próxima versión del protocolo IP, IP v.6, el campo *Type of Service* pasa a denominarse *Traffic Classes* manteniendo la misma funcionalidad.

El PHB por defecto especifica que un paquete marcado con un valor de DSCP (recomendado) de 000000 obtendrá el servicio tradicional *best-effort* por parte de un nodo DiffServ. Además, en el caso de que un paquete llegue marcado con un valor de DSCP que no se corresponde con ninguno de los PHB que están implementados en ese nodo, el paquete recibirá el PHB por defecto.

Para preservar la compatibilidad con el anterior diseño de la cabecera IP en el que existía el campo Precedencia IP, los valores DSCP son de la forma xxx000 donde x es un valor binario. A los distintos valores xxx000 se les llama selectores de clase. Observe que el PHB por defecto se corresponde con uno de los selectores de clase (000000). Estos PHB tienen casi el mismo comportamiento que los nodos que implementan la política de precedencia IP. Como ejemplo, un paquete con un DSCP de 110000 recibirá un trato preferente a la hora de ser encaminado frente a un paquete con un DSCP de 001000. De este modo aseguramos un nivel de compatibilidad entre los Servicios Diferenciados y los nodos que sólo conocen la precedencia IP.

Mencionamos en la introducción que los Servicios Integrados, a través del protocolo de reserva de recursos RSVP, eran capaces de garantizar de modo riguroso un ancho de banda determinado. En los Servicios Diferenciados es el PHB de encaminamiento rápido (EF PHB) el encargado de ofrecer un servicio con pocas pérdidas, baja varianza, poca varianza del retardo y garantizar un ancho de banda. Aplicaciones como VoIP, vídeo bajo demanda o enseñanza virtual requieren este tratamiento robusto por parte de la red. De esta manera, el EF PHB estará indicado para aplicaciones críticas, es decir, para el tráfico que requiera la más alta prioridad. Según las recomendaciones, el EF PHB emplea el DSCP 101110, por lo que sólo habrá una instancia de clase EF en un dominio DiffServ.

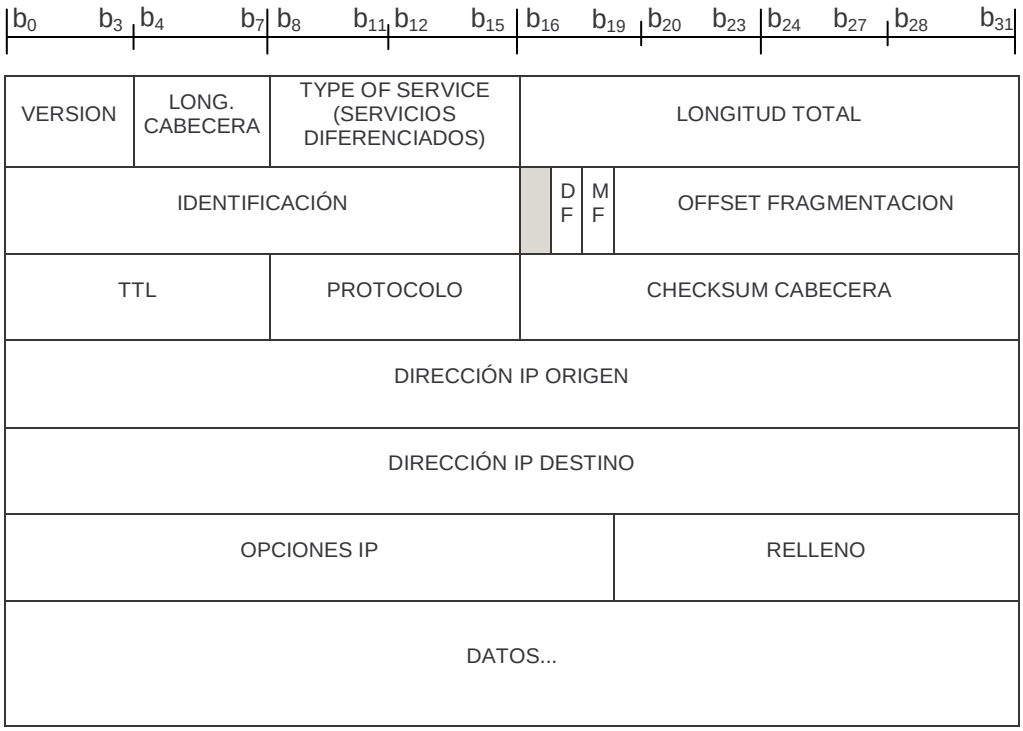


Fig. 1.4 Cabecera del protocolo de Internet IP

La equivalencia, algo tosca, con el servicio de control de carga de los Servicios Integrados sería el servicio asegurado AF PHB. Los objetivos de este servicio son dos: asegurar un caudal mínimo a cada fuente, que normalmente es la tasa contratada CIR y además, permitir a las fuentes consumir más ancho de banda del contratado si la carga de la red es baja. Al ancho de banda excedente cuando todas las fuentes tienen un caudal igual a la velocidad contratada se le denomina ancho de banda en exceso. En [RFC2597] [RFC3260] se definen cuatro instancias diferentes (antes denominadas clases) AF1, AF2, AF3 y AF4. Normalmente, cuando se implementa el AF PHB se asigna una cierta cantidad de ancho de banda a cada una de las instancias, dependiendo del SLA establecido por el administrador o el proveedor de servicios. Dentro de cada instancia AFx se asigna a cada paquete IP uno de entre tres niveles de precedencia o niveles de descarte. Cuando se produce una situación de congestión en la red, los nodos DiffServ intentarán proteger a los paquetes con menor nivel de precedencia descartando antes a los paquetes con mayor nivel de precedencia. Así, los paquetes marcados como AFx1 tendrán menos probabilidad de ser descartados que los AFx2, y a su vez los AFx2 menos probabilidad de ser descartados que los AFx3. Por lo tanto la *y* dentro de una instancia AFxy nos indica la probabilidad de tirar un paquete dentro de una misma clase. La Tabla 1.1 muestra los códigos DSCP recomendados para el servicio AF.

Tabla 1.1 Valores recomendados del campo DSCP para el servicio asegurado AF y sus correspondientes valores de precedencia de descarte

Precedencia de descarte	Instancia 1	Instancia 2	Instancia 3	Instancia 4
Precedencia de descarte baja	001010	010010	011010	100010
Precedencia de descarte media	001100	010100	011100	100100
Precedencia de descarte alta	001110	010110	011110	100110

Es usual que los acondicionadores de tráfico marquen los paquetes conformes con el perfil de tráfico de la fuente de la que proceden como *dentro del perfil* (*in-of-profile*, *in*), y aquellos que caen fuera del mismo como *fuera del perfil* (*out-of-profile*, *out*). En este caso, sólo estamos

empleando dos niveles de precedencia. Si quisiéramos emplear los tres niveles, generalmente los paquetes *in* se colorean como *verdes*, y los paquetes *out* como *amarillos* y *rojos* (con salvedades). Una parte de la literatura sobre Servicios Diferenciados se ha centrado en los acondicionadores de tráfico, buscando aquellos que gracias a su funcionamiento y a una buena interacción con la implementación del PHB son capaces de proporcionar un servicio asegurado AF con QoS extremo a extremo. El primer objetivo, asegurar los contratos del usuario final, lo han alcanzado varios de los esquemas propuestos con mayor o menor exactitud. En cuanto al segundo objetivo, que las fuentes que componen el agregado utilicen de modo justo el ancho de anda en exceso, existe discrepancia sobre qué se entiende por justicia. Algunos autores coinciden en que un reparto justo del ancho de banda en exceso significa que éste se distribuye de modo equitativo entre las distintas fuentes que componen el agregado. Por el contrario, otros autores defienden la postura de que un reparto justo es aquel en el que las fuentes se benefician del ancho de banda en exceso de modo proporcional a sus respectivos contratos. En este estudio, trabajaremos desde los dos enfoques intentando encontrar solución para ambos casos.

El índice de justicia

Para analizar la justicia en la distribución del ancho de banda en exceso utilizaremos el índice de justicia de Jain [JAIN91]. Este índice de justicia *f* se obtiene según la ecuación 1.1. A medida que *f* se aproxima a 1, más justicia habrá en el reparto del ancho de banda no contratado. En función de si se considera un reparto justo como equitativo o como proporcional al contrato nos encontramos con dos significados de la variable x_i:

$$f = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \cdot \sum_{i=1}^{n} x_i^2}; \quad f \leq 1 \tag{1.1}$$

- Si consideramos como justa la distribución equitativa entre las distintas fuentes que componen el agregado, entonces x_i es el caudal en exceso que obtiene la fuente *i*, y *n* es el número de fuentes que componen el agregado.
- En el caso de considerar como justo un reparto proporcional al contrato de cada fuente, x_i es el caudal en exceso que obtiene la fuente *i* dividido por el contrato de esta fuente (ec. 1.2). Mientras *n* continúa siendo el número total de fuentes que llegan al nodo frontera

$$x_i = \frac{caudal\ \ total\ de\ fuente\ \ i - contrato\ de\ fuente\ \ i}{contrato\ de\ \ fuente\ i} \tag{1.2}$$

Es elemental remarcar la importancia de mejorar los índices de justicia aunque sea en pequeñas cantidades. Como ejemplo, imagine que tenemos una red con dos fuentes, s_1 y s_2, que contratan 1 y 10 Mbps respectivamente. Suponga que la capacidad del enlace es de 33 Mbps. Una distribución justa (equitativa) del ancho de banda en exceso significa que s_1 obtiene 2 Mbps y s_2 obtiene 20 Mbps del ancho de banda no contratado. En esta situación tendríamos un índice de justicia de 1. En el caso de que s_1 lograse 3 Mbps y s_2 19 Mbps, un megabit menos de su exceso correspondiente, el índice de justicia disminuye a 0,95 (¡sólo 0,05 puntos menos!). En consecuencia, pequeños incrementos en el valor del índice de justicia representan mejoras notables en las prestaciones finales.

1.1.2. Gestión y control activo de la congestión

Como hemos visto, tanto en los nodos frontera como en los nodos interiores de un dominio DiffServ, es imprescindible realizar tareas de planificación y control de la congestión que ayuden a implementar los comportamientos por salto que los diferentes paquetes deben recibir. Veremos por tanto en esta sección diferentes disciplinas de planificación de colas y métodos de

control de congestión. La planificación gestiona la cantidad de ancho de banda reservada para cada clase de servicio en un puerto de salida del nodo. Controla así el acceso de las diferentes clases de servicio a unos recursos limitados de red como es el ancho de banda del enlace. Por consiguiente, la planificación decide cuándo y qué paquetes se sacan de una cola, conmutando éstos a la interfaz de salida. En cuanto a los métodos de control de congestión, éstos deben controlar el número de paquetes de una cola (la profundidad de la cola), decidiendo cuándo y qué paquetes se descartan debido a congestión o incluso antes de que ésta se produzca.

Algunos de los mecanismos de planificación más comunes, ampliamente conocidos todos ellos [CISC04], son: primero en llegar primero en servirse (*First In First Out*, FIFO), asignación de cola por prioridad (*Priority Queuing*, PQ), asignación justa de cola (*Fair Queuing*, FQ) y asignación justa de cola por pesos (*Weighted Fair Queuing*, WFQ).

El mecanismo FIFO es el más básico de todos. Todos los paquetes, independientemente de la clase (EF, AFxy, etc.) se tratan por igual, colocándolos en una única cola y sirviéndolos en el mismo orden en el que llegaron a ella. Este método supone una carga de trabajo pequeña para el nodo de la red. Además FIFO tiene un comportamiento fácilmente predecible. Como contrapartida, no permite tratar de modo diferente a las distintas clases e impacta en todos los flujos por igual, pudiendo provocar retardos en el tráfico de tiempo real. Asimismo, flujos a ráfagas podrían consumir por completo el espacio de la cola, negando el servicio a otros flujos hasta que la ráfaga se sirva. Resulta evidente que no se utilizan la colas FIFO por sí solas para implementar los PHB.

La asignación de cola por prioridad (PQ) es la base para poder implementar los diferentes PHB (véase la Fig. 1.5). El PQ clásico coloca los paquetes en distintas colas de prioridad tras haberlos clasificado. Los paquetes de una cola son servidos sólo si todas las colas de mayor prioridad están vacías. Dentro de cada cola los paquetes se sirven en orden FIFO. Las ventajas de este método son, que tras el método FIFO, PQ es el que menos carga de trabajo supone al nodo, y que permite organizar los paquetes, tratando a unas clases de modo diferente a otras. La principal limitación que presenta es que una excesiva cantidad de tráfico de alta prioridad puede provocar inanición en las colas de prioridad más baja. La solución que se conoce es la asignación de cola por prioridad con velocidad controlada (*rate-controlled PQ*). En este caso la cola más prioritaria sólo se sirve antes que las de menor prioridad si la cantidad de tráfico en la de mayor prioridad está por debajo de un umbral.

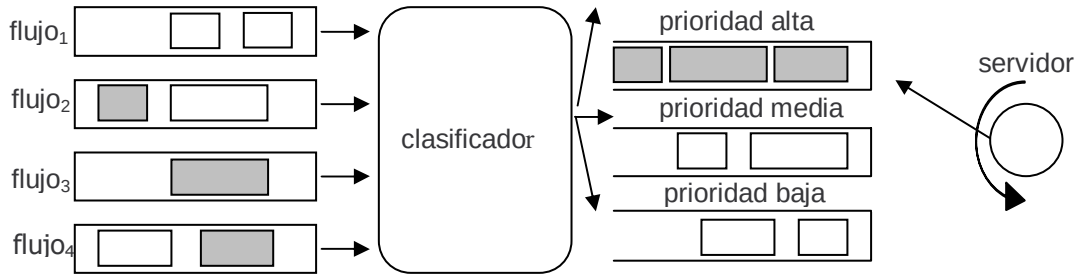


Fig. 1.5 Cola por prioridad (PQ)

La asignación justa de cola (FQ) se diseñó para asegurar que cada flujo disponga de un acceso justo a los recursos de la red, evitando que el tráfico a ráfagas consuma más ancho de banda del que le corresponde. En FQ, el sistema primero clasifica los paquetes en flujos y les asigna una cola dedicada a cada uno de esos flujos. Las colas se sirven en un orden *round-robin*, es decir, en orden secuencial de la primera a la última y vuelta a la primera. La gran ventaja de FQ es que el tráfico a ráfagas de un flujo no degrada la calidad del resto de flujos, puesto que cada flujo tiene su propia cola. Sin embargo, este esquema también manifiesta ciertas

restricciones. Normalmente los proveedores implementan FQ mediante software, lo que limita su aplicación a interfaces de baja velocidad. FQ se diseñó para asignar exactamente la misma cantidad de ancho de banda a cada flujo, luego no soporta flujos con diferentes requerimientos de ancho de banda. Por último, FQ necesita de un proceso de clasificado en flujos, lo que es inviable cuando hablamos de nodos interiores que reciben miles de flujos distintos. Un esquema de FQ se muestra en la Fig. 1.6.

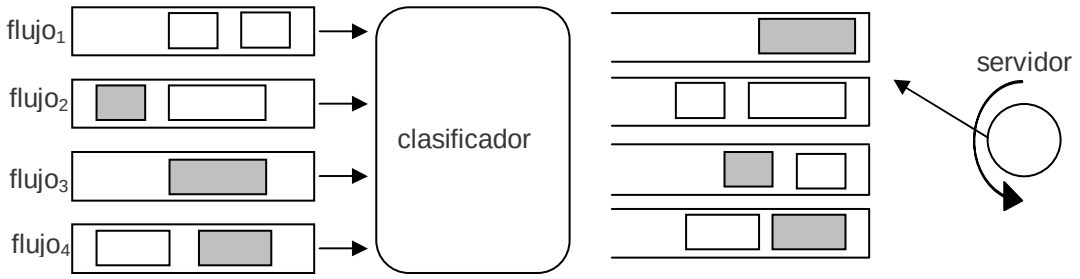


Fig. 1.6 Asignación de cola justa (FQ)

Finalmente, la asignación justa de cola por pesos (WFQ) se diseñó para solventar las deficiencias de FQ. WFQ (véase la Fig. 1.7) da cabida a flujos con diferentes necesidades de ancho de banda, para lo que asigna a cada cola un peso dándoles así un porcentaje de ancho de banda diferente. Además, no le afecta el tamaño de los paquetes, logrando que los flujos con paquetes de tamaño mayor no se beneficien frente a flujos con tamaño de paquete menor, lo que incrementa la complejidad del esquema. El funcionamiento de WFQ se basa en calcular y asignar un tiempo de fin *t* a cada paquete. Conocida la velocidad de la interfaz de salida, el número de colas, el peso relativo asignado a cada cola y la longitud de cada paquete, WFQ calcula y asigna un tiempo de fin *t'* a cada paquete que llega. El planificador escogerá el paquete con menor tiempo *t'* y lo conmutará a la salida. Por lo tanto *t'* representa el orden en el que el paquete se transmite por el puerto de salida. La principal virtud de WFQ es que proporciona la protección adecuada a cada clase de servicio asegurando una porción mínima de ancho de banda independientemente del comportamiento de las otras clases de servicio. Como desventajas, cabe destacar que su implementación es bastante compleja. Además, los proveedores suelen implementarlo en software, limitando su aplicación a interfaces de baja velocidad. Desde su creación han aparecido variantes como WFQ basado en clases (*Class-Based WFQ*, CBWFQ) o WFQ de caso peor (*Worst-Case Fair WFQ*).

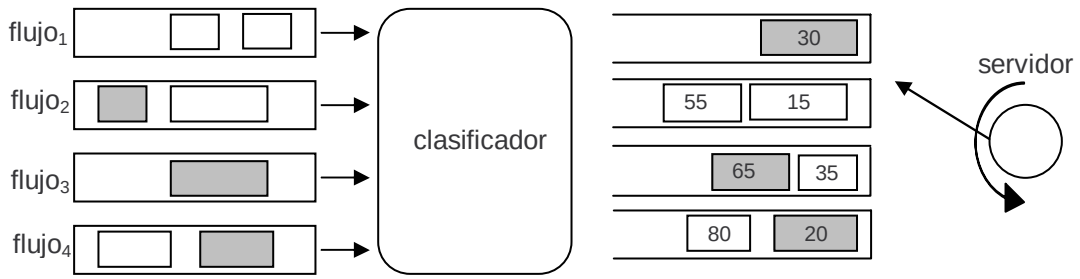


Fig. 1.7 Asignación justa de cola por pesos (WFQ)

Por su parte, los métodos de control de congestión presentan quizá menos variedad. El primero que vamos a ver es *Tail Drop*. Este método significa que hay una ausencia total de control de la congestión. Cuando un paquete llega a una cola que está llena se descarta, y lo

mismo ocurre con los posteriores hasta que haya espacio disponible. La sencillez de este método no es motivo suficiente para emplearlo, puesto que son muchas sus desventajas. El principal problema que plantea es que es extremadamente pobre para emplearlo con tráfico TCP (Protocolo de Control de Transporte, *Transport Control Protocol*). Aproximadamente, entre un 80 y un 95% del tráfico de las redes IP es TCP [CANO01] [FLOY01]. Este protocolo entiende que si se tira un paquete hay congestión en la red, controlando así la tasa de transferencia. *Tail Drop* hace que todas las conexiones TCP que atraviesan la cola reduzcan su ventana al mismo tiempo, resultando en un proceso conocido como sincronización global. Esto produce oscilaciones drásticas en el caudal de tráfico que dan como resultado un ineficiente uso de los recursos.

Existen otros métodos de control de congestión que permiten responder a la congestión antes de que ésta se produzca. Se les califica como métodos activos. Es decir, en vez de esperar hasta que la cola se desborde, los paquetes se marcan de un modo determinado o se descartan para evitar dicho desbordamiento. Los dos mecanismos de control de congestión activos más conocidos son ECN [RFC3168] y la Detección Temprana Aleatoria (*Random Early Detection*, RED) [FLOY93] incluyendo todas sus variantes. ECN es un método que hoy en día todavía se encuentra en fase experimental. Mientras, los mecanismos RED están desplegados actualmente en la mayoría de las redes IP.

Con RED, el descarte de un solo paquete es señal suficiente de congestión para aquellas máquinas que emplean TCP. Al descartar un paquete, el enrutador está enviando una advertencia implícita a una fuente TCP de que el paquete ha sido descartado por detectarse congestión en algún punto a lo largo del camino. Como respuesta a esta advertencia, la fuente TCP reduce su ventana de transmisión. RED emplea un perfil de descarte de paquetes para controlar la posible agresividad del modelo. Así, el perfil define la probabilidad de tirar un paquete en función del estado de ocupación de la cola. Si el estado de ocupación de la cola permanece por debajo de un umbral mínimo min_{th} (configurado por el administrador), el paquete tiene probabilidad de descarte cero. Éste es el modo de operación normal. Si el nivel de ocupación excede un umbral máximo max_{th} (configurado también por el administrador), el paquete se descarta con probabilidad uno, entrando en el modo de control de la congestión. Si el estado de ocupación de la cola permanece entre min_{th} y max_{th}, modo de prevención de la congestión, el paquete se tirará con una probabilidad definida por el administrador.

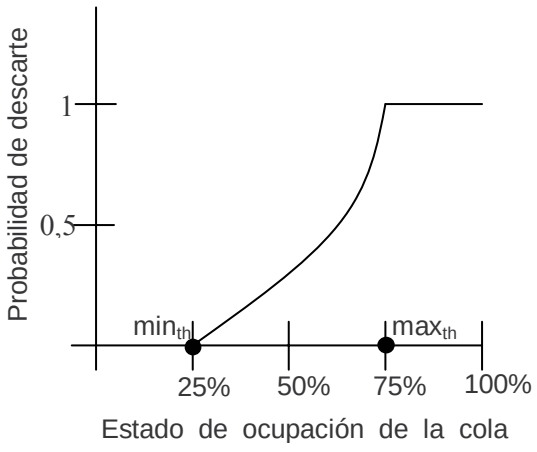


Fig. 1.8 Ejemplo RED. Perfil de descarte

Generalmente, se configurarán los parámetros de RED para mantener la ocupación media de la cola entre estos dos umbrales min_{th} y max_{th}. Como ejemplo, observe la Fig. 1.8 donde una ocupación de la cola del 25% supone una probabilidad de descarte 0. Si la cola llega al 50% de ocupación la probabilidad aumenta a un 0,5, y si llega al 75% de ocupación se descartarán todos los paquetes. Uno de los retos a la hora de implementar RED es configurar los umbrales min_{th} y

max_{th} y la probabilidad de descarte. Además, son varios los métodos propuestos para calcular la profundidad de la cola. En la literatura especializada se han presentado diversas variantes de RED, las cuales referenciaremos a lo largo de esta tesis.

1.2. Estado de la técnica

Clark y Fang introdujeron en [CLAR98] el acondicionador de tráfico denominado Ventana Deslizante Temporal (*Time Sliding Window*, TSW) que se ha convertido en referencia clave de los estudios posteriores sobre acondicionadores de tráfico. TSW tiene dos componentes: un estimador de tasa y un marcador. El estimador de tasa evalúa la tasa media con la llegada de un nuevo paquete y emplea un tiempo *winlenght* para consecutivamente ir eliminando la historia pasada. Por su parte, el marcador etiqueta los paquetes como *in* si la estima no excede la tasa contratada. Si se supera la tasa contratada entonces los paquetes son etiquetados como *in* o *out* con probabilidad *p*. También en este trabajo se presenta RIO (RED (*Random Early Detection*) *In and Out*) [CLAR98] como una de las propuestas más importantes para implementar el control de la congestión en el servicio AF PHB. RIO es la combinación de dos algoritmos RED con diferentes curvas de probabilidad, de manera que los paquetes *out* tienen una mayor probabilidad de ser descartados (véase Fig. 1.9). Cada algoritmo RED tiene tres parámetros (p, min_{th} y max_{th}) que definen la fase de operación normal [0, min_{th}), la fase de prevención de la congestión [min_{th}, max_{th}) y la fase de control de la congestión [max_{th}, ∞). En [FLOY93] se describe RED en profundidad. RIO emplea una única cola FIFO para servir tanto los paquetes *in* como los *out*. La probabilidad de descartar un paquete *out* depende del número total de paquetes que llegan al nodo, mientras que la probabilidad de descartar un paquete *in* depende exclusivamente de los paquetes *in* que ocupan la cola (*buffer*). La combinación TSW-RIO manifiesta algunos problemas de diseño como el hecho de favorecer a aquellas fuentes con menor tiempo de ida y vuelta (*Round Trip Time*, RTT) y la dificultad de seleccionar adecuadamente sus parámetros de configuración. Asimismo, sus autores demuestran que el ancho de banda contratado por cada fuente se garantiza sólo para TCP Sack, que no es una implementación usual de TCP en Internet, y no para TCP Reno que es la implementación más común. En este trabajo no se presentan resultados sobre la distribución del ancho de banda en exceso entre las diferentes fuentes, que era uno de los objetivos del servicio AF.

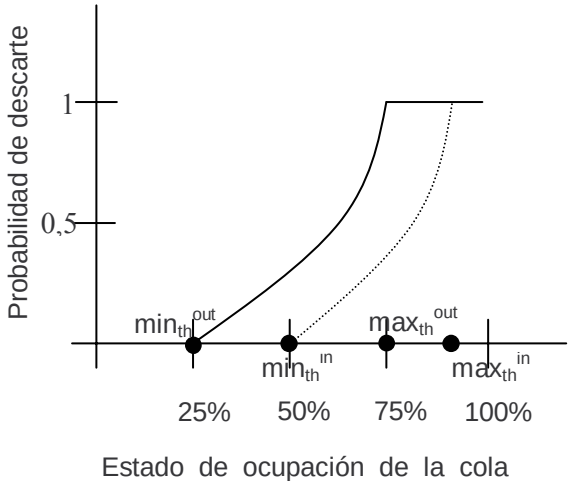


Fig. 1.9 Ejemplo RIO. Perfil de descarte de paquetes *out* (línea continua), perfil de descarte de paquetes *in* (línea discontinua)

En contraste con los resultados ilustrados en [CLAR98], Ibañez y Nichols afirmaron en [IBAN98] que la superioridad de los mecanismos de marcado basados en el uso de fichas (*tokens*) se debe a su capacidad de permitir la transmisión de ráfagas deterministas de paquetes *in*, mientras que los estimadores de tasa media marcan con cierta probabilidad algunos paquetes como *in* y otros como *out*, aún estando por debajo de la tasa contratada. Este artículo también expone la dependencia existente con los parámetros de la red como el RTT o tasas contratadas a la hora de asegurar QoS. En consecuencia, según [IBAN98] el esquema AF no puede ofrecer un servicio cuantificable al tráfico TCP. Las simulaciones de este trabajo se realizaron con RIO y el algoritmo de marcado del cubo de fichas (*token buket*).

El siguiente paso en el desarrollo de DiffServ aparece con el uso de más de dos niveles de precedencia [RFC2697] [RFC2698]. Los paquetes TCP se *colorean* de rojo, amarillo o verde, y los paquetes UDP (*User Datagram Protocol*) sólo como rojos o verdes. Así es posible lograr una distribución justa (equitativa) del ancho de banda en exceso entre fuentes que responden a la congestión y fuentes que no responden. Los resultados de simulación incluidos en [GOYA99], donde se emplea una modificación del algoritmo RED denominada *Single Average Multiple Threshold* para prevenir la congestión, muestran algunas mejoras en el reparto del ancho de banda sobrante cuando se aplica este esquema. Sin embargo, el problema persiste cuando sólo existen fuentes TCP. Ya que no queda claro cómo se distribuye este ancho de banda no contratado entre las distintas fuentes TCP. Llegados a este punto no existía ningún algoritmo con la suficiente precisión como para garantizar con exactitud los contratos de cada fuente TCP y que a la vez, permitiera distribuir de manera justa (equitativa o proporcionalmente) el ancho de banda en exceso.

W. Lin *et al.* introdujo en [LIN99] una versión mejorada del acondicionador TSW al que denominaron TSW mejorado (*Enhanced Time Sliding Window*, ETSW). La principal característica de ETSW es que uno de los parámetros de configuración que en TSW era fijo (parámetro β) se convierte ahora en adaptable. Al comparar las prestaciones de TSW-RIO y ETSW-RIO, este último tiene un comportamiento superior. Sin embargo, no se aseguran los contratos de los usuarios con tanta nitidez como lo hace el algoritmo *token bucket*. Haciendo referencia al problema del reparto del ancho de banda en exceso, el artículo presenta dos modificaciones a RIO: RIO adaptable y RIO dinámico. El primero no funciona del modo esperado en situaciones donde el RTT de los flujos que componen el agregado es heterogéneo, es decir cuando cada fuente tiene un RTT distinto. Por su parte, el RIO dinámico necesita mantener información sobre los flujos que tienen paquetes *out* en la cola, y en consecuencia, presenta problemas de escalabilidad.

En un intento de llegar a la raíz del problema, Seddigh *et al.* realizaron un estudio completo publicado en [SEDD99]. En él, mediante simulación, clarifican cuáles son los factores que afectan a los caudales TCP en una red DiffServ con nodos RIO. Algunos de estos factores fueron: RTT, tamaño de los contratos, tamaño de los paquetes IP, número de flujos e influencia de tráfico UDP (en general tráfico que no responde a la congestión). Como solución, propusieron un acondicionador de tráfico que tomara en consideración los factores anteriores como parámetros de dicho acondicionador. Los problemas que se plantean para su aplicación son: en primer lugar, qué ocurriría si hay que atravesar más de un dominio DiffServ y en segundo lugar, la posible falta de escalabilidad. Siguiendo con esta idea, los mismos autores introdujeron en [NAND00] un acondicionador de tráfico *inteligente*. Para mitigar el efecto del RTT, plantean el uso de un mecanismo que calcula la probabilidad de descarte de un paquete en función del RTT medio del agregado y el mínimo RTT del dominio DiffServ. Este acondicionador se usa en combinación con la versión tricolor de RIO [ELLO99]. Para alcanzar un reparto del ancho de banda excedente justo (proporcional), introducen un nuevo parámetro calculado a partir del contrato menor y del ancho de banda total contratado. Aunque es más sencillo conocer los contratos que los RTT, este conocimiento requiere comunicación entre dispositivos. Además, las suposiciones que se han de hacer para emplear este esquema son demasiado restrictivas (flujos TCP operando en el modo *congestion avoidance*, todos los flujos del agregado tienen el mismo RTT, etc.).

Desde un punto de vista totalmente diferente, H. Kim presenta en [KIM99] un acondicionador al que denomina de marcado justo (*fair marker*). El artículo explota la dualidad entre el uso de memoria de las colas y el consumo de fichas en un marcador de cubo de fichas para forzar un reparto justo de los recursos entre los diferentes flujos. Esta acondicionador está dentro del grupo de los denominados conocedores de flujo (*per-aggregation flow aware*). Es decir, están basados en un conocimiento parcial de los flujos individuales. Básicamente, el acondicionador distribuye fichas de modo justo entre los flujos que componen el agregado, monitorizando las fichas que van consumiendo los flujos activos. Los resultados obtenidos por simulación de este acondicionador combinado con FRED (Descarte Temprano Aleatorio Justo, *Fair Random Early Drop*) [LING97] se presentan en [ALVE00]. Este trabajo demuestra que con una adecuada sintonización de parámetros, se garantizan los contratos pero no se alcanza justicia en la distribución del ancho de banda excedente. Para solventar el problema del reparto del ancho de banda en exceso, [ALVE00] y [ANDR00] sugieren una modificación muy similar del *fair marker*. Esta modificación requiere el uso de un marcador tricolor de tasa doble o simple y FRED. Las simulaciones de ambos estudios muestran mejoras considerables en comparación con el *fair marker* o el cubo de fichas tradicional. Con todo, en estos trabajos sólo se realizaron simulaciones donde todas las fuentes tienen el mismo RTT, y cuando el número de flujos es elevado aparece de nuevo la falta de escalabilidad.

Otras soluciones menos atractivas que también han sido publicadas sugieren por ejemplo modificar el algoritmo TCP, dividiendo la ventana original de congestión en dos partes: ventana reservada (*Reserved WiNDow*, RWND) y ventana de exceso (*Excess WiNDow*, EWND) [FENG97]. Es indiscutible, que resulta una modificación no trivial de un protocolo tan extendido como TCP. Siguiendo la misma tendencia, [JIE99] propone otra modificación del algoritmo TCP, esta vez basada en ajustar el parámetro de disparo *slow start* (normalmente 0,5) a un valor proporcional a la clase de servicio DiffServ. No obstante, significa modificar un protocolo ya muy asentado como es TCP. Otra opción sería el esquema de diferenciación compartida de usuario (*User Shared Differentiation*, USD) [WANG98]. En este caso, se reparte el ancho de banda disponible de modo proporcional empleando una etiqueta de compartición de usuario (*user's shared tag*) como peso. Una de las principales diferencias que exhibe este esquema respecto a más recientes propuestas es que se implementa en los nodos interiores, al contrario de lo que indican los estándares de DiffServ.

Las últimas tendencias en cuanto al desarrollo de acondicionadores de tráfico [GEND03] [TART02] [HABI02] o bien requieren el uso de elevada señalización, o necesitan una monitorización por flujos en el nodo frontera con los consecuentes problemas de escalabilidad. Además, incluso en estas últimas propuestas que consiguen garantizar los CIR, existen claras deficiencias en cuanto al reparto del ancho de banda sobrante entre las distintas fuentes TCP que componen el agregado.

Queda demostrado que hasta la fecha no es posible encontrar un acondicionador de tráfico cuya interacción con los mecanismos de gestión de colas para implementar los PHB permita lograr los dos objetivos del servicio AF. En esta afirmación incluimos los mecanismos que actualmente están incorporados en equipos comerciales (véase Anexo A.). Algunos de los acondicionadores de tráfico propuestos no consiguen garantizar los CIR de modo estricto debido a la gran dependencia que existe con parámetros de la red como por ejemplo el tiempo de ida y vuelta. Otros, aún en condiciones favorables donde no hay diversidad en los parámetros de la red, presentan una configuración demasiado compleja que hace que cualquier pequeña variación en los valores de ésta no garantice los contratos. A su vez, existen propuestas que son capaces de asegurar los contratos de los usuarios pero que a la hora de distribuir el ancho de banda en exceso no lo hacen de modo justo o imparcial (en ninguna de las dos definiciones contempladas para el término justicia).

Es por tanto objetivo de esta tesis introducir nuevos algoritmos acondicionadores de tráfico capaces de garantizar los contratos y de conseguir un reparto justo (equitativo o proporcional según interese) del ancho de banda en exceso si la red tiene poca carga. Asimismo, no hay que olvidar que los resultados van a depender en gran medida de la interacción entre el

acondicionador de tráfico y el mecanismo seleccionado para realizar el PHB. Por lo que se intentarán aportar nuevas ideas de implementación del AF PHB.

1.3. Herramienta de simulación

La herramienta de simulación del protocolo de ventana deslizante TCP Reno fue desarrollada en [CERD00b] y ampliamente utilizada en [CERD00a] [BONI00]. Además, se empleó para validar el estudio analítico realizado en [BONI01]. Centramos este trabajo en tráfico TCP puesto que como ha quedado demostrado en diversos trabajos más del 80% de los paquetes que viajan por la red son TCP. Algunas de las características fundamentales de esta herramienta de simulación son las siguientes:

- No se ha considerado el retardo de procesado entre capas así como la interacción entre TCP y el nivel de aplicación.
- Los destinos sólo envían reconocimientos (*acknowledgements*), que no sufren pérdidas ni retrasos.
- Las fuentes TCP son del tipo *long-lived* que significa que disponen de datos ilimitados para enviar.
- El tamaño de los paquetes es de 9,188 bytes, que se corresponde con IP sobre el modo de transferencia asíncrono (*Asynchronous Transfer Mode*, ATM), y podría representar DiffServ sobre el estándar de conmutación por etiquetas multi-protocolo (*Multi-Protocol Label Switching*, MPLS), donde el uso de la tecnología ATM es inherente. En ocasiones, se utilizarán otros tamaños de paquete que serán indicados en el texto.

El tamaño de ventana máximo iguala al producto retardo por ancho de banda, usual en entornos WAN.

La herramienta de simulación dispone de un sistema de reloj que controla los eventos durante la simulación con una granularidad que depende de la velocidad del enlace. En la mayoría de las simulaciones se utilizan enlaces de 33 Mbps. Debido a la granuralidad del modelo, la capacidad efectiva puede ser un poco menor.

Capítulo 2
Propuestas de acondicionadores de tráfico en DiffServ

2.1. Introducción

El crecimiento de Internet durante los últimos años ha sido un claro exponente del auge de las tecnologías de la información. La cantidad de tráfico que circula por Internet aumenta rápidamente, y cada vez más, este tráfico proviene de aplicaciones multimedia que son sensibles a la disponibilidad de ancho de banda y el retardo que se experimenta en la red. Este hecho ha generado una clara necesidad de QoS [CERD00]. Entre las arquitecturas de QoS propuestas hasta la fecha destaca la de Servicios Diferenciados, definida en el Capítulo 1.

El enfoque de DiffServ define un grupo de mecanismos que permiten tratar los paquetes de los agregados de flujos con diferentes prioridades en función de la información contenida en el campo DiffServ de la cabecera IP. Los paquetes se clasifican y marcan para recibir un trato particular en los nodos a lo largo de su camino. Este trato es conocido como comportamiento por salto. Las funciones de acondicionado y clasificación del tráfico (medición, marcado, espaciado, etc.), más complejas, sólo es necesario implementarlas en los nodos frontera. Los nodos interiores por su parte aplicarán los comportamientos por salto a los agregados de tráfico cuyos paquetes han sido marcados adecuadamente.

El IETF ha estandarizado algunos PHB. Destacan el EF PHB y el AF PHB. El PHB Asegurado (AF PHB) [RFC2597] ofrece hasta cuatro diferentes niveles de servicio, lo que se denominan instancias AF. Dentro de cada instancia AF, un paquete IP puede pertenecer a uno de tres niveles de precedencia. Así, el usuario de un servicio asegurado AF espera una baja probabilidad de descarte mientras envíe tráfico dentro del perfil establecido. Los paquetes de flujos individuales se marcarán como *in* (dentro del perfil) o *out* (fuera del perfil) si sólo empleamos dos niveles de precedencia dentro de cada instancia AF. Los enrutadores de la red no distinguirán entre paquetes de flujos individuales pero implementarán un mecanismo de descarte donde los paquetes *in* tendrán menor probabilidad de ser descartados. De esta manera, el servicio asegurado se diseñó para asegurar al usuario final un caudal mínimo, el contrato, incluso durante periodos de congestión de la red. El ancho de banda no contratado se deberá distribuir de modo justo entre las distintas fuentes que componen el agregado. En este trabajo se entiende por justicia el reparto equitativo del ancho de banda en exceso.

En este capítulo, presentamos un estudio comparativo mediante simulaciones de diferentes tipos de acondicionadores de tráfico. La meta será deducir si son capaces de garantizar los objetivos del servicio asegurado AF. El uso de estos acondicionadores de tráfico involucra tal cantidad de parámetros (tasa de envío TPC instantánea, tiempos de ida y vuelta, *thresholds*, tasas de velocidad contratadas, etc.) que el número de paquetes marcados como *in* o *out* dependerá en gran medida de ellos. Además, los acondicionadores de tráfico interaccionan con el mecanismo de gestión de la congestión seleccionado en el enrutador, en este caso RIO [CLAR98]. Esta interacción se debe a que, en función del grado de congestión existente, el descarte de muchos paquetes *out* puede provocar una reducción drástica de la ventana TCP,

afectando en consecuencia a la tasa de envío. Introduciremos también en este capítulo un nuevo acondicionador de tráfico basado en contadores denominado *Counters Based*. A diferencia de TSW o el algoritmo de cubo licuante (*Leaky Bucket*, LB), nuestra propuesta ofrece mejores prestaciones, pues garantiza estrictamente los contratos de los usuarios. Vamos a explorar la correcta configuración de los tres mecanismos y confirmaremos si son capaces o no de ofrecer un servicio AF, para lo que no sólo tendrán que garantizar los contratos de los usuarios sino que además deben conseguir una distribución justa del ancho de banda en exceso.

EL resto del capítulo queda organizado de la siguiente manera. En la Sección 2.2 mostraremos la topología empleada en el estudio comparativo. En la Sección 2.3 describimos, evaluamos y comparamos las prestaciones de los dos acondicionadores clásicos TSW y LB. En la Sección 2.4 presentamos la nueva propuesta de acondicionador *Counters Based*. Finalmente, dedicaremos la última sección a las conclusiones.

2.2. Configuración de las simulaciones

La topología empleada en las simulaciones se muestra en la Fig. 2.1. Ocho fuentes TCP Reno generan tráfico a la máxima velocidad del enlace, establecida a 33 Mbps. Los acondicionadores de tráfico se sitúan junto a la fuente generadora, pero fuera del alcance del usuario final. El contrato de las fuentes así como los tiempos de ida y vuelta de las conexiones (RTT) será variable con el fin de estudiar diferentes escenarios.

El enrutador que multiplexa el tráfico proveniente de las ocho fuentes E_1, almacena y envía al destino los paquetes del agregado. Para evitar y gestionar la congestión en la cola del enrutador se utilizan dos algoritmos RED con diferente perfil de descarte. Este mecanismo es conocido como RIO [CLAR98] (véase el Capítulo 1 Sección 1.2). RIO calcula la ocupación media de la cola de paquetes *in*, y la ocupación media de la cola de paquetes *in* y *out*. La probabilidad de descartar un paquete *in* depende sólo del número de paquetes *in* de la cola, mientras que la probabilidad de descartar un paquete *out* depende del número total de paquetes de la cola. Cada algoritmo RED tiene tres parámetros (min_{th}, max_{th}, p), que definen el modo normal [0, min_{th}), el modo de prevención de la congestión [min_{th}, max_{th}), y el modo de control de la congestión [max_{th}, ∞). Estos parámetros se han escogido iguales a [40/70/0,02] para los paquetes *in* y [10/40/0,2] para los paquetes *out*. Para calcular el tamaño medio de la cola los valores $weight_{in}$ y $weight_{out}$ se fijan a 0,002 como recomienda [FLOY93]. Los resultados de las simulaciones tienen un intervalo de confianza del 95% calculado con una distribución normal usando 30 muestras, con un valor aproximado ±0,01 para los contratos alcanzados.

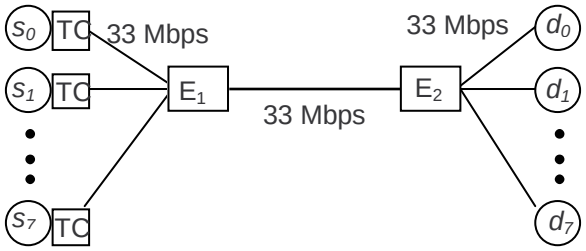


Fig. 2.1 Topología de red para las simulaciones (TC≡Acondicionador de tráfico)

2.3. Acondicionadores de tráfico clásicos

En este apartado vamos a estudiar tres acondicionadores de tráfico: *Time Sliding Window*, *Leaky Bucket* y la nueva propuesta *Counters Based*. Evaluaremos las prestaciones de cada uno de ellos, comparando los resultados, en términos de garantías de alcanzar los contratos de los usuarios cuando existen variaciones de retardo o de contrato entre las distintas fuentes, y de cómo se distribuye el ancho de banda excedente.

2.3.1. Algoritmo del cubo licuante o *Leaky Bucket*

De acuerdo con el funcionamiento de este algoritmo, es posible marcar más o menos paquetes como *in*, en función de cuán permisivo queremos que sea el sistema. *Leaky Bucket* define la variable *incremento* como el número de unidades de tiempo que pasan desde la llegada de un paquete hasta la llegada del siguiente paquete si el flujo fluye a la velocidad contratada. Se define la variable *tolerancia* como el número de unidades de tiempo que añadidas a la variable *incremento* fijan la tolerancia del sistema. Por simplicidad, expresaremos *tolerancia* como ζ veces el *incremento*, donde ζ es un valor entero o racional.

La implementación del algoritmo se puede describir como sigue. Dado el tiempo de llegada teórico (*Theoretical Arrival Time*, TAT) de los paquetes, comparamos el tiempo de llegada del paquete (*Packet Arrival Time*, PAT) con el TAT. En el caso de que TAT sea menor o igual que PAT, entonces el paquete se marca como *in*. Si el paquete llegó antes que el instante teórico TAT, pero dentro de un margen (ζ), entonces el paquete también queda marcado como *in*. De lo contrario, el paquete es *out*. Las Fig. 2.2 y 2.3 incluyen el pseudo-código del algoritmo y el diagrama de flujo respectivamente.

```
Inicialmente:
   x=0
   lct=0
   aux=0
   t_a indica instante actual de tiempo

En cada llegada de paquete:
   aux=x-(t_a-lct)
   if aux<0
      {
         aux=0
         x=incremento
         lct=t_a
         marcar paquete como in
      }
   else if aux<ζ
      {
         x=incremento+aux
         lct=t_a
         marcar paquete como in
      }
   else
      marcar paquete como out
```

Fig. 2.2 Pseudo-código del algoritmo *Leaky Bucket*. Significado de las variables: *x* → contador del cubo licuante, *aux* → variable auxiliar, *lct* → último tiempo conforme (ultimo PAT conforme), ζ → tolerancia, I → incremento y t_a → instante de tiempo actual

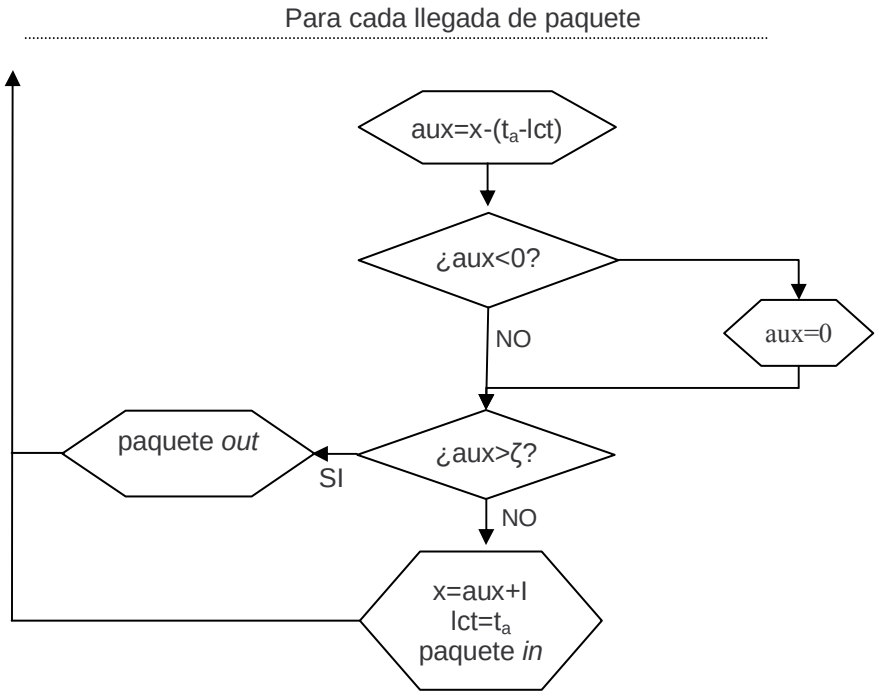


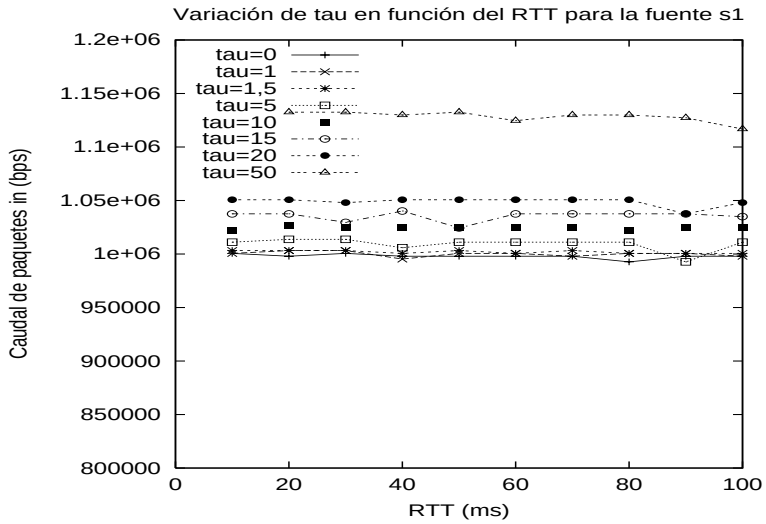
Fig. 2.3 Diagrama de flujo del algoritmo *Leaky Bucket*

2.3.1.1 Tasa de velocidad de paquetes *in*

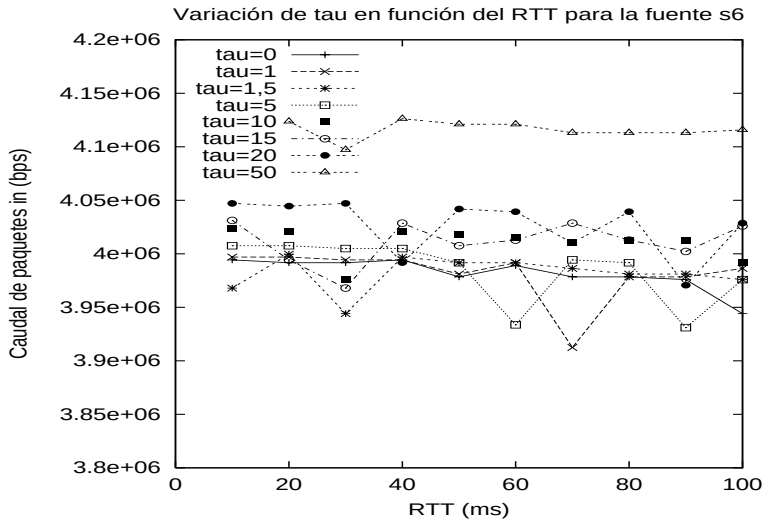
Las prestaciones de los tres acondicionadores en estudio dependerán de diversos parámetros. En este caso, vamos a evaluar el acondicionador *Leaky Bucket* variando los valores del parámetro ζ. Así, encontraremos el rango de valores de ζ que permiten garantizar los contratos de los usuarios con independencia del RTT o de los contratos.

De acuerdo con las simulaciones realizadas, para $0 \leq \zeta \leq 20$ es posible alcanzar unas buenas prestaciones, puesto que con los paquetes *in* cada fuente asegura o excede mínimamente su contrato. Como era lógico, a mayor ζ (desde 0 hasta 20) mayor es el caudal de paquetes *in*. Es decir, cuanto mayor es la *tolerancia* más paquetes caerán dentro del perfil. Gracias a esta propiedad en el funcionamiento de *Leaky Bucket*, es posible maquillar unas prestaciones de red deficientes incrementando el valor de ζ. Sin embargo, esta solución debe emplearse con precaución para evitar consumir más ancho de banda del destinado a cada fuente. En consecuencia, si no existen carencias en la red que deban ser subsanadas, los contratos quedan garantizados con un valor de ζ entre 0 y 10. La Fig. 2.4 muestra el caudal de paquetes *in* en bps (bits por segundo) en función del RTT (ms) para diferentes valores de ζ y suponiendo que todas las fuentes tienen el mismo RTT. En la Fig. 2.4 a) se muestran los resultados para la fuente s_1 y en la Fig. 2.4. b) los de la fuente s_6. Estas simulaciones que se muestran se realizaron en una topología de 8 fuentes de tráfico (s_0 a s_7), con contratos de 1, 1, 2, 2, 3, 3, 4 y 4 Mbps respectivamente.

Cuando todas las fuentes de tráfico tienen un RTT similar, los contratos se alcanzan independientemente del valor de aquel. Hay que señalar, que conforme mayores son los contratos mayores son las variaciones producidas, pero sin bajar un 2,5 % de los contratos para $\zeta < 20$. En el caso de que cada fuente tenga diferente RTT, los resultados son también satisfactorios, como se desprende de las Tablas 2.1 y 2.2. Si comparamos las columnas tercera y cuarta de estas tablas verificamos que el caudal de paquetes *in* cumple con los contratos fijados.



a)



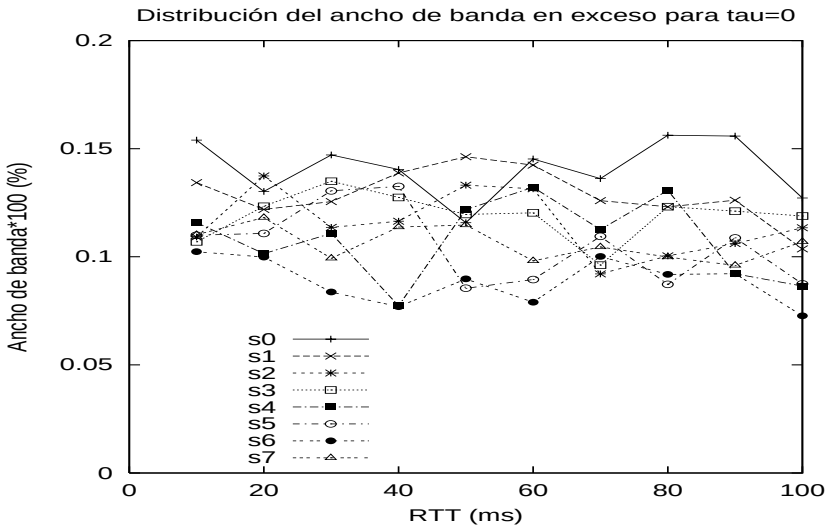
b)

Fig. 2.4 τ vs. RTT. La figura muestra el caudal de paquetes *in* para a) la fuente s_1 cuyo contrato es de 1 Mbps y b) la fuente s_6 cuyo contrato es de 4 Mbps. Todas las fuentes presentan un RTT similar pero contratos desiguales

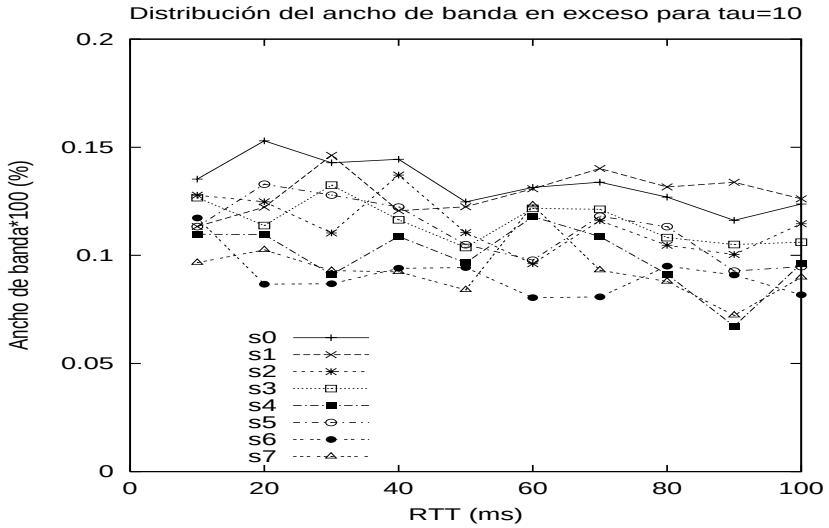
2.3.1.2 Distribución del ancho de banda excedente

En cuanto a la distribución del ancho de banda sobrante, podemos indicar que si todas las fuentes tienen el mismo RTT, el exceso se distribuye prácticamente de modo justo entre todas (véase la última columna de la Tabla 2.1). De las Fig. 2.5 a) y 2.5. b) se desprende la dependencia que existe entre la distribución del ancho de banda en exceso y ζ. En estas figuras se mantiene el mismo RTT para todas las fuentes, realizando simulaciones con valores de RTT desde 10 ms hasta 100 ms. En el caso ideal (cada fuente recibiendo la misma porción de ancho de banda en exceso) el gráfico sería una única línea, de manera que cuanto más pequeña sea el área del gráfico más cerca estamos del resultado óptimo. Vemos que la distribución es más imparcial para ζ = 10 que para ζ = 0. La Tabla 2.1 muestra los valores cuantitativos de las simulaciones con mismo RTT.

En el caso de que cada fuente tenga un RTT distinto, el reparto del exceso se vuelve más irregular. En concreto, las fuentes con menor RTT y menor contrato se ven favorecidas (observe la última columna de las Tablas 2.1 y 2.2). Nótese, que las partes que obtienen las distintas fuentes no varían más de un 12% en el caso peor y un 4% en el caso mejor.



a)



b)

Fig. 2.5 % del ancho de banda excedente que obtiene cada fuente en función del RTT, con a) ζ=0 y b) ζ=10

Tabla 2.1 Simulaciones con *Leaky Bucket* con el mismo RTT para cada fuente. (S ≡ fuente; RTT ≡ tiempo de ida y vuelta; C ≡contrato; BW exceso ≡ ancho de banda en exceso)

Simulación con ζ =0					
S	**RTT (ms)**	**C (Mbps)**	**Caudal in (Mbps)**	**Caudal total (Mbps)**	**BW exceso (Mbps)**
0	50	1	0,99	2,50	1,51 (11,6%)
1	50	1	1,00	2,90	1,90 (14,7%)
2	50	2	1,98	3,73	1,75 (13,5%)
3	50	2	1,99	3,55	1,56 (12,0%)
4	50	3	2,99	4,58	1,59 (12,3%)
5	50	3	2,99	4,11	1,12 (8,6%)
6	50	4	3,98	5,17	1,19 (9,1%)
7	50	4	3,99	5,49	1,50 (11,5%)

Simulación con ζ =10					
S	**RTT (ms)**	**C (Mbps)**	**Caudal *in* (Mbps)**	**Caudal total (Mbps)**	**BW exceso (Mbps)**
0	50	1	1,02	2,82	1,80 (13,8%)
1	50	1	1,02	2,79	1,77 (13,6%)
2	50	2	1,99	3,64	1,65 (12,6%)
3	50	2	2,02	3,45	1,43 (11,1%)
4	50	3	3,02	4,35	1,33 (10,3%)
5	50	3	3,03	4,46	1,43 (11,1%)
6	50	4	4,02	5,33	1,31 (10,0%)
7	50	4	4,02	5,20	1,18 (9,1%)

Tabla 2.2 Simulaciones con *Leaky Bucket* con distintos RTT cada fuente. (S ≡ fuente; RTT ≡ tiempo de ida y vuelta; C ≡contrato; BW exceso ≡ ancho de banda en exceso)

Simulación con ζ =0					
S	**RTT (ms)**	**C (Mbps)**	**Caudal *in* (Mbps)**	**Caudal total (Mbps)**	**BW exceso (Mbps)**
0	10	1	1,00	3,78	2,78 (21,5%)
1	20	1	0,99	2,99	2,00 (15,4%)
2	30	2	1,99	3,93	1,94 (14,9%)
3	40	2	1,99	3,47	1,48 (11,4%)
4	50	3	2,99	4,35	1,36 (10,5%)
5	60	3	2,99	3,93	0,94 (7,2%)
6	70	4	3,98	4,92	0,94 (7,2%)
7	80	4	3,94	4,72	0,78 (5,9%)

Simulación con ζ =10					
S	**RTT (ms)**	**C (Mbps)**	**Caudal *in* (Mbps)**	**Caudal total (Mbps)**	**BW exceso(Mbps)**
0	10	1	1,00	3,72	2,72 (20,9%)
1	20	1	0,99	2,94	1,95 (15,1%)
2	30	2	2,00	3,87	1,87 (14,4%)
3	40	2	1,99	3,47	1,48 (11,4%)
4	50	3	2,99	4,50	1,51 (11,6%)
5	60	3	2,98	3,88	0,90 (6,9%)
6	70	4	3,99	4,84	0,85 (6,5%)
7	80	4	3,99	4,85	0,86 (6,7%)

Para resumir el comportamiento de *Leaky Bucket*, a partir de los resultados obtenidos, todas las fuentes están ligeramente por encima de sus respectivos contratos y la distribución del ancho de banda en exceso es bastante regular si el RTT es el mismo para todas las fuentes y se usa un ζ

= 10. Fíjese que para situaciones extremas en las que las fuentes de tráfico multiplexadas tienen un RTT muy dispar (diferencias entre fuentes por encima de 60 ms aproximadamente), el caudal de paquetes *in* asegura los contratos pero el reparto del ancho de banda no contratado no es justo. Este inconveniente puede deberse a las propias características de las conexiones TCP, dado que para aquellas conexiones con RTT más alto transcurre más tiempo antes de recibir los reconocimientos, lo que además aumenta el tiempo de detección de pérdida de un paquete y retransmisión del mismo.

2.3.2. Algoritmo de ventana deslizante temporal o *Time Sliding Window*

En este punto vamos a analizar el algoritmo acondicionador de tráfico introducido por Clark y Fang en [CLARK93], denominado algoritmo de ventana de deslizante TSW. Este algoritmo consta de dos partes, un estimador de tasa y un algoritmo de marcado. El estimador de tasa obtiene, como su nombre indica, una estimación de la tasa de velocidad de la fuente en cada llegada de un paquete y mantiene un intervalo de historia pasada denominado *winlength*. *Winlength* es un parámetro constante configurado inicialmente. La implementación del estimador queda descrita en la Fig. 2.6. Según se indica en [CLARK93], un acondicionador ideal debería mantener una conexión TCP oscilando desde 0,66 a 1,33 veces su velocidad contratada, de manera que en media la conexión alcance el contrato establecido.

```
Inicialmente:
   winlength=una constante
   tasa_media=tasa contratada de la fuente;
   t_front=0;

En cada llegada de un paquete:
      bytes_en_TSW=tasa_media*winlength
      nuevos_bytes=bytes_en_TSW+tamaño_paq
      tasa_media=nuevos_bytes/(ahora-t_front+winlength)
      t_front=ahora
```

Fig. 2.6 Implementación de TSW

Cuando la tasa media estimada supera un cierto umbral, el acondicionador TSW marca los paquetes como *out* con una probabilidad *p* (ec. 2.1). En este caso se pueden utilizar dos métodos diferentes. El primero tiene una memoria mayor y los paquetes se marcan como *out* con probabilidad *p* cuando la media excede el contrato. El segundo método tiene una memoria menor y los paquetes se marcan como *out* con probabilidad *p* cuando la estimación media excede β veces el contrato, donde β es un número real mayor que 1 (el caso $\beta = 1$ se corresponde con el primer método). En general, TSW es muy sensible a este umbral β debido al mecanismo de control de la congestión de TCP, que es el que determina la tasa de envío instantánea de la conexión.

$$p = \frac{tasa\ media\ estimada - contrato}{tasa\ media\ estimada} \tag{2.1}$$

En las simulaciones optaremos por el segundo método, pues es el recomendado por sus creadores. Aunque TSW ha recibido muchas críticas en algunos artículos como [IBANEZ98], y algunas de las mejoras propuestas presentan problemas de escalabilidad [LIN99], intentaremos a lo largo de este apartado conocer mejor las prestaciones que ofrece.

2.3.2.1 Dependencia entre los valores de β y *winlength*

La primera pregunta a la que debemos hallar respuesta en TSW es qué valores de β y *winlength* son los más adecuados para alcanzar unas mejores prestaciones. Para ello

estudiaremos el efecto que estos parámetros tienen en el caudal de paquetes *in*, es decir, los paquetes que nos deben garantizar el contrato. En las simulaciones empleamos la misma topología de ocho fuentes TCP que utilizamos en las secciones 2.3.1.1 y 2.3.1.2.

Tras realizar las primeras simulaciones pudimos observar hechos comunes en todas ellas:

a) Conforme mayor es el tamaño de la ventana *winlength* mayor es el número de paquetes que se marcan como *in*.

b) Conforme mayor es β mayor es el número de paquetes marcados como *in*.

Los puntos a) y b) son demasiado genéricos. Los caudales de paquetes *in* que se consiguen mediante simulación muestran desviaciones irregulares alrededor de los contratos. Estas irregularidades pueden deberse por ejemplo a los diferentes RTT o a la naturaleza a ráfagas del tráfico TCP. Lo que hace muy complicado averiguar los valores más adecuados para los parámetros *winlength* y β. Las Fig. 2.7 y 2.8 reflejan los caudales de paquetes *in* de las fuentes s_0 y s_6 para diferentes valores de *winlength* y β. Fijándonos en estas figuras resulta complejo saber qué valores escoger, puesto que los que son buenos para un contrato no lo son para otros.

c) Se produce una relativa estabilidad en los caudales de paquetes *in* para tamaños de ventana 160 ms ≤ *winlength* ≤ 200 ms y 1 < β ≤ 2, lo que permite asegurar los contratos.

Este punto nos impulsó a ejecutar nuevas simulaciones, variando en este caso el número de fuentes y sus contratos. Seleccionamos un tamaño de ventana *winlength* = 200 ms (zona estable según el punto c)) y aplicamos distintos RTT a cada fuente. A partir de estas simulaciones, encontramos un método sencillo para seleccionar el valor de β en función del contrato de la fuente. La Fig. 2.9 muestra los valores recomendados de β para RTT por debajo de 50 ms. En la Tabla 2.3 presentamos un ejemplo con una topología de cuatro fuentes, donde podemos observar el caudal obtenido de paquetes *in* cuando seguimos las indicaciones de la Fig. 2.9.

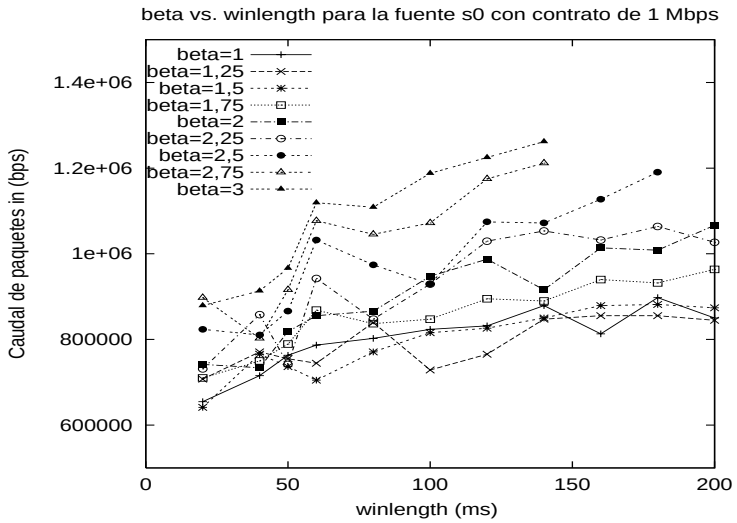


Fig. 2.7 β vs. *winlength*

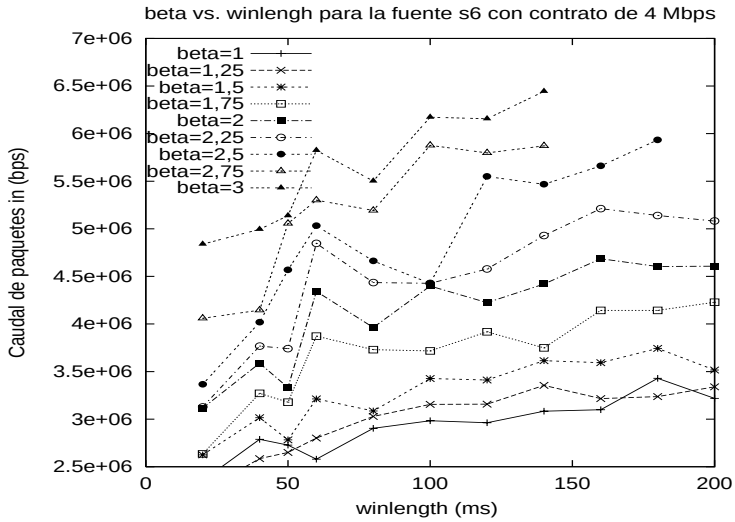


Fig. 2.8 β vs. *winlength*

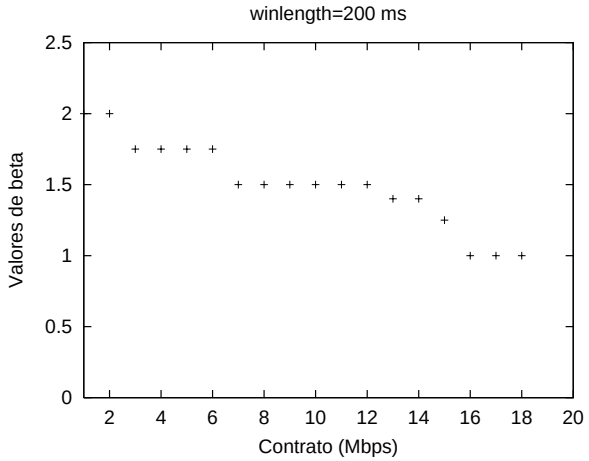


Fig. 2.9 Guía de configuración de valores de β

Tabla 2.3 TSW con los valores β según la Fig. 2.9 y *winlength* = 200 ms (S ≡ fuente; RTT ≡ tiempo de ida y vuelta; C ≡contrato)

Primer caso: distintos RTT (retardos pequeños para fuentes con menores contratos)				
S	**RTT (ms)**	**β**	**C (Mbps)**	**Caudal *in* (Mbps)**
0	10	2	1	1,00
1	20	1,75	4	4,36
2	30	1,5	8	8,39
3	40	1,4	14	13,74

Segundo caso: distintos RTT (retardos pequeños para fuentes con mayores contratos)				
S	**RTT (ms)**	**β**	**C (Mbps)**	**Caudal *in* (Mbps)**
0	40	2	1	1,02
1	30	1,75	4	4,50
2	20	1,5	8	8,45
3	10	1,4	14	14,56

Con el objetivo de comprobar el efecto de no seleccionar β según se indica en la Fig. 2.9, llevamos a cabo nuevas simulaciones. En este caso, la topología consta de ocho fuentes s_0 a s_7 con contratos de 1, 1, 2, 2, 3, 3, 4 y 4 Mbps respectivamente. La Fig. 2.10 representa el caudal de paquetes *in* para la fuente s_7 con un tamaño de ventana *winlength* = 30 ms y β = 2,5. La Fig. 2.11 muestra el caudal obtenido de paquetes *in* para la fuente s_7 con *winlength* igual a 200 ms y β igual a 1,75 como indica la guía de la Fig. 2.9. Observe la gran diferencia que existe entre ambas figuras. Claramente obtenemos mejores prestaciones cuando seguimos las indicaciones de configuración señaladas anteriormente.

También se desprende de las simulaciones el hecho de que conforme el RTT es mayor (RTT > 40 ms aproximadamente), el caudal de paquetes *in* desciende de manera abrupta en fuentes con contratos elevados (>12 Mpbs). El efecto contrario se produce para fuentes con contratos pequeños, pero en este caso el incremento no es tan abultado. Así, convenimos que para posteriores simulaciones donde se produjesen estas circunstancias sería necesario modificar los valores de β con un factor de corrección de +0,25. Como ejemplo observe la Tabla 2.4, donde la topología consta de dos fuentes s_0 y s_1 con contratos de 1 y 16 Mbps respectivamente. Es obvio que las prestaciones mejoran al aplicar el factor de corrección.

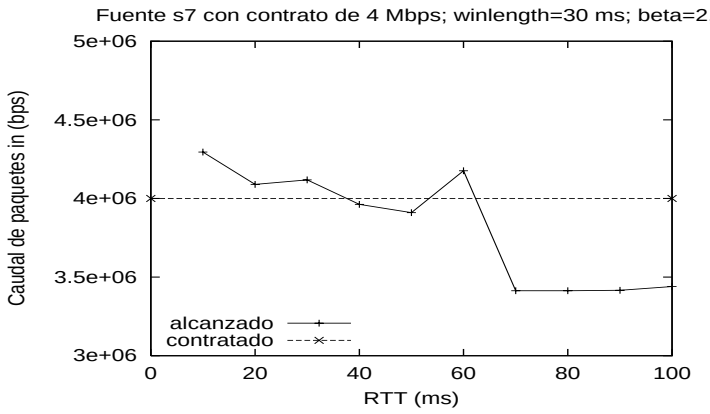


Fig. 2.10 Ejemplo de una configuración errónea

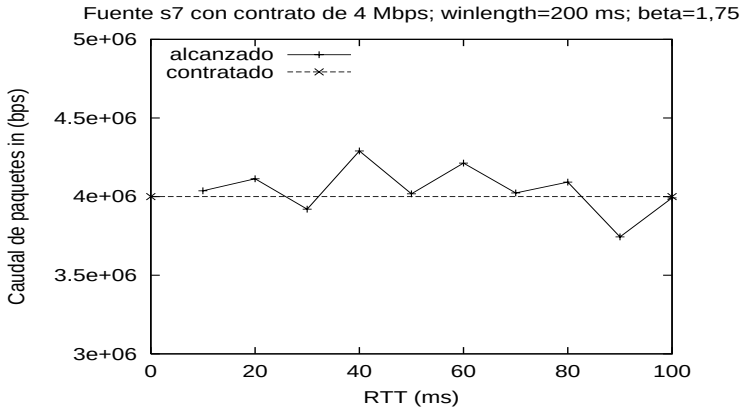


Fig. 2.11 Ejemplo de una configuración correcta

Tabla 2.4 Necesidad del factor de corrección de β por la influencia del RTT. En el primer caso a) RTT > 40 ms y el contrato de la fuente s_1 es mayor de 12 Mbps y puesto que no se aplica el factor de corrección la fuente s_1 no garantiza su contrato. En b) se muestra el primer caso pero aplicando el factor de corrección, claramente mejora el resultado. En c) no es necesario aplicar el factor de corrección puesto que RTT > 40 ms pero el contrato es de 1 Mbps

a)

S	RTT (ms)	β	C (Mbps)	Caudal *in* (Mbps)
0	10	2	1	1,10
1	50	1	16	13,12

b)

S	RTT (ms)	β	C (Mbps)	Caudal *in* (Mbps)
0	10	2	1	1,13
1	50	1,25	16	16,42

c)

S	RTT (ms)	β	C (Mbps)	Caudal *in* (Mbps)
0	50	2	1	1,05
1	10	1	16	15,93

2.3.2.2 Tasa de velocidad de paquetes *in*

Una vez resuelto el problema de la configuración de parámetros de TSW, vamos a analizar en primer lugar las prestaciones en cuanto a garantías de alcanzar los contratos establecidos. Utilizaremos la misma topología que en las secciones 2.3.1.1 y 2.3.1.2: ocho conexiones TCP con contratos de 1, 1, 2, 2, 3, 3, 4 y 4 Mbps respectivamente. El tamaño de ventana *winlength* queda fijado a 200 ms y β igual a 2 para las fuentes s_0 a s_3, e igual a 1,75 para las fuentes s_4 a s_7 (véase Fig. 2.9). Compararemos los resultados de las simulaciones con los obtenidos mediante el algoritmo *Leaky Bucket*.

Hemos dibujado los caudales de paquetes *in* en función de varios RTT para las fuentes s_1 y s_6 (Fig. 2.12 y 2.13 respectivamente). En estas gráficas todas las fuentes tienen el mismo valor de RTT. Se observa que el caudal es mucho más irregular con TSW que con *Leaky Bucket*. Las prestaciones se mantienen al mismo nivel cuando cada conexión tiene un RTT diferente (véanse Tablas 2.5 y 2.6). Como primera conclusión afirmamos que *Leaky Bucket* ofrece un mejor servicio.

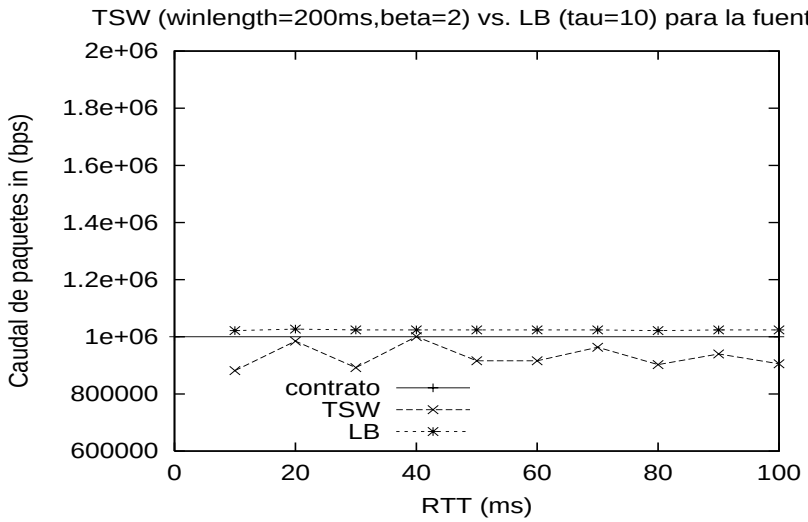


Fig. 2.12 Caudal de paquetes *in* de la fuente s_1

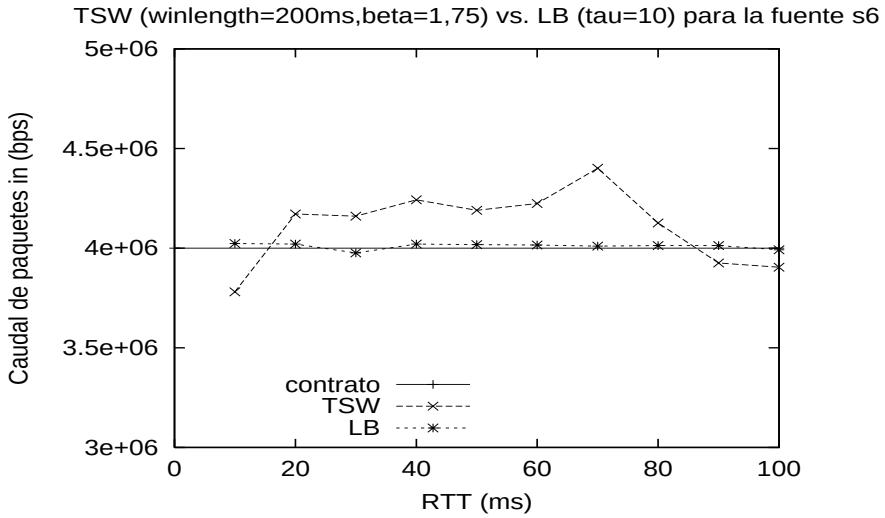


Fig. 2.13 Caudal de paquetes *in* de la fuente s_6

Tabla 2.5 Comportamiento de TSW con el mismo RTT todas las fuentes

S	RTT (ms)	β	C (Mbps)	Caudal *in* (Mbps)	Caudal total (Mbps)	BW exceso (Mbps)
0	50	2	1	1,00	2,65	1,65 (12,7%)
1	50	2	1	0,92	2,77	1,85 (14,2%)
2	50	2	2	2,04	3,46	1,42 (10,9%)
3	50	2	2	2,03	3,26	1,23 (9,5%)
4	50	1,75	3	2,96	4,13	1,17 (9,0%)
5	50	1,75	3	2,94	4,03	1,09 (8,4%)
6	50	1,75	4	4,19	5,38	1,19 (9,1%)
7	50	1,75	4	4,02	4,99	0,97 (7,5%)

Tabla 2.6 Comportamiento de TSW cuando cada fuente presenta un RTT diferente

S	RTT (ms)	β	C (Mbps)	Caudal *in* (Mbps)	Caudal total (Mbps)	BW exceso (Mbps)
0	10	2	1	0,99	2,72	1,73 (13,3%)
1	20	2	1	0,82	2,59	1,77 (13,7%)
2	30	2	2	2,05	3,76	1,71 (13,1%)
3	40	2	2	2,06	3,44	1,38 (10,7%)
4	50	1,75	3	2,88	4,47	1,59 (12,2%)
5	60	1,75	3	2,91	4,28	1,37 (10,5%)
6	70	1,75	4	3,78	4,91	1,13 (8,7%)
7	80	1,75	4	4,04	5,39	1,35 (10,4%)

S	RTT (ms)	β	C (Mbps)	Caudal *in* (Mbps)	Caudal total (Mbps)	BW exceso (Mbps)
0	80	2	1	1,01	1,95	0,94 (7,3%)
1	70	2	1	0,98	2,10	1,12 (8,6%)
2	60	2	2	2,10	3,25	1,15 (8,9%)
3	50	2	2	2,17	3,23	1,06 (8,1%)
4	40	1,75	3	3,17	4,51	1,34 (10,3%)
5	30	1,75	3	3,17	4,41	1,24 (9,5%)
6	20	1,75	4	4,54	6,02	1,48 (11,4%)
7	10	1,75	4	4,27	6,21	1,94 (14,9%)

2.3.2.3 Distribución del ancho de banda excedente

Queda por examinar cómo se realiza el reparto del ancho de banda en exceso cuando se utiliza el acondicionador de tráfico TSW. Según los resultados de las simulaciones podemos distinguir dos casos: el primero es el que se produce en una topología donde todas las fuentes tienen un RTT similar, y el segundo es aquel en el que cada fuente tiene un RTT distinto. En el primer caso (RTT homogéneo), los resultados son peores que los obtenidos con *Leaky Bucket*. Esto significa que *Leaky Bucket* realiza una distribución más equitativa del ancho de banda en exceso (compare las Tablas 2.1 y 2.5). En el segundo caso, la distribución tiende a ser más justa con TSW que con *Leaky Bucket* (compare las Tablas 2.2 y 2.6).

Será necesario por tanto un compromiso entre garantizar estrictamente los contratos o repartir el ancho de banda en exceso de modo más justo cuando el RTT difiera entre las distintas fuentes. Desde nuestro punto de vista, optamos por emplear *Leaky Bucket* si los RTT de las fuentes no son excesivamente distintos, principalmente por la mayor complejidad de configuración que TSW presenta.

2.4. Algoritmo basado en contadores *Counters Based*

El tercer acondicionador de tráfico que vamos a discutir en este capítulo es el basado en contadores, al que hemos denominado *Counters Based* (CB). El algoritmo que proponemos presenta la ventaja de no tener apenas parámetros de configuración, como sí ocurre con LB o TSW, lo que lo aumenta su facilidad de implementación.

El algoritmo basa su funcionamiento en dos contadores, llamémosles C_1 y C_2. C_1 se inicializa a cero para que el primer paquete que llega al acondicionador se marque siempre como *in*. Por su parte, C_2 toma como valor inicial la inversa del contrato. Cada unidad de tiempo, C_2 decrece una unidad. Si C_2 es cero, entonces incrementamos C_1 una unidad y C_2 vuelve a su valor inicial. Cuando se produce la llegada de un paquete, comprobamos el valor de C_1. Si C_1 es mayor que cero, entonces el paquete se marca como *in* y restamos una unidad a C_1. Si no, el paquete se

marca como *out*. Las Fig. 2.14 y 2.15 muestran el pseudo-código del algoritmo CB y su diagrama de flujo.

```
Inicialmente:
   C1=1
   C2=velocidad enlace/contrato
Para cada unidad de tiempo:
   C2--
   if C2<=0
      {
         C1++
         C2=velocidad enlace/contrato;
      }
   if llega un paquete
      if C1>0
         marcar paquete como in
      else
         marcar paquete como out
```

Fig. 2.14 Pseudo-código del algoritmo *Counters Based*

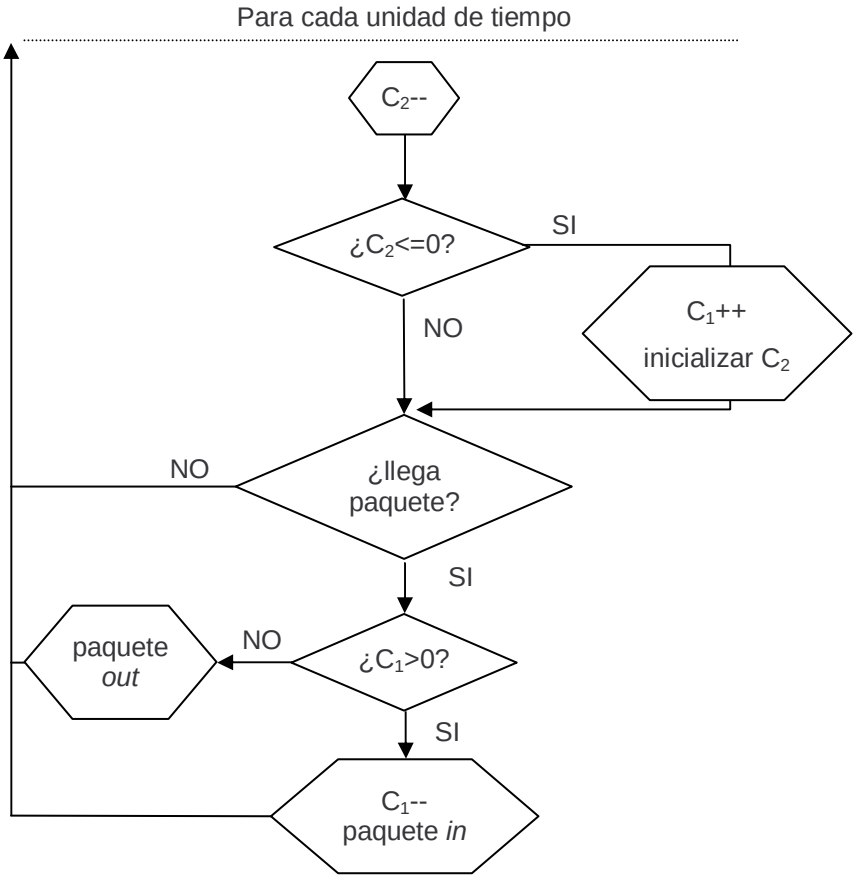


Fig. 2.15 Diagrama de flujo del algoritmo *Counters Based*

2.4.1. Tasa de velocidad de paquetes *in*

Las simulaciones realizadas con el algoritmo CB revelan que los caudales de paquetes *in* garantizan de modo estricto los contratos, con variaciones que apenas llegan al 1%. En las Fig.

2.16 a 2.18 se puede apreciar la gran exactitud con la que se aseguran los contratos empleando CB. En estas figuras se ha representado el caudal de paquetes *in* de las fuentes s_1, s_4 y s_6. Las simulaciones se realizaron variando el RTT en las distintas simulaciones pero manteniéndolo igual para todas las fuentes. La topología empleada es la indicada en la Sección 2.2.

Según los resultados obtenidos en simulaciones con distinto RTT para cada fuente, CB mantiene unas garantías de contratos del 99%. Si comparamos los resultados con los procedentes de LB y TSW, queda manifiesta la superioridad de CB en términos de garantías de contrato.

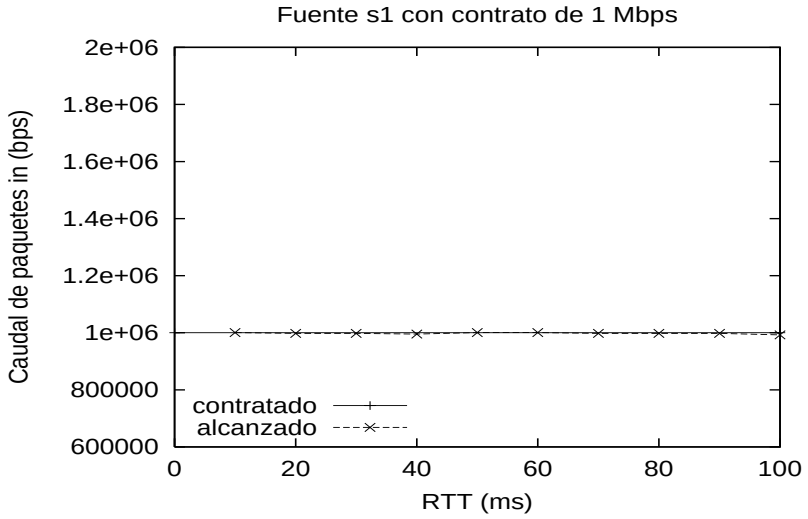


Fig. 2.16 Caudal alcanzado de paquetes *in* por la fuente s_1 (contrato de 1 Mbps)

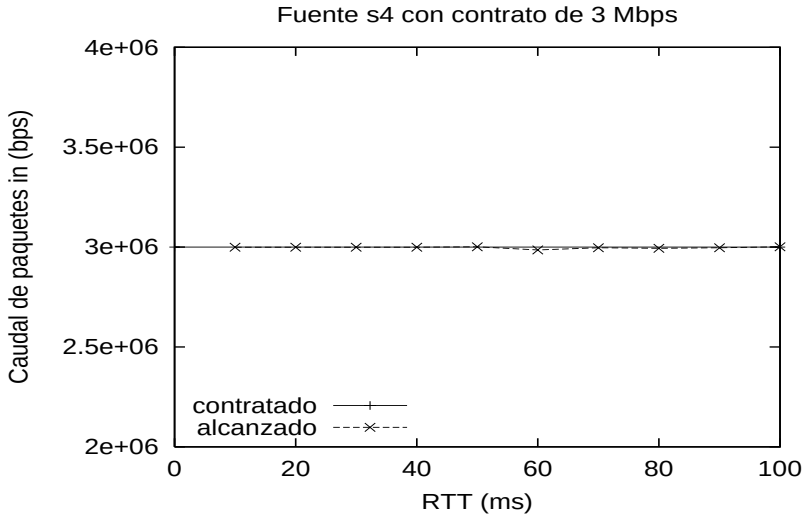


Fig. 2.17 Caudal alcanzado de paquetes *in* por la fuente s_4 (contrato de 3 Mbps)

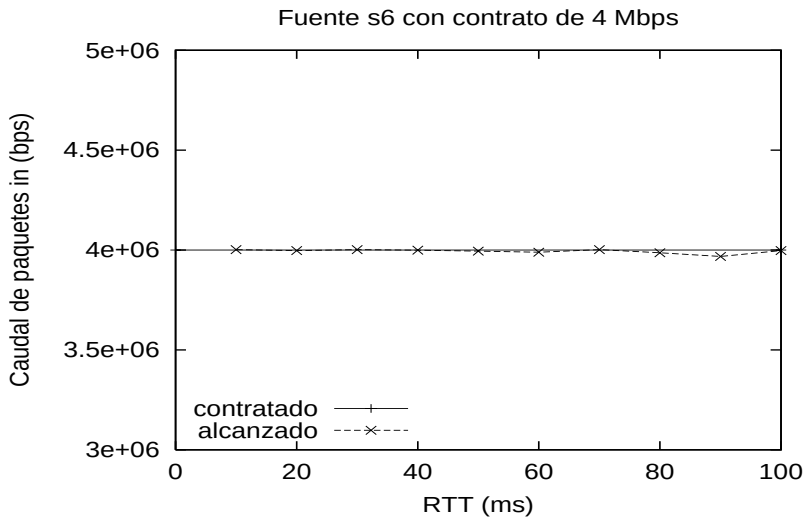


Fig. 2.18 Caudal alcanzado de paquetes *in* por la fuente s_6 (contrato de 4 Mbps)

2.4.2. Distribución del ancho de banda excedente

Resta por estudiar cómo CB realiza el reparto del ancho de banda excedente. Cuando todas las fuentes tienen un RTT similar, el exceso de ancho de banda se reparte de modo equitativo entre las fuentes (véase la Fig. 2.19). Los resultados son incluso más justos que los obtenidos con LB (para RTT < 50 ms). En el caso peor, la máxima diferencia en el reparto de ancho de banda entre las distintas fuentes es de un 12%, mientras que disminuye hasta un 4% en el caso mejor. Para comparar la justicia de CB con la de LB y TSW contrastamos las Tablas 2.1, 2.5 y 2.7. En ellas observamos que CB hace una distribución más justa.

No obstante, en la Tabla 2.8 se aprecia que hay una clara carencia de justicia cuando las fuentes tienen RTT distintos como ocurría con LB y TSW (donde TSW respondía mejor bajo la apropiada configuración). Este hecho se desprende de las Tablas 2.2, 2.6 y 2.8. Los flujos de tráfico con RTT más pequeños se ven claramente favorecidos, recibiendo además más ancho de banda dentro de este conjunto las fuentes con menores contratos.

Con los resultados obtenidos determinamos que no es posible, con los acondicionadores de tráfico conocidos hasta la fecha, realizar un reparto equitativo de los recursos no contratados. En consecuencia, al menos deberemos disponer de un acondicionador que garantice los contratos de los usuarios. Como queda demostrado en este capítulo, el algoritmo basado en contadores *Counters Based* es el que mejor se adapta a este objetivo.

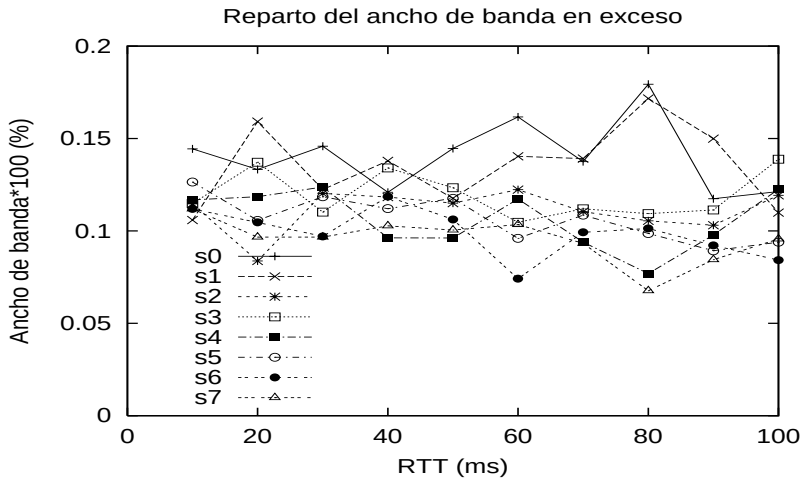


Fig. 2.19 Distribución del ancho de banda en exceso con el algoritmo *Counters Based* para una topología donde todas las fuentes tienen RTT similares

Tabla 2.7 Resultados de simulaciones con el algoritmo *Counters Based* con RTT similares para cada fuente

S	RTT (ms)	C (Mbps)	Caudal *in* (Mbps)	Caudal total (Mbps)	BW exceso (Mbps)
0	50	1	1,00	2,88	1,88 (14,45%)
1	50	1	1,00	2,53	1,53 (11,76%)
2	50	2	2,00	3,50	1,50 (11,51%)
3	50	2	2,00	3,60	1,60 (12,35%)
4	50	3	3,00	4,25	1,25 (9,60%)
5	50	3	3,00	4,53	1,53 (11,76%)
6	50	4	3,99	5,38	1,39 (10,67%)
7	50	4	3,99	5,31	1,32 (10,09%)

Tabla 2.8. Resultados de simulaciones con el algoritmo *Counters Based* con distintos RTT para cada fuente

S	RTT (ms)	C (Mbps)	Caudal *in* (Mbps)	Caudal total (Mbps)	BW exceso (Mbps)
0	10	1	1,00	3,83	2,83 (21,80%)
1	20	1	1,00	3,21	2,21 (16,99%)
2	30	2	2,00	3,89	1,89 (14,53%)
3	40	2	2,00	3,52	1,52 (11,74%)
4	50	3	3,00	4,26	1,26 (9,67%)
5	60	3	3,00	3,99	0,99 (7,65%)
6	70	4	4,00	4,80	0,80 (6,19%)
7	80	4	3,97	4,68	0,71 (5,48%)

S	RTT (ms)	C (Mbps)	Caudal *in* (Mbps)	Caudal total (Mbps)	BW exceso (Mbps)
0	80	1	1,00	2,14	1,14 (8,79%)
1	70	1	1,00	2,64	1,64 (12,61%)
2	60	2	2,00	3,19	1,19 (9,16%)
3	50	2	1,98	3,22	1,24 (9,58%)
4	40	3	3,00	4,28	1,28 (9,83%)
5	30	3	3,00	4,80	1,80 (13,89%)
6	20	4	4,00	5,71	1,71 (13,14%)
7	10	4	4,00	6,39	2,39 (18,38%)

2.5. Conclusiones

En este capítulo hemos realizado un estudio comparativo mediante simulaciones de diferentes acondicionadores de tráfico. El objetivo de estos acondicionadores es ayudar a alcanzar las metas del servicio asegurado AF. Para las simulaciones se han empleado fuentes de tráfico TCP Reno. En este estudio se ha explorado tanto la configuración más idónea de los acondicionadores clásicos TSW y *Leaky Bucket*, como las prestaciones del nuevo acondicionador de tráfico basado en contadores *Counters Based* presentado en este capítulo.

Leaky Bucket ofrece unas prestaciones adecuadas en términos de garantías de contrato de usuarios finales para valores de ζ en el intervalo [0,10]. En función de cuán permisivo queremos que sea el caudal de paquetes *in* escogeremos $\zeta = 0$ (más estricto) o $\zeta = 10$ (más tolerante). En este último caso, las fuentes no sólo estarán por encima de su contrato con el caudal de paquetes *in* sino que la distribución del ancho de banda en exceso será más justa que con $\zeta = 0$. Refiriéndonos a esta distribución del ancho de banda excedente, es bastante equitativa en topologías donde las fuentes tienen RTT similares. En situaciones más dispares, donde las conexiones tienen RTT muy diferentes, las fuentes con mayores contratos y RTT más elevados alcanzan los contratos con sus respectivos caudales de paquetes *in*, pero el reparto del ancho de banda que sobra es injusto. Este inconveniente puede deberse a las características de las conexiones TCP, al funcionamiento del acondicionador seleccionado y/o a la interacción de éste con la política RIO.

En relación con el acondicionador TSW, en este capítulo mostramos cómo seleccionar para cada fuente los valores de β y *winlength* en función del contrato. Siguiendo esta guía de configuración, las prestaciones de TSW mejoran notablemente en cuanto a disponibilidad de los contratos. Sin embargo, TSW presenta más variaciones que *Leaky Bucket* o CB tanto en topologías donde las fuentes tienen RTT similar como en topologías con RTT diferentes. Con respecto al reparto del ancho de banda en exceso, TSW establece una entrega más justa que *Leaky Bucket* o CB cuando los RTT difieren entre las distintas fuentes. No obstante, debe tenerse en cuenta la compleja implementación de TSW, siendo éste un factor de decisión importante a la hora de escoger un algoritmo u otro.

La nueva propuesta en acondicionadores de tráfico que hemos presentado se denomina *Counters Based* o algoritmo basado en contadores. Este algoritmo basa su uso en dos contadores configurados a partir del contrato de cada fuente. Como el contrato es un valor preestablecido a priori, CB no necesita sintonización de parámetros, a diferencia de TSW o *Leaky Bucket*. Las prestaciones que ofrece son notables en términos de garantías de los contratos de los usuarios. Con referencia a la distribución del ancho de banda excedente, CB se presenta como el más justo de los tres acondicionadores estudiados cuando todas las fuentes tienen RTT similares, no siendo así para conexiones con RTT dispares. Aún así, por su gran simplicidad y exactitud a la hora de garantizar los contratos, consideramos que es un mecanismo a considerar en futuros trabajos sobre Servicios Diferenciados y en concreto para el servicio asegurado AF.

Capítulo 3
Counters Based Dual Queuing: Propuesta alternativa para la diferenciación de servicios

3.1. Introducción

En la arquitectura DiffServ se definen varios comportamientos por salto o PHB. Entre ellos, destaca el AF PHB, que ofrece un servicio Asegurado que garantiza el ancho de banda contratado por el usuario y en el caso de existir ancho de banda no contratado lo reparte entre las distintas fuentes que componen el agregado. Hasta la fecha, ninguna propuesta de implementación ha podido cumplir ambos objetivos.

Un servicio asegurado se construye con acondicionadores de tráfico y esquemas de gestión de colas para el PHB. El primer algoritmo de gestión de cola que se utilizó en DiffServ fue RIO [CLAR98]. RIO (véase Capítulo 1 Sección 1.2) es una combinación de dos algoritmos RED [FLOY93] y ha sido empleado como opción clásica de gestión de cola para implementar el servicio asegurado. Varios estudios han empleado RIO para analizar las prestaciones de diversos acondicionadores de tráfico [NAND00][KIM99][ANDR00][LIN99]. Las distintas propuestas sólo consiguieron asegurar los contratos, dejando la cuestión del reparto del ancho de banda sin resolver. Así, posteriores trabajos se centraron en hacer modificaciones a RIO [LIN99][ELLO99][PARK00] para intentar realizar un reparto justo del ancho de banda excedente. Sin embargo, conforme las propuestas presentaban mejores resultados la complejidad iba en aumento, en contra de la tendencia simplista de DiffServ.

En este capítulo introducimos un nuevo enfoque que asegure a las conexiones TCP justicia (distribución equitativa) en el reparto del ancho de banda en exceso así como el caudal contratado. La idea planteada se basa en utilizar un esquema de gestión en el enrutador diferente del tradicional RIO. Además del esquema de gestión se empleará el mecanismo CB (explicado en el Capítulo 2) para el acondicionado del tráfico, dando lugar al mecanismo de gestión de doble cola (*Counters Based Dual Queuing*, CBDQ). CBDQ garantiza los caudales contratados por las conexiones TCP y reparte el ancho de banda no contratado de modo equitativo entre las fuentes. La novedad de CBDQ subyace en eliminar posibles interferencias entre paquetes *in* y *out*, lo que se consigue situando paquetes *in* y *out* en colas distintas servidas con un algoritmo de planificación adecuado. Como se muestra en los resultados de simulación, es posible garantizar los objetivos de un servicio asegurado. Las prestaciones de las conexiones TCP se analizarán en términos de *throughput* sin considerar los paquetes retransmitidos. En este capítulo demostraremos que con el esquema CBDQ las fuentes TCP garantizan sus respectivos contratos y se realiza una distribución justa del ancho de banda no contratado.

El resto del capítulo se organiza de la siguiente manera. La Sección 3.2 detalla las características de la nueva propuesta CBDQ. En la Sección 3.3 presentamos el escenario de simulación. A continuación en la Sección 3.4 se discuten los resultados obtenidos. El capítulo concluye en la Sección 3.5 con las conclusiones más relevantes.

3.2. El esquema de gestión de doble cola CBDQ

La gestión de doble cola aparece con el propósito de sustituir al esquema RIO tradicional. Al igual que en RIO, la idea básica es dar un trato diferente a los paquetes *in* y *out*. Sin embargo, si ambos tipos de paquete comparten la misma cola en el enrutador, ambos tipos de tráfico se interfieren, lo que complica poder conseguir una asignación justa del ancho de banda no contratado. Este efecto ha sido ampliamente estudiado en redes ATM con el servicio de velocidad de trama garantizada (*Guaranteed Frame Rate*, GFR) [CERD00b][BONA97]. El servicio GFR de ATM ofrece unas prestaciones similares al servicio asegurado AF. Para eliminar cualquier tipo de interferencia entre paquetes *in* y *out*, CBDQ encola y sirve los paquetes *in* y *out* desde dos colas diferentes. Véase la Fig. 3.1.

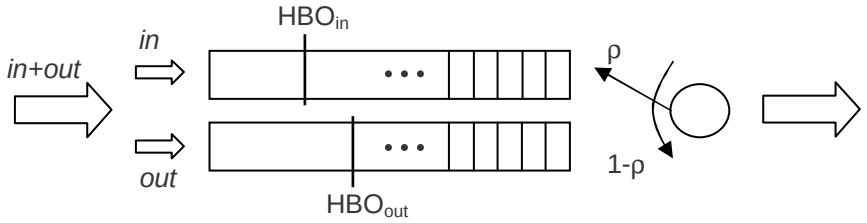


Fig. 3.1 Esquema de gestión del cola de doble cola CBDQ

A la hora de seleccionar la cola que ha de servirse se utilizará un algoritmo de planificación apropiado. Para disminuir la complejidad, el algoritmo de planificación seleccionado sirve ambas colas con una política *round-robin*. El planificador sirve la cola de paquetes *in* con una probabilidad ρ equivalente a la carga de la red (ec. 3.1 donde *n* es el número total de fuentes que componen el agregado). La cola de paquetes *out* se sirve con probabilidad complementaria 1-ρ. Si el planificador visita una cola vacía conmutará a la otra con el fin de tener un sistema de trabajo conservativo. Además, cada cola tendrá un límite denominado ocupación alta de la cola (*High Buffer Occupancy*, HBO) que delimita el número máximo de paquetes que se pueden almacenar y a partir del cual se aplica *Tail Drop*.

$$\rho = \frac{\sum_{i=1}^{n} tasa\ contratada}{capacidad\ enlace} \tag{3.1}$$

Este nuevo esquema de gestión de cola se puede emplear en combinación con cualquiera de los acondicionadores de tráfico estudiados en el Capítulo 2. En este caso, el estudio de prestaciones lo hemos realizado empleando CB como mecanismo de acondicionado del tráfico. Es importante remarcar que el sistema propuesto es una alternativa al estándar AF PHB. En la descripción del AF PHB [RFC2597] se indica claramente que los paquetes pertenecientes a un mismo flujo de tráfico se deben servir en el mismo orden en el que llegan al nodo de la red. Así, desde nuestro punto de vista, el estándar intenta proponer una implementación de los nodos interiores de dificultad mínima. En nuestra propuesta, el uso de una segunda cola para los paquetes *out* puede resultar en paquetes servidos fuera de orden. A pesar de este hecho, es evidente que TCP soluciona implícitamente cualquier pérdida de secuencia en el orden de los paquetes. De la misma forma, con la tecnología actual el uso de dos colas no supone una dificultad importante en temas de implementación.

3.3. Escenarios de simulación

Para el estudio por simulación se ha utilizado la topología de la Fig. 3.2. Ocho fuentes TCP Reno generan tráfico a la velocidad del enlace, establecida a 33 Mbps. En las simulaciones analizaremos el impacto causado por tener fuentes con diferentes contratos. Asimismo, también estudiaremos la influencia de encontrarnos con distintos RTT para cada conexión. En un escenario TCP homogéneo, aquel en el que todas las fuentes tienen el mismo RTT, éste queda fijado a 50 ms. Mientras que en un escenario TCP heterogéneo, el valor del RTT varía de 10 ms a 80 ms (a incrementos de 10 ms). Los parámetros HBO_{in} y HBO_{out} especifican el máximo número de paquetes permitidos en cada cola. En nuestras simulaciones, HBO_{in} y HBO_{out} se limitan a 20 y 10 paquetes respectivamente.

Como se indica en el Capítulo 1 Sección 1.3, hemos utilizado como primera aproximación un tamaño de paquete de 9,188 bytes que se corresponde con IP sobre ATM. No obstante, para incluir en este trabajo casos peores de estudio, realizaremos simulaciones con otros tamaños de paquete. Recordemos que trabajamos con fuentes *avariciosas* o *long-lived*, es decir que disponen de una cantidad ilimitada de datos para enviar.

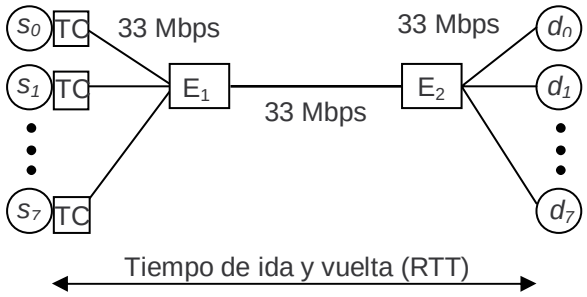


Fig. 3.2 Topología para las simulaciones (TC ≡ Acondicionador de Tráfico). Ocho fuentes TCP Reno s_0 a s_7 transmitiendo a velocidad del enlace (33 Mbps). El cuello de botella se encuentra entre los nodos E_1 y E_2

Obtendremos resultados para cinco escenarios diferentes, todos ellos con una carga de red del 60% de la capacidad del enlace. En esta situación podremos analizar cómo se distribuye el 40% del ancho de banda en exceso entre las distintas fuentes.

- Escenario A. Todas las conexiones tienen el mismo RTT y el mismo contrato. Las simulaciones se han realizado con contratos de 2,4 Mbps y 50 ms para todas las fuentes. Este es el escenario más simple y el más utilizado en estudios relacionados con acondicionado de tráfico y DiffServ.
- Escenario B. Todas las conexiones tienen el mismo RTT pero diferente contrato. Las simulaciones se han realizado con contratos de 1, 1, 2, 2, 3, 3, 4 y 4 Mbps de s_0 a s_7 respectivamente. El RTT es de 50 ms para todas las fuentes. En este escenario podemos medir la influencia de los contratos.
- Escenario C. Todas las conexiones tienen distinto RTT pero el mismo contrato. Las simulaciones se han realizado con contratos de 2,4 Mbps y RTT de 10, 20, 30, 40, 50, 60, 70 y 80 ms de s_0 a s_7 respectivamente. En este escenario podemos medir la influencia del RTT.
- Escenario D. Todas las conexiones tienen diferente RTT y diferente contrato. En concreto, las fuentes con menor contrato tendrán los RTT más bajos. Las

simulaciones se han realizado con contratos de 1, 1, 2, 2, 3, 3, 4 y 4 Mbps (de s_0 a s_7) y RTT de 10, 20, 30, 40, 50, 60, 70 y 80 ms (de s_0 a s_7). En este caso analizaremos ambos efectos (RTT y contrato) de modo simultáneo.

- Escenario E. Todas las conexiones tienen diferente RTT y diferente contrato. En concreto las fuentes de mayor contrato tendrán los RTT más bajos. Las simulaciones se han realizado con contratos de 4, 4, 3, 3, 2, 2, 1 y 1 Mbps (de s_0 a s_7) y RTT de 10, 20, 30, 40, 50, 60, 70 y 80 ms (de s_0 a s_7).

Los resultados de las simulaciones tienen un intervalo de confianza del 95% calculado con una distribución normal usando 30 muestras, con un valor aproximado de ±0,002 para los valores de justicia y ±0,01 para los contratos alcanzados.

3.4. Resultados

En las siguientes secciones presentamos los resultados obtenidos a través de simulaciones para todos los escenarios descritos en la Sección 3.3. Como comentamos en la introducción, analizaremos las prestaciones en función del caudal obtenido por cada fuente, sin considerar paquetes retransmitidos (*goodput*) y de la justicia en el reparto del ancho de banda excedente. Ésta se puede observar calculando el índice de justicia de Jain explicado en la Sección 1.1.1.3 (ec. 1.1). Además, demostraremos que el esquema CBDQ no afecta al funcionamiento del acondicionador de tráfico CB, en tanto que, como veremos, se siguen asegurando los contratos.

3.4.1. Distribución del ancho de banda no contratado

Las simulaciones llevadas a cabo para los cinco escenarios descritos en la Sección 3.3 muestran que las prestaciones mejoran notablemente cuando empleamos CBDQ. En concreto, el reparto del ancho de banda en exceso se realiza de modo más justo (equitativo) que en otras implementaciones con TSW, LB o CB y RIO (véase la Tabla 3.1). De los resultados incluidos en esta tabla, podemos observar que la mejora del índice de justicia es pequeña en los casos A y B (entorno a 0,1 puntos). Esta mejora tan reducida se debe a las llanas características de estos dos escenarios, en los que el RTT es siempre el mismo para todas las conexiones. Con estas particularidades, varias parejas (acondicionador de tráfico-RIO) presentan buenas prestaciones, tal y como ocurre con TSW, LB y CB.

En los casos C, D y E, el efecto de tener RTT distintos en cada fuente pone de manifiesto las ventajas de utilizar CBDQ. Según muestra la Tabla 3.1, en escenarios heterogéneos (con diferentes RTT y diferentes contratos) el reparto del ancho de banda en exceso no es tan equitativo como con CBDQ. Una gestión de doble cola que trata a los paquetes *in* y *out* de modo distinto, pero evitando interferencias entre ambos, proporciona mejores resultados. Es de destacar también la simpleza de CBDQ en comparación con la cantidad de parámetros necesaria para configurar RIO.

Tabla 3.1 Tabla comparativa del índice de justicia

Tipo de Acondicionador de Tráfico – Esquema de gestión	Escenario A	Escenario B	Escenario C	Escenario D	Escenario E
CBDQ	0,999	0,971	0,971	0,879	0,952
CB – RIO	0,854	0,855	0,781	0,708	0,836
TSW – RIO	0,582	0,807	0,631	0,489	0,562
LB – RIO	0,853	0,687	0,740	0,817	0,832

De igual modo, hemos realizado simulaciones utilizando una memoria compartida para la cola de paquetes *out*, es decir, la cola de paquetes *out* se separa en varias colas virtuales con el objetivo de conocer el efecto sobre el índice de justicia. El hecho de separar en varias colas virtuales puede incrementar el índice de justicia, puesto que así los paquetes *out* se sitúan de manera aleatoria entre las distintas colas virtuales y éstas se sirven también de modo aleatorio cada vez que se visita la cola de paquetes *out* (nótese que esta cola se visita con probabilidad 1-ρ). Sin embargo, los resultados de simulación obtenidos revelan que la mejora en el índice de justicia es reducida.

3.4.2. Garantías de contrato

Otro tema importante es la interacción entre el acondicionador de tráfico CB y la gestión de doble cola DQ. En esta sección, presentamos los resultados de simulación que confirman la buena relación entre ambos. Esta afinidad es la responsable de que los contratos se aseguren al cien por cien con los paquetes marcados como *in*. Las Fig. 3.3 a 3.7 representan el caudal que alcanzan las fuentes TCP con paquetes *in* en todos los escenarios desde el A hasta el E. Según se desprende de estas figuras, tras un corto periodo transitorio el ancho de banda contratado se garantiza de modo estricto.

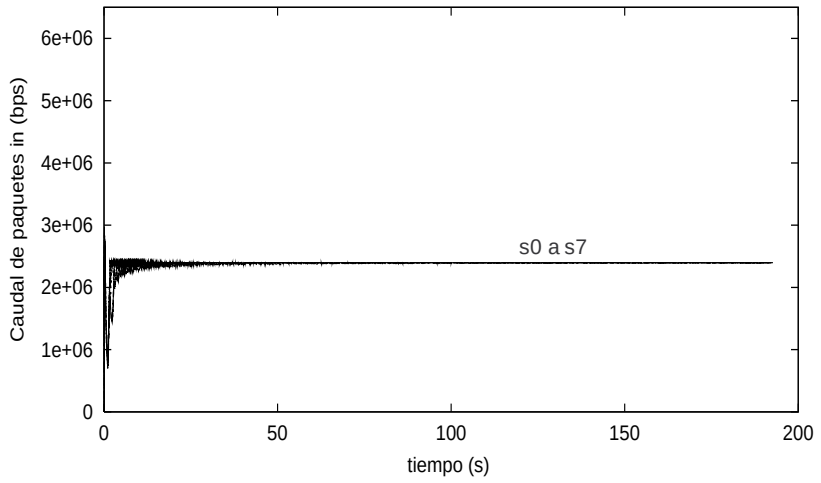


Fig. 3.3 Caudal de paquetes *in* en el escenario A. Todas las fuentes contratan un ancho de banda de 2,4 Mbps. El RTT es el mismo, 50 ms, para todas las conexiones

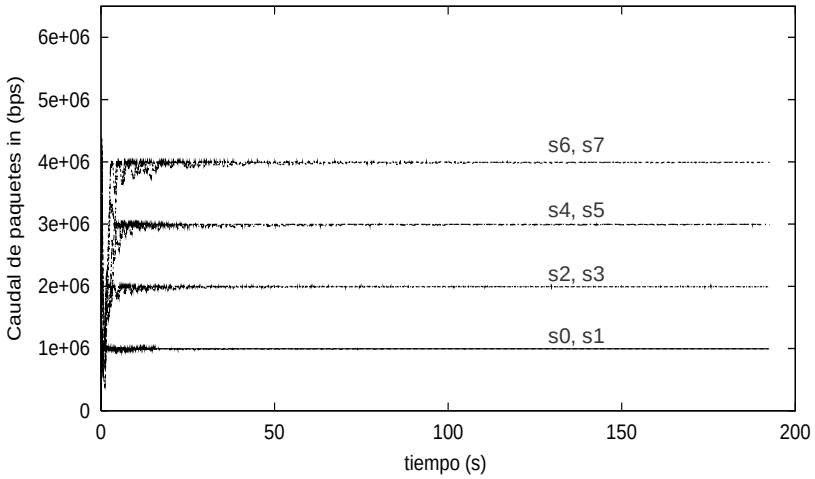


Fig. 3.4 Caudal de paquetes *in* en el escenario B. El contrato de cada fuente es de 1, 1, 2, 2, 3, 3, 4 y 4 Mbps de s_0 a s_7 respectivamente. El RTT es el mismo, 50 ms, para todas las conexiones

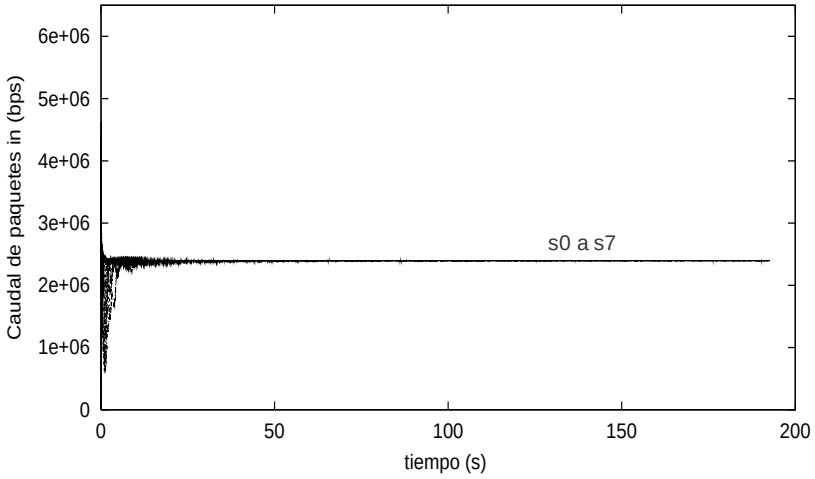


Fig. 3.5 Caudal de paquetes *in* en el escenario C. Todas las fuentes contratan un ancho de banda de 2,4 Mbps. El RTT es de 10, 20, 30, 40, 50, 60 70 y 80 ms para las conexiones s_0 a s_7 respectivamente

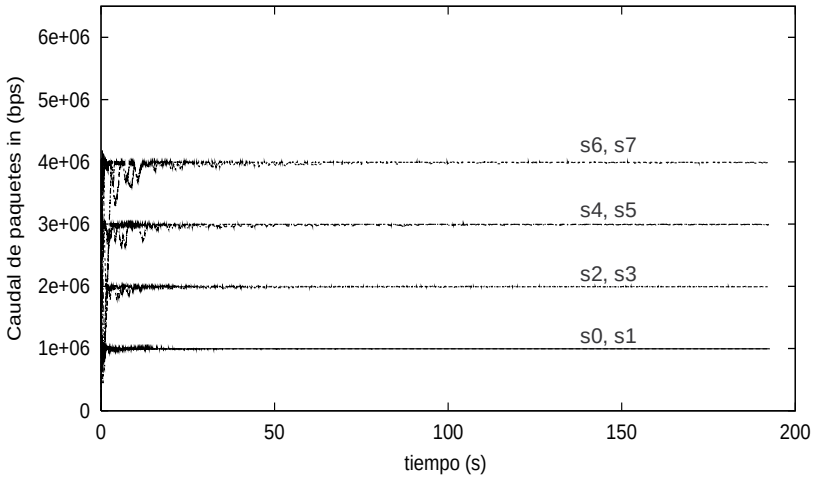


Fig. 3.6 Caudal de paquetes *in* en el escenario D. El contrato de cada fuente es de 1, 1, 2, 2, 3, 3, 4 y 4 Mbps de s_0 a s_7 respectivamente.El RTT es de 10, 20, 30, 40, 50, 60 70 y 80 ms para las conexiones s_0 a s_7 respectivamente

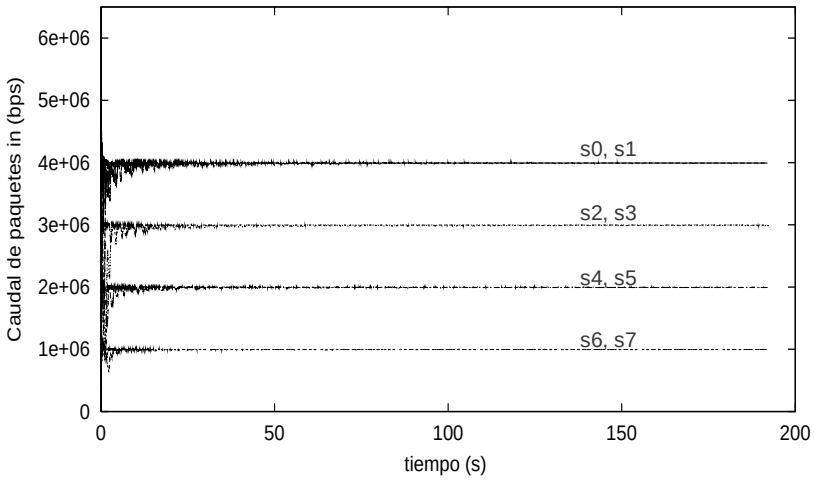


Fig. 3.7 Caudal de paquetes *in* en el escenario E. El contrato de cada fuente es de 4, 4, 3, 3, 2, 2, 1 y 1 Mbps de s_0 a s_7 respectivamente.El RTT es de 10, 20, 30, 40, 50, 60 70 y 80 ms para las conexiones s_0 a s_7 respectivamente

3.4.3. Efecto de los tamaños de paquete

Puesto que en una situación real los nodos de la red reciben paquetes de distinta longitud, consideramos interesante estudiar el efecto que produce en las prestaciones de CBDQ tener distintos tamaños de paquete. De nuevo, analizaremos el impacto centrándonos en la influencia sobre la distribución del ancho de banda no contratado. En primer lugar, hemos realizado simulaciones donde todas las fuentes generan paquetes del mismo tamaño. Los tamaños de

paquete escogidos son 1.500, 5.300 y 10.000 bytes. Las simulaciones se han realizado para todos los escenarios descritos en la Sección 3.3. Aunque un paquete de tamaño 5.300 bytes no es característico de ningún servicio conocido, lo hemos incluido como valor intermedio para disponer de simulaciones con paquetes de tamaño pequeño, mediano y grande. Tras las simulaciones, hemos calculado los índices de justicia reflejados en la Tabla 3.2. Como se aprecia, el índice de justicia está por encima de 0,8 en la gran mayoría de los casos analizados (considérese que a partir de este valor el reparto es justo). Son sólo dos los casos en los que el índice de justicia permanece por debajo de 0,8: escenarios C y D con un tamaño de paquete de 1.500 bytes. Según las simulaciones, el sistema CBDQ no estaría preparado para trabajar con paquetes de tamaño pequeño en escenarios heterogéneos con grandes variaciones en los RTT de cada conexión. Con todo, puede que estos resultados pocos favorecedores se deban a la propia granularidad del simulador, en principio desarrollado para trabajar con paquetes de gran tamaño.

Tabla 3.2 Índice de justicia aplicando CBDQ con diferentes tamaños de paquete (no simultáneos)

Tamaño de paquete (bytes)	Escenario A	Escenario B	Escenario C	Escenario D	Escenario E
1.500	0,99	0,90	0,64	0,40	0,86
5.300	0,99	0,94	0,90	0,76	0,98
10.000	0,99	0,97	0,97	0,88	0,95

Para examinar con más profundidad la influencia del tamaño de paquete, hemos realizado nuevas simulaciones donde simultáneamente nos encontramos con paquetes de diferente tamaño. Las fuentes s_0, s_3 y s_6 generan paquetes de 10.000 bytes. Las fuentes s_1, s_4 y s_7 generan paquetes de 5.300 bytes. Por último, las fuentes s_2 y s_5 generan paquetes de 1.500 bytes. Los índices de justicia obtenidos se incluyen en la Tabla 3.3. Podemos observar que el índice de justicia decrece cuando por la red viajan paquetes de diverso tamaño. A pesar de que algunos resultados parecen recomendar el uso de TSW o LB con RIO, debemos recordar que ninguno de ellos es capaz de garantizar con total exactitud los contratos. Sin olvidar además la compleja configuración de RIO. Es por esto que los resultados de la Tabla 3.3 no se pueden tomar como una conclusión general y requieren de un estudio más exhaustivo.

Tabla 3.3 Tabla comparativa del índice de justicia con diferentes esquemas de acondicionado y gestión de cola con tamaños de paquete variables (simultáneos)

Tipo de Acondicionador de Tráfico – Esquema de gestión	Escenario A	Escenario B	Escenario C	Escenario D	Escenario E
CBDQ	0,70	0,56	0,63	0,45	0,61
CB-RIO	0,64	0,38	0,68	0,76	0,66
TSW-RIO	0,56	0,51	0,57	0,49	0,63
LB-RIO	0,75	0,63	0,73	0,62	0,70

3.5. Conclusiones

Como se ha mencionado en este capítulo y en capítulos anteriores, la investigación en el campo de los Servicios Diferenciados no ha sido capaz hasta el momento de encontrar una solución para el servicio asegurado que garantice contratos y un reparto justo del ancho de banda en exceso. En este capítulo hemos propuesto un nuevo esquema de gestión de cola para los nodos de la red diferente del esquema RIO tradicional. Esta nueva propuesta da un trato

diferente a los paquetes marcados como *in* o *out* (al igual que RIO) pero evitando interferencias entre ellos. Con este objetivo empleamos dos colas, una para los paquetes *in* y otra para los paquetes *out*. El algoritmo de planificación que sirve las colas funciona del siguiente modo. La cola de paquetes *in* es atendida con probabilidad igual a la carga de la red, es decir, la suma total de todos los anchos de banda contratados dividido por la capacidad del enlace. La cola de paquetes *out* se visita con probabilidad complementaria. Nótese que este esquema no sigue rigurosamente la definición del AF PHB, puesto que los paquetes *out* puede que no se sirvan en el orden de llegada. La combinación del acondicionador de tráfico CB (encargado de marcar los paquetes) y la gestión de doble cola da lugar al esquema CBDQ (*Counters Based Dual Queuing*).

Los resultados de simulación muestran que empleando CB para marcar los paquetes y DQ en los nodos de la red podemos alcanzar un servicio asegurado que garantiza los contratos y distribuye equitativamente el ancho de banda en exceso entre las distintas fuentes que forman el agregado. Si comparamos las prestaciones de CBDQ con las implementaciones clásicas TSW-RIO y LB-RIO, vemos que CBDQ ofrece un índice de justicia por encima de 0,8 en la mayoría de los casos analizados. Mientras que, TSW-RIO y LB-RIO presentan un índice de justicia casi siempre inferior a 0,8. En conclusión, la combinación CBDQ carece de complejidad en su implementación y ofrece unas prestaciones comparativamente más notables que sus predecesores TSW-RIO, LB-RIO y CB-RIO.

Capítulo 4

Counters Based Modified: Nuevo método de acondicionado de tráfico basado en el descarte anticipado de paquetes

4.1. Introducción

En este capítulo introducimos una nueva propuesta para lograr un servicio asegurado que realice un reparto equitativo del ancho de banda no contratado entre las distintas fuentes TCP que forman el agregado. Partiendo de las buenas cualidades del acondicionador CB, como su gran exactitud a la hora de garantizar un ancho de banda mínimo a los usuarios finales, añadiremos una modificación a este algoritmo basada en el descarte probabilístico de paquetes *out* en el propio acondicionador de tráfico. A esta nueva versión la denominamos acondicionador de tráfico basado en contadores modificado (*Counters Based Modified*, CBM). El uso de CBM implica usar cierta señalización que ayudará a determinar la probabilidad de descarte de los paquetes *out*, tales como la cantidad de ancho de banda no contratado o el tiempo de ida y vuelta medio de las conexiones. Sin embargo, esta señalización sólo es necesaria en el bucle de usuario (distancias cortas), lo que lo convierte en más factible que otras propuestas de acondicionado como [LIN99] [NAND00] [KIM99] [ALVE00] [ANDR00].

En este capítulo describiremos las características de CBM y estudiaremos su posible implementación a través de simulación en topologías heterogéneas. El análisis extremo a extremo lo realizaremos en términos de garantías de proporcionar al usuario final el ancho de banda contratado y de conseguir un reparto justo del ancho de banda en exceso. Este estudio sobre fuentes TCP Reno lo cometeremos primero en una topología de un único cuello de botella con características misceláneas (variación de contratos, variación de retardos y compartición de recursos con fuentes *best-effort*). Compararemos las prestaciones de la pareja CBM-RIO con CB-RIO, TSW-RIO y LB-RIO. A continuación, evaluaremos las prestaciones de CBM en una topología de tres nodos y así valorar su robustez en escenarios más realistas.

El resto del capítulo se estructura del siguiente modo. La Sección 4.2 define las bases del nuevo acondicionador de tráfico CBM. En la Sección 4.3 se describen los escenarios de simulación para realizar el estudio de prestaciones. Los resultados obtenidos se discuten en la Sección 4.4. El capítulo finaliza con las conclusiones en la Sección 4.5.

4.2. El acondicionador de tráfico CBM

En el Capítulo 2 se introdujo el acondicionador de tráfico *Counters Based* o CB. Este acondicionador, que básicamente marca los paquetes como *in* o *out*, ofrece mejores prestaciones que otras propuestas como TSW. CB garantiza los contratos de modo estricto en escenarios misceláneos con diversidad de contratos y de retardo. Sus características más destacables son la

sencillez de uso y su gran exactitud. Para implementar este algoritmo sólo se necesitan dos contadores y no requiere ninguna configuración de parámetros. Del estudio mediante simulación desarrollado en el Capítulo 2, observamos que la combinación CB-RIO es comparativamente mejor que TSW-RIO o LB-RIO. En concreto, CB-RIO asegura los contratos de los usuarios con pequeñas fluctuaciones que no superan el 1% del ancho de banda contratado. Sin embargo, CB no es capaz de resolver el problema de la distribución justa del ancho de banda en exceso entre las fuentes que componen un agregado. La Fig. 2.14 incluye el pseudo-código del algoritmo CB. No obstante y dada su simplicidad, intentamos modificarlo para conseguir una distribución equitativa del ancho de banda en exceso.

Idealmente, si todos los paquetes tienen un tamaño similar y todas las fuentes introducen el mismo número de paquetes *out* en la red, entonces todas las fuentes obtienen la misma porción de ancho de banda en exceso. Este comportamiento ideal se ve afectado por las características propias de cada conexión TCP, como diferentes retardos, diferentes contratos, interacción con los mecanismos de gestión de colas de los enrutadores, etc.

Para hacer frente a estas influencias, proponemos que aquellas conexiones que envían paquetes *out* por encima de su cuota ideal deben ser penalizados mediante descarte probabilístico de paquetes *out* en el propio acondicionador de tráfico. La duda que surge es cómo seleccionar qué paquetes se descartan y cuáles se envían a la red. Para solucionar esta cuestión, estudiamos el comportamiento del ancho de banda en exceso desde una perspectiva diferente. Mediante simulaciones, observamos el número de paquetes *out* que se generan entre paquetes *in* consecutivos. A partir de estas simulaciones advertimos que:

(i) Una fuente con un contrato pequeño genera más paquetes *out* entre dos paquetes *in* consecutivos que una fuente con contrato mayor. Las fuentes TCP intentan transmitir información a la velocidad del enlace, de modo que cuanto menor sea el contrato más paquetes *out* se inyectan en la red. Ésta es la razón de que este grupo de fuentes pueda obtener más recursos.

(ii) La rápida respuesta de las fuentes TCP con tiempos de ida y vuelta pequeños las hace inyectar más tráfico en la red, es decir más paquetes *out*. Por tanto, una fuente con RTT pequeño puede generar más paquetes *out* entre paquetes *in* consecutivos que una fuente con RTT mayor.

Un ejemplo ilustrando este hecho se representa en las Fig. 4.1 y 4.2. Para las simulaciones utilizamos el algoritmo CB y RIO, con ocho fuentes TCP Reno (s_0 a s_7) generando tráfico a la velocidad del enlace (33 Mbps). Véase la Fig. 4.10. Cada fuente, de s_0 a s_7, contrata un ancho de banda de 1, 1, 2, 2, 3, 3, 4 y 4 Mbps respectivamente. El RTT se establece a 10, 20, 30, 40, 50, 60, 70 y 80 ms de s_0 a s_7. El eje de abcisas representa el tiempo en segundos y el eje de ordenadas el número de paquetes *out* entre paquetes *in* consecutivos. Contemplando ambas figuras vemos que la fuente con RTT más bajo y contrato más pequeño inyecta más paquetes *out* en la red. Por ejemplo, el número total de paquetes *out* generados por s_1 (contrato de 1 Mbps y RTT de 20 ms) es de 6.031, mientras que para s_7 (contrato de 4 Mbps y RTT de 80 ms) es casi la tercera parte, 2.331 paquetes.

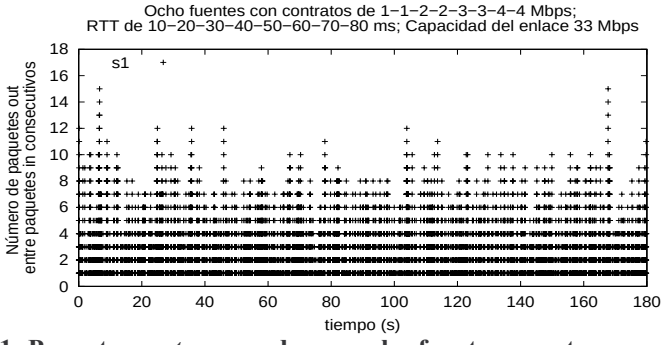


Fig. 4.1 Paquetes *out* generados por la fuente s_1 entre paquetes *in* consecutivos (número total de paquetes *out* = 6.031)

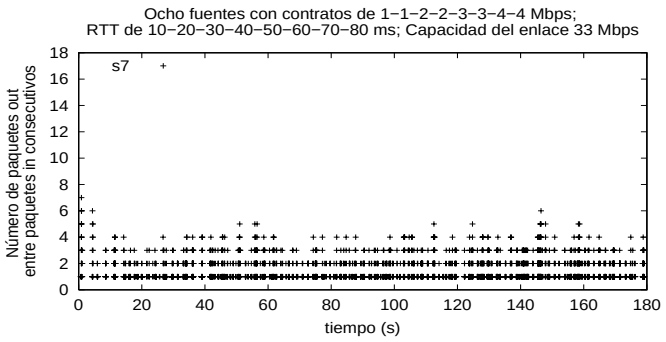


Fig. 4.2 Paquetes *out* generados por la fuente s_7 entre paquetes *in* consecutivos (número total de paquetes *out* = 2.331)

De estas observaciones la idea que sugerimos es la siguiente (Fig. 4.3). El acondicionador de tráfico CBM, situado junto a las fuentes de tráfico TCP, pero fuera del alcance del usuario final, emplea una variable que cuenta el número de paquetes marcados como *out* entre paquetes *in* consecutivos. Cada vez que un paquete se marca como *out*, el acondicionador CBM comprueba esta variable. Si la variable no excede un valor mínimo *min*, entonces el paquete *out* se transmite a la red. Si la variable excede un valor máximo *max*, el paquete *out* se descarta. Finalmente, si la variable queda dentro del rango (*min*, *max*), el paquete *out* se descarta con probabilidad *p*. Este nuevo algoritmo del tipo RED intenta prevenir una alta densidad de paquetes *out* procedentes de la misma conexión TCP.

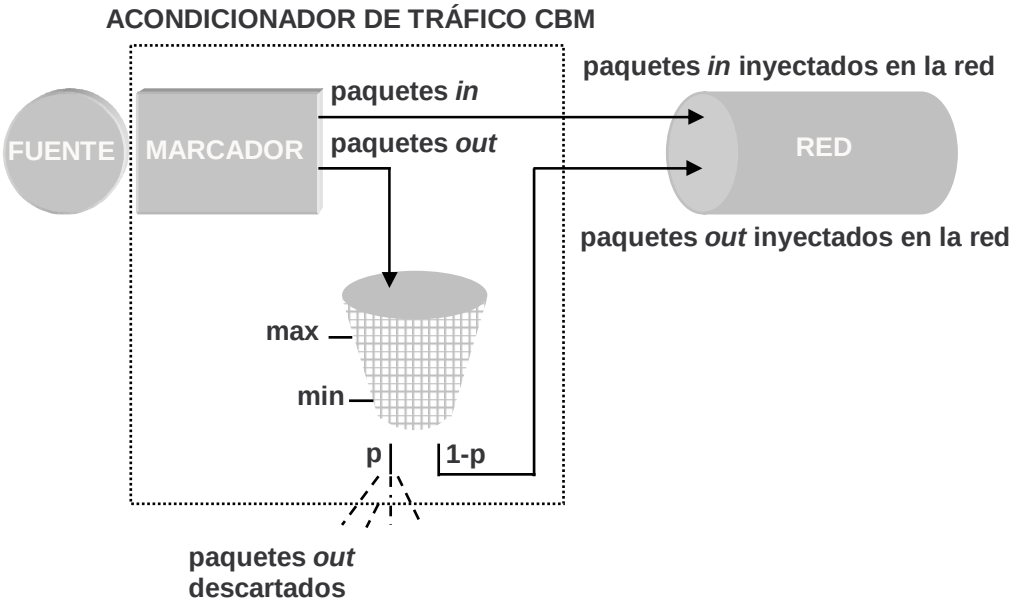


Fig. 4.3 Descarte de paquetes *out* en el acondicionador de tráfico CBM

4.2.1. Cómo establecer los límites *max* y *min*

Para sintonizar los parámetros *max* y *min* seguimos las ecuaciones (4.1) y (4.2), donde MSS es el tamaño máximo de segmento (*Maximum Segment Size*). Podemos ver el ancho de banda en exceso como el ancho de banda consumido por otra fuente TCP, cuyo máximo tamaño de ventana TCP viene determinado por el producto ancho de banda en exceso ($bandwidth_{exceso}$) por el RTT medio (RTT_{medio}). Así, el límite *max* queda fijado a este valor. Una fuente que genere un número de paquetes *out* cercano a este límite consume en su totalidad casi todo el ancho de banda en exceso. Al mismo tiempo, si este límite se excede la fuente podría estar *robando* parte del ancho de banda garantizado de otras conexiones, por lo que la fuente no puede inyectar paquetes *out* por encima de este valor máximo. Es bien conocido que en TCP/IP, un incremento aditivo y un decremento multiplicativo satisface las condiciones necesarias para converger a un estado eficiente de la red, y se utiliza para implementar esquemas de prevención de la congestión. Por esta razón, el valor escogido para el límite inferior *min* es la mitad del máximo.

$$max = \left\lceil \frac{bandwidth_{exceso} \cdot RTT_{medio}}{MSS} \right\rceil \tag{4.1}$$

$$min = \left\lceil \frac{max}{2} \right\rceil \tag{4.2}$$

4.2.2. Estimación del tiempo de ida y vuelta

Aunque está fuera del alcance de este trabajo identificar la técnica que ofrece una mejor aproximación del RTT promedio, sugerimos un posible método. Esta estima se puede obtener a partir de señalización periódica desde el nodo frontera de la red a las fuentes de tráfico. El protocolo TCP implementa un algoritmo que estima el tiempo de ida y vuelta de la conexión [STEV94]. Esta estima se puede enviar de modo periódico al dispositivo frontera, que calcula el RTT promedio. Este valor se envía entonces de vuelta a los acondicionadores de tráfico. La señalización no resulta un gran problema dado que este camino se corresponde con el bucle de usuario local (distancias cortas). Nótese que no se requiere monitorización por flujos en el nodo frontera, en el sentido de que este dispositivo no contiene información permanente de cada flujo individual de paquetes. Sólo ha de determinar el RTT promedio con la información que recibe de las conexiones TCP, y una vez fijado, esta información no queda almacenada, a diferencia de otras implementaciones como [LIN99] [KIM99] [ALVE00] [ANDR00]. Pruebas experimentales para corroborar este método se dejan para estudios futuros.

4.2.3. Probabilidad de descarte de los paquetes *out*

Definimos la probabilidad de descarte *p* según indica la ecuación (4.3). Cada fuente tendrá una probabilidad de descarte distinta, entre 0 y 1, basada en el valor del ancho de banda contratado. Observe que *p* sólo se recalcula en el caso de que un usuario varíe su contrato.

$$p = 2 \cdot \frac{contrato / capacidad\ del\ enlace}{1 + contrato / capacidad\ del\ enlace} \tag{4.3}$$

De las ideas planteadas en (i) e (ii), sería intuitivo aplicar una ecuación de descarte de la forma $p=1-x$ (véase Fig. 4.4), donde x es el cociente de dividir el contrato entre la capacidad del enlace. Si hubiésemos utilizado esta ecuación, las conexiones de contratos más pequeños tirarían más paquetes *out*. Sin embargo, una vez el límite máximo *max* queda establecido, el

acondicionador de tráfico produce la pérdida de todos los paquetes *out* por encima de este límite. La pérdida de paquetes disminuye la velocidad de las fuentes, de manera que otras fuentes pueden enviar más tráfico en la red, es decir, más paquetes *out*. Si usamos una ecuación que favorece a las fuentes de contrato mayor, estaremos penalizando en exceso a las fuentes de menor contrato. Por lo que cuando estas fuentes se recuperen de las pérdidas los recursos estarán siendo utilizados por las de mayor contrato. Esta situación causaría nuevas pérdidas y haría que las fuentes de menor contrato redujesen de nuevo su velocidad, originando el efecto opuesto a (i) e (ii), que tampoco es deseable. En consecuencia, debemos utilizar una ecuación para la probabilidad de descarte que dé una mínima preferencia a las fuentes de contrato menor.

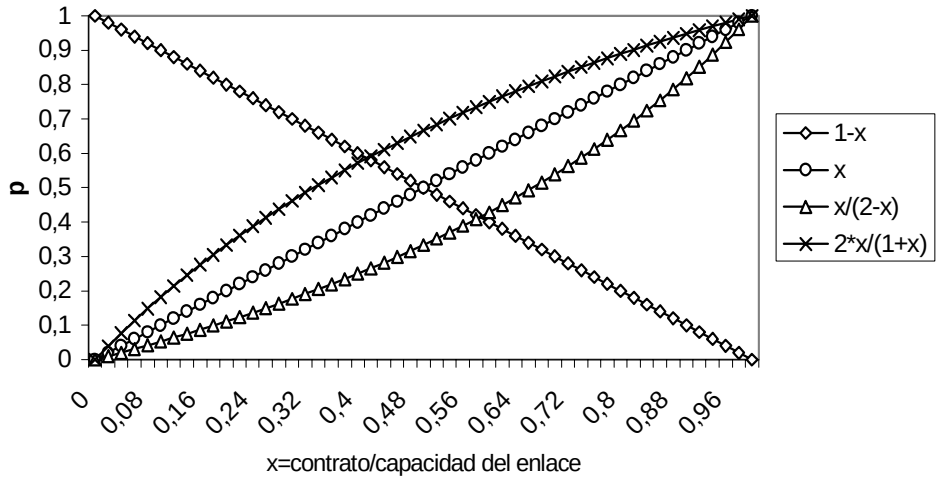


Fig. 4.4 Ecuaciones examinadas para encontrar la curva de probabilidad de descarte de paquetes *out* en el acondicionador de tráfico CBM

En primer lugar evaluamos una ecuación de la forma $p=x$ (véase Fig. 4.4). Las simulaciones mostraron que con esta opción, CBM realizaba un reparto más justo del ancho de banda que su predecesor CB, pero todavía estaba lejos del comportamiento ideal. Estudiamos como afectaría a las prestaciones de CBM variar la ecuación de probabilidad y convertirla en una línea curva. Efectuamos nuevas simulaciones con $p=2 \cdot x/(1+x)$ y $p=x/(2-x)$ (véase la Fig. 4.4), dos curvas que dan un poco de más preferencia a las fuentes de menor contrato. Es importante señalar que mínimas diferencias en el valor de p causan diferencias notables en las prestaciones finales.

De todos estos resultados, concluimos que la expresión (4.3) es la más apropiada para mejorar las prestaciones de CBM. Nótese que la ecuación (4.3) se aplica únicamente cuando el número de paquetes *out* está en el intervalo (*min*, *max*). El pseudo-código simplificado del algoritmo CBM se muestra en la Fig. 4.5.

4.2.4. El control de la generación de paquetes *out*

En esta sección demostraremos que los límites *max* (ec. 4.1) y *min* (ec. 4.2), junto con el uso de la probabilidad de descarte p (ec. 4.3), permiten que el acondicionador de tráfico CBM controle el número de paquetes *out* que cada fuente introduce en la red.

Este efecto se desprende de las Fig. 4.6 a 4.9. Las simulaciones para obtener estos resultados se hicieron en una topología con un único cuello de botella (Fig. 4.10) con ocho fuentes TCP Reno transmitiendo a la velocidad del enlace (33 Mbps). El cuello de botella se sitúa entre los nodos E_1 y E_2, que implementan RIO. Cada conexión (s_0 a s_7) contrata un ancho de banda de 1,

1, 2, 2, 3, 3, 4 y 4 Mbps respectivamente. El RTT queda fijado a 10, 20, 30, 40, 50, 60, 70 y 80 ms de s_0 a s_7. La Sección 4.3 detalla con mayor profundidad las topologías simuladas.

```
Inicialmente:
   C1=1
   C2=capacidad del enlace/contrato
   C3=0
   Calcular los valores de la probabilidad p y los límites max y min
Para cada unidad de tiempo:
   C2--
   if C2 <= 0
        C1++
        C2=capacidad del enlace/contrato
   if llegada de paquete
        if C1>0
           paquete marcado como in
           C1--
           C3=0
        else
           paquete marcado como out
           C3++
           if tiempo>tiempo inicio de descarte
               if C3>max
                  paquete out descartado
               else
                  if C3>min
                     paquete out descartado con probabilidad p
                  else
                     paquete out se envía a la red
```

Fig. 4.5 Pseudo-código simplificado del acondicionador de tráfico CBM

La Fig. 4.6 y la Fig. 4.8 representan el número de paquetes *out* entre paquetes *in* consecutivos usando el acondicionador de tráfico CB, es decir, sin descarte probabilístico de paquetes *out* en el acondicionador. La Fig. 4.6 muestra la fuente s_1 (contrato de 1 Mbps y RTT de 20 ms). La Fig. 4.8 muestra la fuente s_7 (contrato de 4 Mbps y RTT de 80 ms).

Por su parte, la Fig. 4.7 y la Fig. 4.9 ilustran el control que tiene CBM sobre la generación de paquetes *out*. En este caso, el ancho de banda en exceso es de 13 Mbps, el RTT promedio es de 45 ms y el MSS es de 9.188 bytes. Con estos datos los límites *max* y *min* se establecen a 9 y 5 paquetes (ec. 4.1 y 4.2). La probabilidad de descarte *p* es de 0,059 para s_1 (Fig. 4.7) y 0,216 para s_7 (Fig. 4.9).

Comparando las Fig. 4.6 y 4.7, observamos que CBM fuerza a las fuentes con contratos más pequeños y RTT más bajos (por ejemplo s_1) a generar menos paquetes *out*. Del mismo modo, con este mecanismo las conexiones de mayor contrato y RTT más altos (por ejemplo s_7) incrementan el número de paquetes *out* generados, como muestran las Fig. 4.8 y 4.9. Cuando se supera el límite máximo *max* de paquetes *out* entre paquetes consecutivos *in*, aquellos paquetes se descartan. Las conexiones TCP reflejan estas pérdidas disminuyendo su velocidad, y se inyecta más tráfico en exceso (paquetes *out*) de otras fuentes. Puesto que CBM es capaz de controlar el número de paquetes *out* que se introducen en la red, presumimos que también podrá ejecutar un reparto justo del ancho de banda en exceso, tal como veremos en las siguientes secciones.

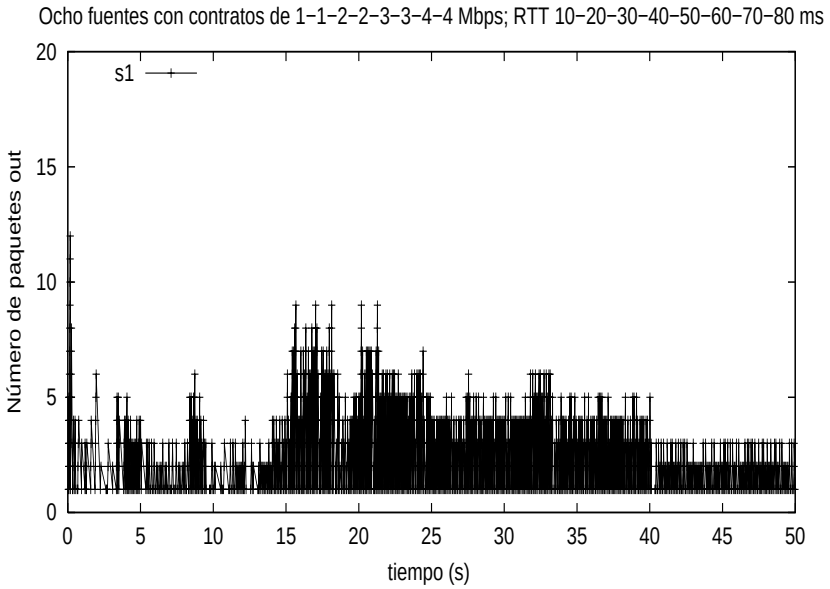


Fig. 4.6 Paquetes *out* entre paquetes *in* consecutivos en la fuente s_1 *sin* descarte de paquetes *out* en el acondicionador de tráfico (número total de paquetes *out* = 7.183). Se emplea el acondicionador de tráfico CB

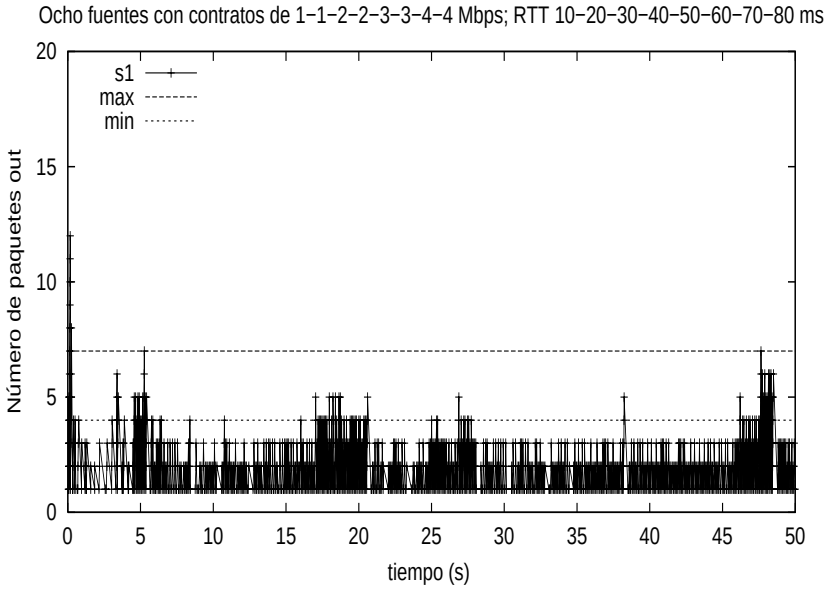


Fig. 4.7 Paquetes *out* entre paquetes *in* consecutivos en la fuente s_1 *con* descarte de paquetes *out* en el acondicionador de tráfico (número total de paquetes *out* = 5.674). Se emplea el acondicionador de tráfico CBM

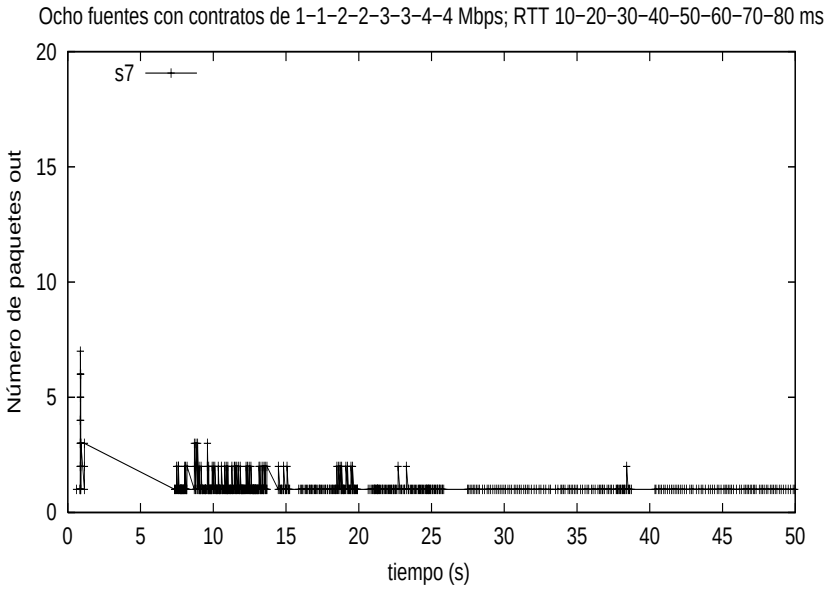


Fig. 4.8 Paquetes *out* entre paquetes *in* consecutivos en la fuente s_7 *sin* descarte de paquetes *out* en el acondicionador de tráfico (número total de paquetes *out* = 435). Se emplea el acondicionador de tráfico CB

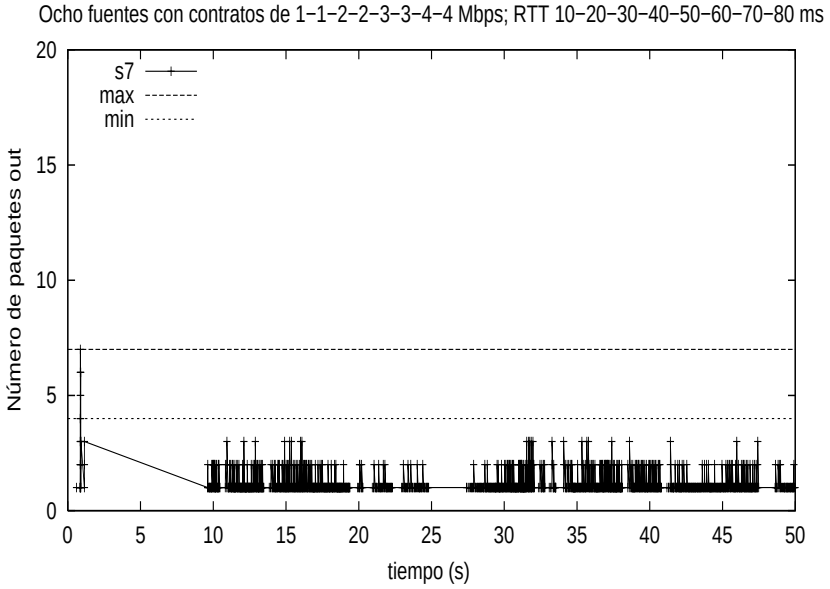


Fig. 4.9 Paquetes *out* entre paquetes *in* consecutivos en la fuente s_7 *con* descarte de paquetes *out* en el acondicionador de tráfico (número total de paquetes *out* = 2.781). Se emplea el acondicionador de tráfico CBM

4.3. Topologías y escenarios de simulación

Evaluamos las prestaciones del acondicionador de trafico CBM mediante simulaciones. Para conseguir un análisis lo más amplio posible, empleamos en primer lugar una topología de un único cuello de botella y en segundo lugar una topología de dos cuellos de botella. Las Fig. 4.10 y 4. 11 representan las topologías generales de uno y dos cuellos de botella respectivamente. El tráfico lo generan *n* fuentes TCP Reno que transmiten a la velocidad del enlace, 33 Mbps. Los nodos de la red almacenan y encaminan el tráfico. El sistema de prevención de la congestión empleado en las colas de estos nodos es RIO, a menos que se indique otro esquema de gestión. Los parámetros utilizados con RIO son [40/70/0,02] para los paquetes *in* y [10/40/0,2] para los paquetes *out*. Los valores de los parámetros de RED (*weight_in* y *weight_out*) utilizados para calcular el tamaño medio de la cola se han escogido iguales a 0,002 como se recomienda en [FLOY93].

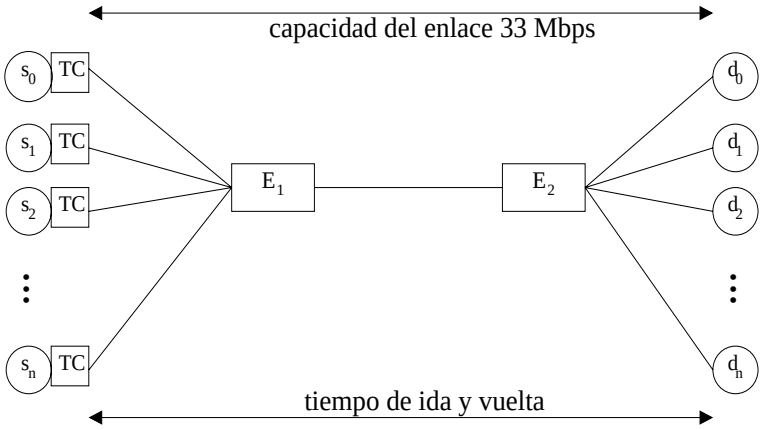


Fig. 4.10 Topología general de un único cuello de botella entre los nodos E_1 y E_2 (TC ≡ Acondicionador de tráfico)

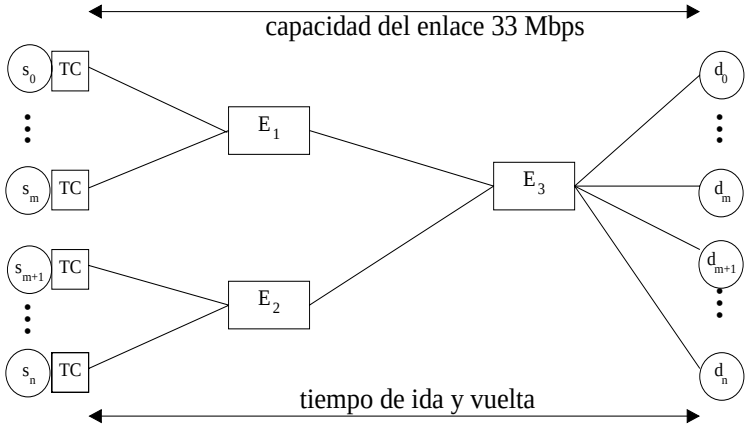


Fig. 4.11 Topología general de tres nodos

Tanto en topologías de un único cuello de botella como en las de dos cuellos de botella, verificaremos el impacto de la variabilidad de retardos y de contratos sobre las prestaciones finales simulando distintos escenarios. En todos ellos trabajaremos con una carga de red del 60% aproximadamente. Los características de los escenarios se resumen a continuación, aunque los valores de los contratos y del RTT utilizados se indicarán específicamente en cada sección.

- Escenario A. Todas las conexiones tienen el mismo RTT y el mismo contrato. Este es el escenario más simple y el más utilizado en estudios relacionados con acondicionado de tráfico y DiffServ.
- Escenario B. Todas las conexiones tienen el mismo RTT pero diferente contrato. En este escenario podemos medir la influencia de los contratos.
- Escenario C. Todas las conexiones tienen distinto RTT pero el mismo contrato. En este escenario podemos medir la influencia del RTT.
- Escenario D. Todas las conexiones tienen diferente RTT y diferente contrato. En concreto, las fuentes con menor contrato tendrán los RTT más bajos. En este caso analizaremos ambos efectos (RTT y contrato) de modo simultáneo. Este es el caso peor de estudio, porque el hecho de que las fuentes con contratos más pequeños tengan los retardos más bajos hace que estas conexiones TCP se vean favorecidas a la hora de obtener recursos de red, tal y como se indica el Capítulo 2 y en estudios previos [LIN99] [SEDD99].
- Escenario E. Todas las conexiones tienen diferente RTT y diferente contrato. En concreto las fuentes de mayor contrato tendrán los RTT más bajos. Éste es también uno de los escenarios más complejos, pero al tener las conexiones de menor contrato los retardos más elevados se evita un favoritismo excesivo como el que ocurre en el escenario D.

Los resultados de las simulaciones tienen un intervalo de confianza del 95% calculado con una distribución normal usando 30 muestras, con un valor aproximado de ±0,002 para los valores de justicia y ±0,01 para los contratos alcanzados.

4.4. Resultados

En la Sección 4.2 explicamos los principios del algoritmo CBM, destacando su habilidad para controlar la generación de paquetes *out* gracias al descarte probabilístico de éstos en el propio acondicionador. La probabilidad de descarte de un paquete *out* se calcula a partir del ancho de banda en exceso y el tiempo de ida y vuelta promedio de las conexiones, intentando de este modo evitar la influencia que estos parámetros puedan tener sobre las prestaciones finales extremo a extremo de cada fuente TCP.

En esta sección, se discuten los resultados de simulación obtenidos. En primer lugar examinamos el comportamiento de CBM en una topología de un único cuello de botella de características heterogéneas (diferentes contratos, tiempos de ida y vuelta, número de fuentes, etc.). Como caso particular, se estudiará la robustez de nuestra propuesta cuando están presentes dos tipos de tráfico: asegurado y *best-effort*. En segundo lugar, se realizarán simulaciones en una topología más completa con dos cuellos de botella también con escenarios heterogéneos. Los parámetros de medida de QoS serán el caudal obtenido, para comprobar si se garantizan los anchos de banda contratados, y la adjudicación del ancho de banda no contratado entre las distintas fuentes de tráfico. Para evaluar la justicia del reparto de ancho de banda en exceso emplearemos el índice de justicia *f* de Jain (ec. 1.1). Recordemos que conforme *f* se aproxima a 1, más justo (en sentido equitativo) es el reparto.

4.4.1. Resultados en la topología de un solo nodo

Las dos siguientes secciones muestran los resultados obtenidos mediante simulación en una topología de un único cuello de botella, representada en la Fig. 4.10. Esta topología consta de ocho fuentes TCP Reno y los nodos de la red utilizan RIO.

4.4.1.1. Simulaciones con tráfico asegurado

Como primera situación de estudio suponemos que todas las fuentes contratan un servicio asegurado. Llevamos a cabo simulaciones en los cinco escenarios tipificados en la Sección 4.3. La Tabla 4.1 muestra los valores de los umbrales *min* y *max* (ec. 4.1 y 4.2), calculados a partir de las características de cada escenario.

La Fig. 4.12 muestra los índices de justicia en cada escenario. Se han incluido también los índices de justicia alcanzados con las propuestas clásicas TSW y LB, y los del acondicionador CB. Nótese que ni TSW, ni LB ni CB hacen descarte probabilístico de paquetes *out*. Las simulaciones en las que se empleó el acondicionador de tráfico TSW se realizaron siguiendo las pautas de configuración de la Sección 2.3.2 del Capítulo 2. Es importante establecer adecuadamente los parámetros de configuración de TSW, ya que pequeñas variaciones en éstos provocan variaciones importantes en los resultados obtenidos. Del mismo modo, seguimos las indicaciones de la Sección 2.3.1 del Capítulo 2 para configurar LB.

Los resultados de la Fig. 4.12 revelan que es posible asegurar una distribución justa del ancho de banda en exceso cuando se utiliza el acondicionador de tráfico CBM, con un valor medio de índice de justicia de 0,95. Se observa además que en los escenarios A y B, los algoritmos TSW y LB consiguen un nivel de justicia alto. Sin embargo, no hay que olvidar que si optamos por estos algoritmos no es posible garantizar el ancho de banda contratado.

Se han incluido las Fig. 4.13 y 4.14 para verificar que el descarte probabilístico de paquetes *out* en el acondicionador de tráfico CBM y la interacción de éste con la gestión de cola RIO no tiene ningún efecto sobre las prestaciones en términos de garantías de contratos. Como ilustran las figuras, los contratos se garantizan al 100%. CBM se presenta por tanto como un avance importante para lograr un servicio asegurado con las suficiente garantías, gracias al control que ejerce sobre la generación de paquetes *out*.

Tabla 4.1 Valores de los umbrales *max* y *min* que se utilizan para CBM en cada escenario sobre la topología de la Fig. 4.10

Escenario	A	B	C	D	E
Capacidad Enlace (Mbps)	33	33	33	33	33
Contratos (Mbps) de s_0 a s_7	2,5	1-1-2-2-3-3-4-4	2,5	1-1-2-2-3-3-4-4	4-4-3-3-2-2-1-1
Σcontratos Mbps)	20	20	20	20	20
BW_{exceso} (Mbps)	13	13	13	13	13
RTT (ms) de s_0 a s_7	50	50	10-20-30-40-50-60-70-80	10-20-30-40-50-60-70-80	10-20-30-40-50-60-70-80
RTT_{medio} (ms)	50	50	45	45	45
max (nº paquetes *out*)	9	9	8	8	8
min (nº paquetes *out*)	5	5	4	4	4

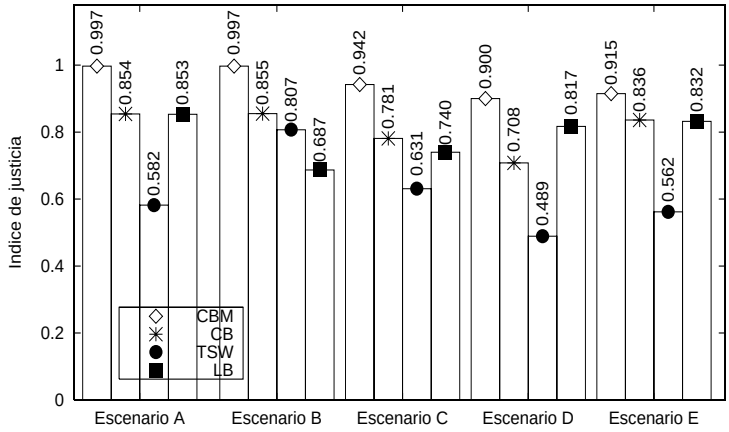


Fig. 4.12 Índices de justicia alcanzados en los escenarios A, B, C, D y E dentro de la topología de un único cuello de botella con los acondicionadores de tráfico *Counters-Based Modified* (CBM), *Counters-Based* (CB), *Time Sliding Window* (TSW) y *Leaky Bucket* (LB)

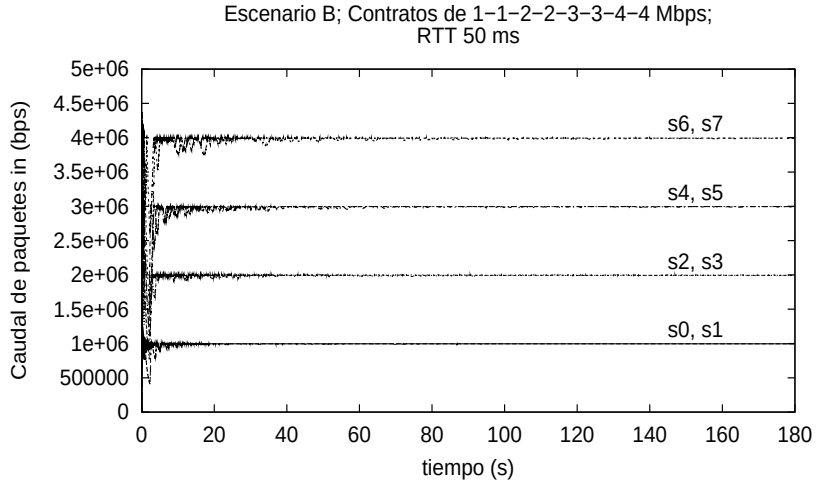


Fig. 4.13 Caudal de paquetes *in* en el escenario B en la topología de un solo cuello de botella

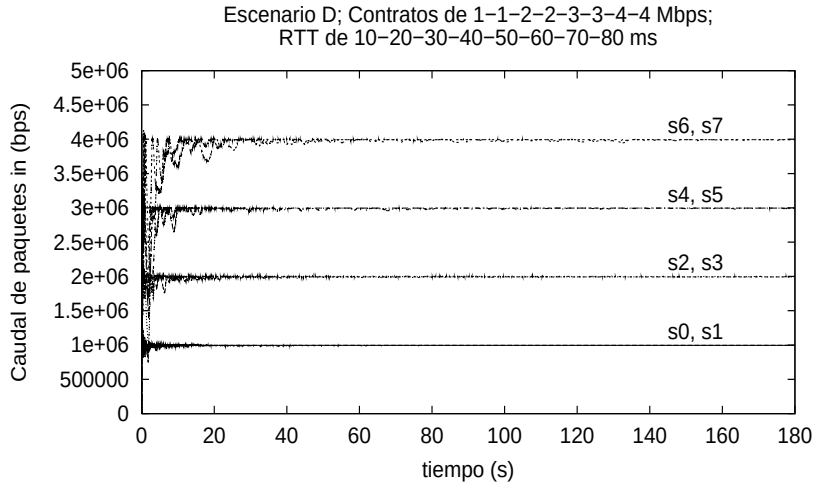


Fig. 4.14 Caudal de paquetes *in* en el escenario D en la topología de un solo cuello de botella

4.4.1.2. Simulaciones con tráfico asegurado y *best-effort*

En este caso, fuentes *best-effort* van a competir por los recursos disponibles con fuentes que generan tráfico asegurado. Normalmente, las implementaciones de DiffServ no mezclan tráfico *best-effort* y tráfico asegurado en una misma cola de un nodo de la red, sino que sitúan los paquetes de estas dos clases de tráfico en colas distintas. Tampoco es nuestra intención implementar el servicio de esta manera. El caso de estudio que proponemos se corresponde con los siguientes posibles contextos. En primer lugar, nos podemos encontrar en una situación transitoria en la que el ISP no puede reasignar las conexiones ni configurar una arquitectura más específica, viéndose obligado a mezclar ambos tráficos. En segundo lugar, podemos tener una situación en la que el acondicionador de tráfico falla y marca todos los paquetes como *out*, o simplemente no marca ningún paquete.

Los resultados se han obtenido empleando la topología descrita en la Fig. 4.10. Ocho fuentes TCP Reno, donde las cuatro fuentes s_0 a s_3 contratan un servicio asegurado y las cuatro fuentes s_4 a s_7 tienen un servicio *best-effort*. El hecho de disponer de un servicio *best-effort* significa que estas fuentes no tienen contratos y los paquetes se tratan como *out*. Efectuamos simulaciones en los cinco escenarios descritos en la Sección 4.3 con pequeñas modificaciones que se indican a continuación.

- En el escenario A las fuentes con servicio asegurado tienen un contrato de 5 Mbps. El RTT es de 50 ms para todas las fuentes incluidas las *best-effort*. De las ecuaciones (4.1) y (4.2) obtenemos unos umbrales *max* y *min* de 9 y 5 paquetes *out* respectivamente:

$$max = \left\lceil \frac{bandwdith_{exceso} \cdot RTT_{medio}}{MSS} \right\rceil = \left\lceil \frac{13\ Mbps \cdot 50\ ms}{9.188\ bytes} \right\rceil = 9$$

$$min = \left\lceil \frac{max}{2} \right\rceil = 5$$

  Idealmente, si repartimos los 13 Mbps de ancho de banda no contratado entre las ocho fuentes cada conexión debería obtener 1,625 Mbps del ancho de banda en exceso. La Fig. 4.15 detalla el caudal que obtienen las fuentes *best-effort*. Vemos como todas consiguen prácticamente la misma porción de ancho de banda en exceso tras un pequeño intervalo transitorio. Según el índice de justicia representado en la Fig. 4.20 (f = 0,937), las fuentes aseguradas y las *best-effort* obtienen prácticamente la misma cantidad de ancho de banda en exceso.

- El escenario B es igual al escenario A, pero las cuatro fuentes con servicio asegurado contratan un ancho de banda de 4, 5, 6 y 7 Mbps cada una. Aquí los umbrales *max* y *min* son de 8 y 4 paquetes respectivamente ($bandwidth_{exceso}$ = 11 Mbps; RTT_{medio} = 50 ms; MSS = 9.188 bytes). El valor del índice de justicia en este escenario es de 0,864. La Fig. 4.16 muestra el caudal alcanzado por las fuentes *best-effort*. De nuevo, apreciamos que las fuentes *best-effort* no perjudican a las aseguradas y logran una porción de ancho de banda en exceso casi ideal.

- En el escenario C las fuentes con servicio asegurado tienen un contrato de 5 Mbps. Cada conexión asegurada (s_0 a s_3) tiene un RTT de 10, 20, 30 y 40 ms respectivamente. Las conexiones *best-effort* (s_4 a s_7) presentan un RTT de 50, 60, 70 y 80 ms respectivamente. Los límites *max* y *min* toman un valor de 8 y 4 paquetes cada uno ($bandwidth_{exceso}$ = 13 Mbps; RTT_{medio} = 45 ms; MSS = 9.188 bytes). Con el amcho de banda en exceso disponible, el *goodput* ideal de las conexiones *best-effort* es de 1,625 Mbps. Según los resultados de simulación (Fig. 4.17), las fuentes *best-effort* alcanzan un *goodput* cercano al ideal. La diferencia entre los caudales obtenidos por cada una de estas fuentes no supera los 0,5 Mbps. Apenas es apreciable el impacto

sobre la distribución del ancho de banda en exceso por tener RTT dispares en el agregado, lo que se refleja en un índice de justicia cercano a 1 (f = 0,847). Véase Fig. 4.20.

- Por último, los escenarios más heterogéneos D y E también presentan unos valores del índice de justicia por encima de 0,8. En el escenario D, las cuatro fuentes aseguradas (s_0 a s_3) tienen contratos de 4, 5, 6 y 7 Mbps, y un RTT de 10, 20, 30 y 40 ms. El RTT de las fuentes *best-effort* (s_4 a s_7) es de 50, 60, 70 y 80 ms. El escenario E difiere del D únicamente en que los contratos de las fuentes aseguradas son en este caso de 7, 6, 5 y 4 Mbps. Los umbrales *max* y *min* quedan establecidos a 7 y 4 paquetes en ambos escenarios ($bandwidth_{exceso}$ = 11 Mbps; RTT_{medio} = 45 ms; MSS = 9.188 bytes). Las Fig. 4.18 y 4.19 reflejan el caudal alcanzado por las fuentes *best-effort* en los escenarios D y E, nuevamente, acercándose a la situación ideal.

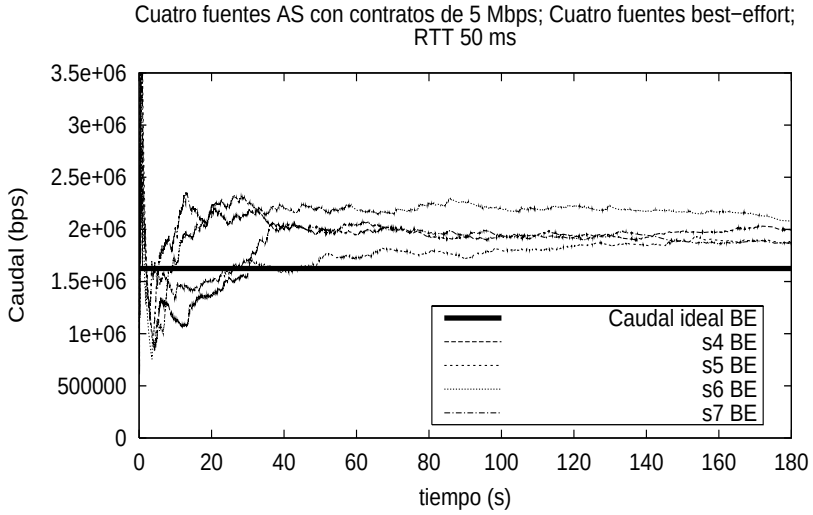


Fig. 4.15 *Goodput* (bps) de las fuentes *best-effort* (s_4 a s_7) en el escenario A

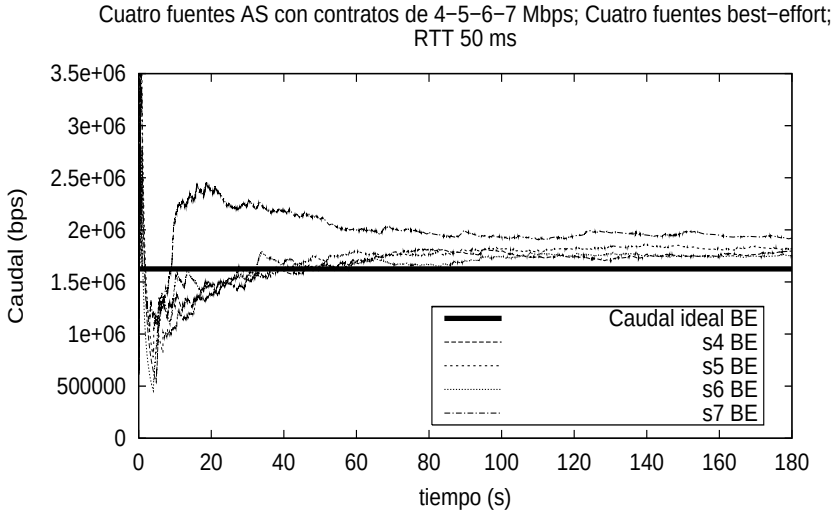


Fig. 4.16 *Goodput* (bps) de las fuentes *best-effort* (s_4 a s_7) en el escenario B

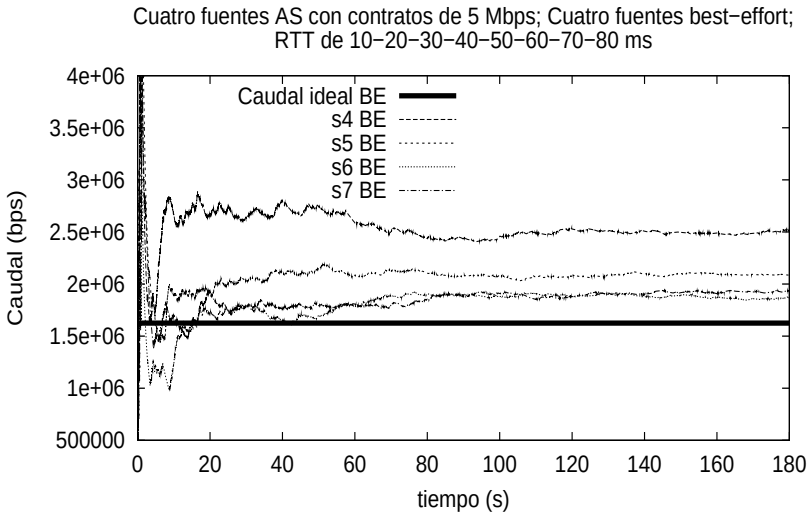


Fig. 4.17 *Goodput* (bps) de las fuentes *best-effort* (s_4 a s_7) en el escenario C

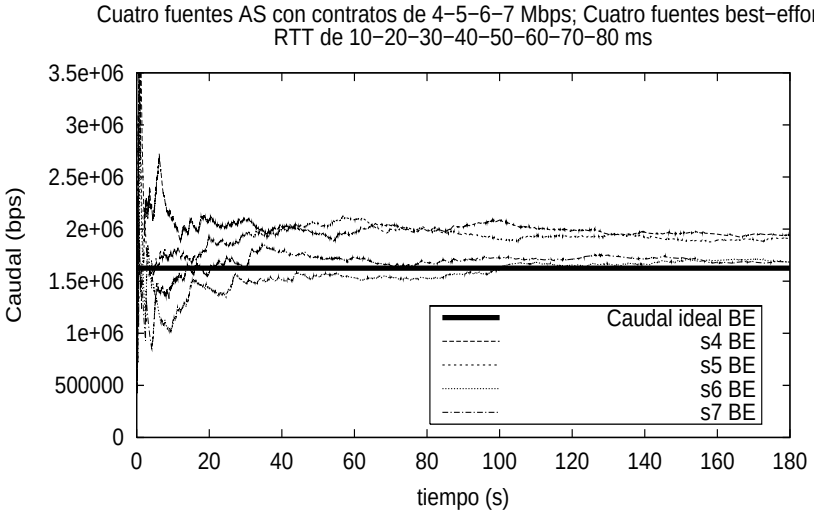


Fig. 4.18 *Goodput* (bps) de las fuentes *best-effort* (s_4 a s_7) en el escenario D

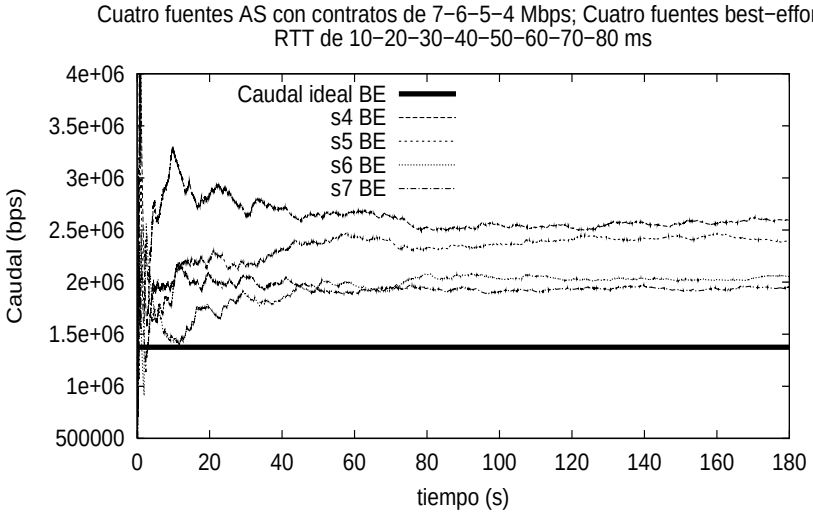


Fig. 4.19 *Goodput* (bps) de las fuentes *best-effort* (s_4 a s_7) en el escenario E

La Fig. 4.20 ilustra a través de los valores del índice de justicia el buen comportamiento de CBM respecto a la distribución de los recursos no contratados aún cuando coexisten fuentes aseguradas y *best-effort*. La Fig. 4.21 representa el caudal de paquetes *in* en el escenario D (caso más desfavorecedor) para remarcar que las fuentes *best-effort* tampoco afectan a las garantías de contratos de los usuarios con servicio asegurado.

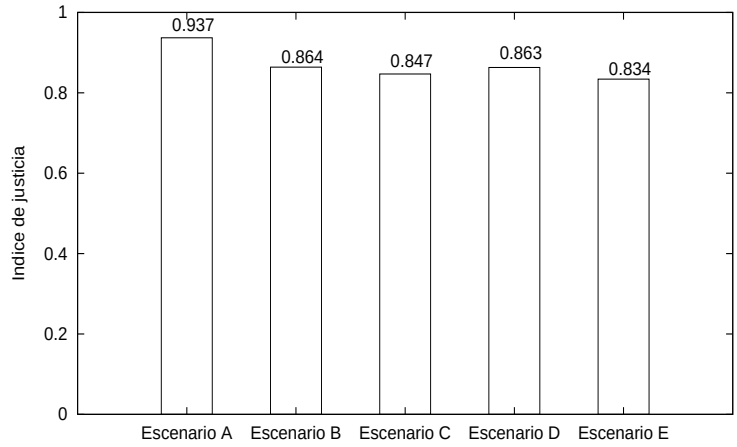


Fig. 4.20 Índice de justicia en los escenarios A, B, C, D y E donde coexisten fuentes aseguradas y *best-effort*

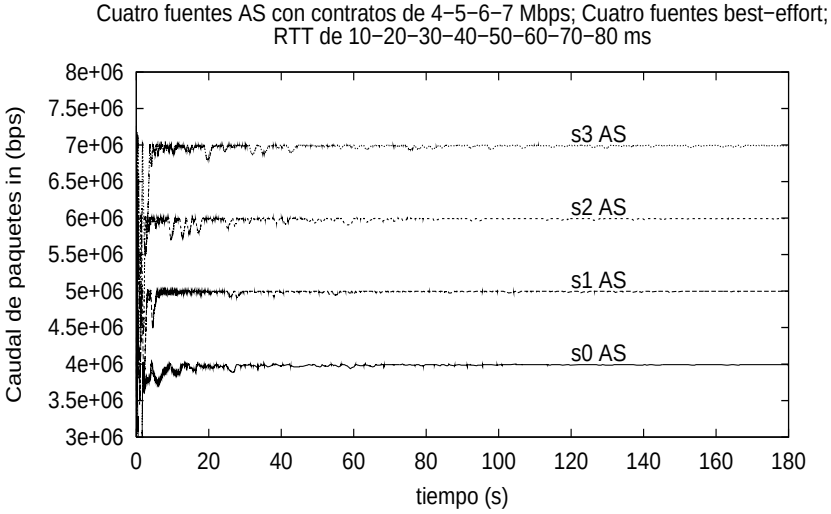


Fig. 4.21 Caudal de paquetes *in* en el escenario D con fuentes aseguradas y *best-effort*

Volviendo al origen de esta sección, en periodos transitorios en los que sea necesario acomodar ambos tipos de tráfico, asegurado y *best-effort*, en la misma cola por necesidades de arquitectura, o nos encontremos con acondicionadores de tráfico que no funcionan correctamente y sólo marcan paquetes como *out*, el ISP deberá tener las suficientes garantías de que las fuentes *best-effort* (los paquetes *out*) no agotarán los recursos. Esto requiere el uso de herramientas que limiten el efecto de este tipo de conexiones. En esta sección queda demostrado que la pareja CBM-RIO es suficientemente robusta como para proporcionar un servicio resistente, en el que a pesar de las dificultades las fuentes aseguradas alcanzan sus contratos y se benefician del ancho de banda en exceso. Incluso en escenarios heterogéneos (coexistencia de dos tipos de tráfico, diferencias de contratos y de RTT), un buen acondicionador de tráfico como CBM contribuye a ofrecer QoS extremo a extremo, asegurando los contratos de los

usuarios finales y distribuyendo de modo justo entre las distintas fuentes el ancho de banda no contratado.

4.4.2. Topología de tres nodos

En esta sección ofrecemos un aproximación más cercana a un contexto real. De los resultados logrados en esta sección podremos adquirir un mejor conocimiento sobre las posibles aplicaciones del acondicionador de tráfico CBM en Internet. Trabajos anteriores en esta misma línea concluyen que no es factible un servicio con estrictas garantías para las fuentes TCP [IBAN98] [REZE99]. Aunque las últimas investigaciones presentan resultados más favorables, por ejemplo [NAND00], una implementación factible de acondicionador de tráfico no parece obvia. Dada la superioridad de CBM frente a TSW y LB, en esta sección sólo presentamos los resultados obtenidos con CBM.

Analizaremos las situaciones descritas en las Fig. 4.22, 4.23, 4.26 y 4.27. En principio, los nodos etiquetados como E_1, E_2 y E_3 emplean el mecanismos RIO con parámetros [40/70/0,02] para los paquetes *in* y [10/40/0,2] para los paquetes *out*. En algunos casos de estudio, el nodo E_1 o el nodo E_2 no realizará diferenciación de servicios, utilizando un mecanismos RED con parámetros [10/40/0,2]. Como se indica en la Sección 4.3, seguimos las recomendaciones de [FLOY93]. Los agregados de cada nodo están compuestos de tráfico asegurado o de una mezcla de tráfico asegurado y *best-effort*, de nuevo dependiendo del caso de estudio. Los distintos casos de estudio nos ayudarán a evaluar la robustez del algoritmo CBM en combinación con RIO.

Los acondicionadores CBM se sitúan junto a las fuentes de tráfico que contratan un servicio asegurado pero fuera del alcance de los usuarios finales. De otro modo consideramos que las fuentes son *best-effort* y sus paquetes se tratan como *out* sin recibir ningún acondicionado. Los umbrales *max* y *min* (ec. 4.1 y 4.2) de CBM se incluyen en la Tabla 4.2. Calculamos estos valores para E_1 y E_2 tal y como se describe en la Sección 4.2 suponiendo que el nodo E_1 no sabe de la existencia de E_2 y viceversa. A continuación presentamos los resultados obtenidos.

4.4.2.1. Simulaciones con tráfico asegurado

Este primer caso (caso 1) se desarrolla en una topología de tres nodos RIO y ocho fuentes TCP Reno que contratan un servicio asegurado (Fig. 4.22). Las fuentes generan tráfico a la velocidad del enlace, establecida a 33 Mbps. Las diferentes características de los escenarios A, B, C, D y E descritos en la Sección 4.3 se incluyen en la Tabla 4.2 junto con los límites *max* y *min*.

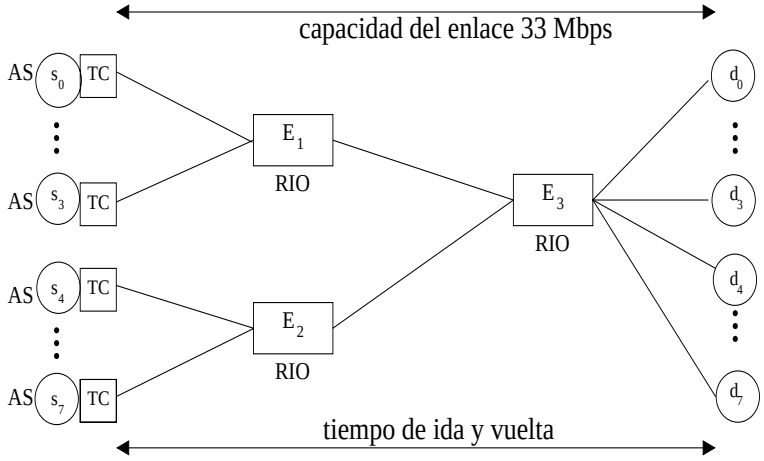


Fig. 4.22 Topología de tres nodos en el caso 1 (TC≡Acondicionador de tráfico)

Tabla 4.2 Contratos, tiempos de ida y vuelta y límites *max-min* de las fuentes TCP Reno para los casos 1 y 2 (los datos se corresponden con las fuentes s_0 a s_7 respectivamente)

Escenarios	Contrato (Mbps)	RTT(ms)	E_1 s_0 a s_3		E_2 s_4 a s_7	
			max	*min*	*max*	*min*
A	2,5	50	16	8	16	7
B	1-1-2-2-3-3-4-4	50	19	10	13	7
C	2,5	10 a 80	8	4	21	11
D	1-1-2-2-3-3-4-4	10 a 80	10	5	17	9
E	4-4-3-3-2-2-1-1	10 a 80	7	4	24	12

Los resultados obtenidos mediante simulación se muestran en la Tabla 4.3. La pareja CBM-RIO permite a los usuarios alcanzar los contratos establecidos a pesar de la diversidad presente en cada escenario. El hecho de descartar paquetes *out* de modo probabilístico en el acondicionador de tráfico, hace que las fuentes TCP se adapten a las condiciones de la red. Una vez garantizados los contratos cada conexión obtiene una porción similar del ancho de banda en exceso como indican los índices de justicia de la última fila de la Tabla 4.3.

Tabla 4.3 Caudal alcanzado de paquetes *in* (Mbps) e índices de justicia en los cinco escenarios del caso 1

Fuente	Escenario A	Escenario B	Escenario C	Escenario D	Escenario E
0	2,50	0,99	2,50	0,99	3,85
1	2,49	1,00	2,50	0,99	3,99
2	2,49	1,99	2,49	2,00	3,00
3	2,49	1,99	2,49	2,00	2,99
4	2,49	2,99	2,50	2,99	1,99
5	2,49	2,99	2,50	2,99	1,99
6	2,49	3,99	2,49	3,95	1,00
7	2,50	3,99	2,48	3,70	1,00
Índice Justicia	0,998	0,857	0,907	0,623	0,803

4.4.2.2. Simulaciones con tráfico asegurado y un nodo sin diferenciación de servicios

Según [RFC2475], es impredecible conocer el resultado de trabajar en un dominio DiffServ si alguno de los nodos no implementa esta arquitectura (nodo al que denominamos *non-DiffServ compliant*). Hasta la fecha no se ha realizado ningún estudio en este sentido. Pueden ser varias las razones para encontrar un nodo *non-DiffServ compliant*, por ejemplo que un nodo de la red falle y tenga que ser sustituido por otro que no dispone de herramientas para implementar DiffServ (por ejemplo viéndose obligado a utilizar RED clásico). Es por esto que el presente caso de estudio es interesante para un ISP porque si logramos buenos resultados, el ISP puede ofrecer un servicio asegurado con una implementación más sencilla (o incluso más económica). La Fig. 4.23 detalla la topología con las ocho fuentes TCP Reno que contratan un servicio asegurado, donde las fuente s_0 a s_3 confluyen en el nodo RIO E_1 y las fuentes s_4 a s_7 en el nodo RED E_2 (caso 2a). Hemos realizado simulaciones en todos los escenarios expuestos en la Tabla 4.2.

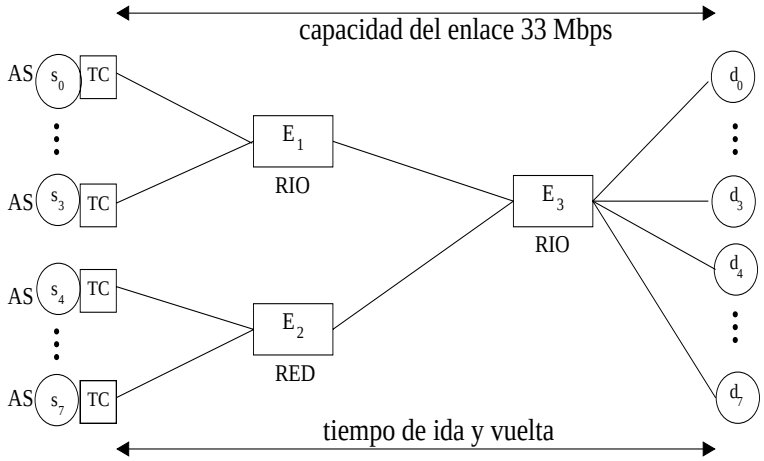


Fig. 4.23 Topología de tres nodos en el caso 2 con el nodo E_2 sin diferenciación de servicios (caso 2a)

Los resultados revelan que los contratos quedan garantizados (Fig. 4.24 y 4.25). Según se aprecia en la Fig. 4.25, debe pasar un pequeño tiempo transitorio de unos 40 ms aproximadamente antes de alcanzar los contratos para el escenario D. Recordemos que este escenario es el más desfavorable para conseguir un buen servicio puesto que el nodo E_2 recoge el tráfico proveniente de las fuentes s_4 a s_7, que poseen mayores contratos y por tanto generan más paquetes marcados como *in*. Además, estas fuentes también tienen los RTT más elevados según las características del escenario D. Dado que el nodo E_2 no hace diferenciación de servicios (sólo implementa RED), descartará paquetes *in* o *out* indistintamente. Como el contrato debe quedar garantizado mediante el caudal de paquetes *in*, descartar estos paquetes puede retardar la consecución de dicho contrato. Para el resto de escenarios (véase la Tabla 4.4) los contratos se alcanzan al 100%.

Cuando no hay diferenciación de servicios en el nodo E_2, el índice de justicia alcanzado en las simulaciones está por encima de 0,8 excepto para el escenario D (observe la última fila de la Tabla 4.4). El origen de este bajo valor del índice de justicia es debido, como dijimos en el párrafo anterior, a tener un nodo sin diferenciación de servicios en el que confluyen las conexiones con mayores contratos y RTT más altos. En los demás escenarios, la distribución del ancho de banda en exceso se puede considerar como justa. De nuevo, gracias al descarte probabilístico de paquetes *out* en los acondicionadores de tráfico, podemos controlar el ancho de banda no contratado que cada fuente obtiene sin interferir en el caudal de paquetes *in*.

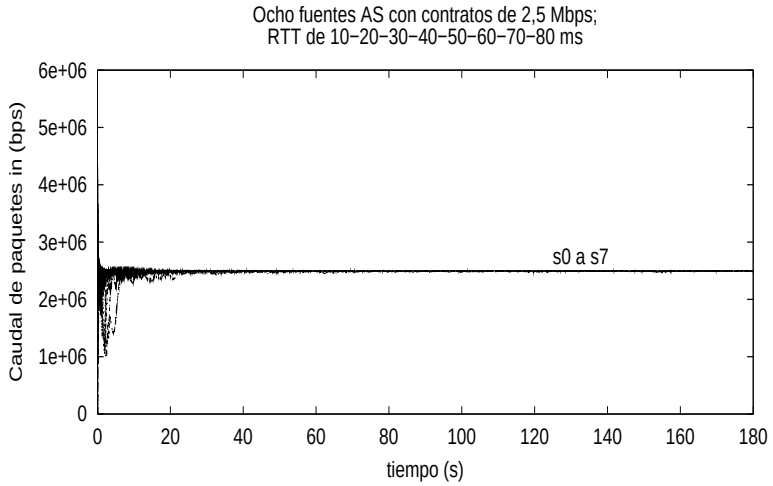


Fig. 4.24 Caudal de paquetes *in* obtenido en el caso 2a (RED en el nodo E_2) utilizando CBM. Simulaciones realizadas en el escenario C

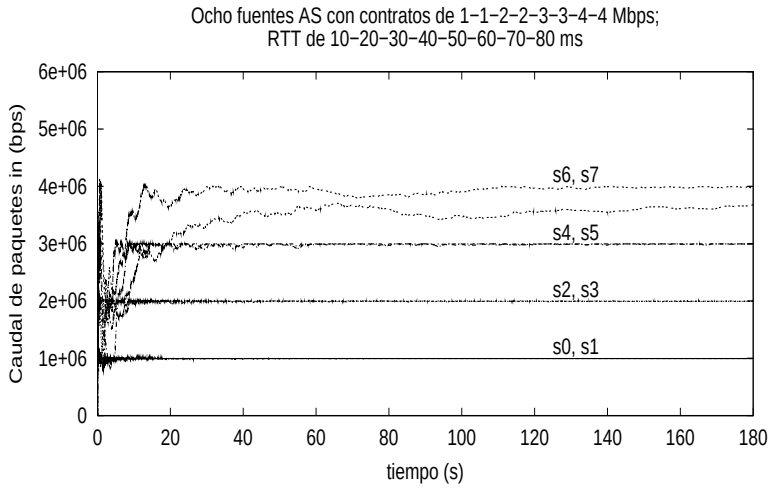


Fig. 4.25 Caudal de paquetes *in* en el caso 2a (RED en el nodo E_2) utilizando CBM. Simulaciones realizadas en el escenario D

Tabla 4.4 Caudal de paquetes *in* (Mbps) e índices de justicia en los cinco escenarios del caso 2a (RED en el nodo E_2)

Fuente	Escenario A	Escenario B	Escenario C	Escenario D	Escenario E
0	2,49	1,00	2,49	1,00	3,99
1	2,50	1,00	2,50	0,99	3,99
2	2,49	1,99	2,50	1,99	2,99
3	2,50	1,99	2,49	2,00	3,00
4	2,49	2,99	2,49	2,99	1,99
5	2,49	2,99	2,49	2,99	1,99
6	2,49	4,00	2,49	3,99	1,00
7	2,50	3,99	2,49	3,44	1,00
Índice Justicia	0,997	0,848	0,906	0,612	0,803

Para completar el estudio de una topología con un nodo *non-DiffServ compliant*, llevamos a cabo nuevas simulaciones utilizando el esquema RED en el nodo E_1, y RIO en los nodos E_2 y E_3 (caso 2b). Puesto que el nodo E_1 no recibe las conexiones de mayor contrato ni las de RTT más elevado, son de esperar mejores resultados. Según refleja la Tabla 4.5, las prestaciones mejoran levemente. En el escenario A se mantiene el mismo índice de justicia. En los escenarios B y C el valor del índice de justicia aumenta ligeramente. En el escenario D, se consigue una distribución del ancho de banda no contratado más equitativa, evidenciada por un incremento de casi una décima en el valor del índice de justicia. Por último en el escenario E obtenemos prácticamente las mismas prestaciones.

En conclusión, y según muestran las simulaciones, gracias a CBM es posible ofrecer un servicio asegurado a los usuarios finales, garantizándoles sus contratos y un uso equitativo del ancho de banda no contratado aún cuando un nodo de la red no dispone de herramientas para hacer diferenciación de servicios.

Tabla 4.5 Caudal de paquetes *in* (Mbps) e índices de justicia en los cinco escenarios del caso 2b (RED en el nodo E_1)

Fuente	Escenario A	Escenario B	Escenario C	Escenario D	Escenario E
0	2,49	0,99	2,50	1,00	3,83
1	2,50	1,00	2,50	1,00	3,97
2	2,49	1,99	2,50	2,00	2,99
3	2,50	1,99	2,49	1,99	3,00
4	2,49	2,99	2,50	2,99	1,99
5	2,49	2,99	2,49	2,99	1,99
6	2,49	3,99	2,49	3,96	1,00
7	2,50	3,99	2,49	3,73	1,00
Índice Justicia	0,997	0,862	0,913	0,695	0,794

4.4.2.3. Simulaciones con tráfico asegurado y tráfico *best-effort*

En esta sección, pretendemos examinar la robustez de la propuesta CBM. Al igual que hicimos en la topología de un único cuello de botella (Sección 4.4.1.2), estudiamos el efecto que tienen fuentes *best-effort* sobre las conexiones de tráfico asegurado. No hay que olvidar que las fuentes *best-effort* sólo generan paquetes *out* porque no contratan un ancho de banda mínimo. Para estudiar este efecto, ambos tipos de tráfico se van a almacenar en la misma cola dentro de los nodos de la red. Como dijimos con anterioridad esta situación puede caracterizar dos escenarios, uno en el que el ISP se ve obligado a mezclar ambos tipos de tráfico en una misma cola en el enrutador por problemas de arquitectura; o bien, otro en el que algunos acondicionadores fallan y generan exclusivamente paquetes *out*.

Para esta sección, realizamos las simulaciones con tráfico generado por doce fuentes TCP Reno. Las fuentes s_0 a s_3 y s_6 a s_9, contratan un servicio asegurado. Las fuentes s_4, s_5, s_{10} y s_{11} serán fuentes *best-effort* (Fig. 4.26). Las simulaciones se llevaron a cabo en distintos escenarios como muestra la Tabla 4.6. Es importante destacar que aunque confluyen el mismo número de fuentes aseguradas y best-*effort* en cada nodo frontera, E_1 y E_2, la carga de la red no está balanceada ya que los contratos de cada una de las fuentes aseguradas es distinto. Es de esperar que las fuentes *best-effort* traten de obtener tantos recursos como les sea posible. En una primera aproximación, los tres nodos de la topología implementan RIO (caso 3a).

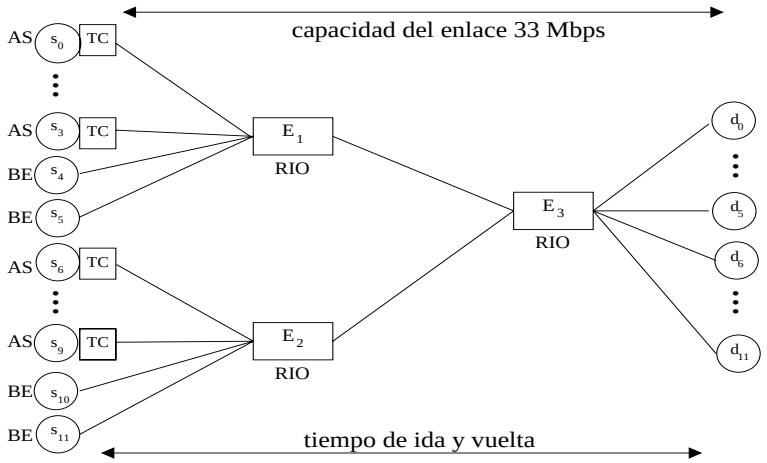


Fig. 4.26 Topología de tres nodos RIO (caso 3a)

Tabla 4.6 Contratos, tiempos de ida y vuelta y límites *max-min*, si aplica, de las fuentes TCP Reno para el caso 3 (los datos se corresponden con las fuentes s_0 a s_{11} respectivamente)

Escenarios	Contrato excepto para *best-effort* (Mbps)	RTT(ms)	E_1 s_0 a s_3		E_2 s_6 a s_9	
			max	*min*	*max*	*min*
A	2,5	50	16	8	16	8
B	1-1-2-2-3-3-4-4	50	19	10	13	7
C	2,5	10 a120	11	6	30	15
D	1-1-2-2-3-3-4-4	10 a120	13	7	25	13
E	4-4-3-3-2-2-1-1	10 a 120	10	5	35	18

Según los resultados obtenidos en las simulaciones, tres de las conexiones aseguradas no alcanzan el 100% del contrato con sus respectivos caudales de paquetes *in* (Tabla 4.7). Estas fuentes son s_8 y s_9 en el escenario D, con contratos de 4 Mbps y RTT de 90 y 100 ms respectivamente, y la fuente s_0 en el escenario E, con contrato de 4 Mbps y RTT de 10 ms. Debido a las sustanciales diferencias entre retardos y contratos entre conexiones, junto con el hecho de tener tráfico del tipo asegurado y *best-effort* en una misma cola de los nodos de la red, no es posible garantizar de modo riguroso los contratos de todas las fuentes para todos los escenarios. Si el ISP considera que es necesario que el tráfico asegurado y el *best-effort* compartan la misma cola durante algo más que un tiempo transitorio serán imprescindibles mejoras en la implementación. Aún así, observamos que en un caso peor, los contratos quedan garantizados en un 70%. Desde nuestro punto de vista, ofrecer un servicio asegurado de estas características es un claro avance en el desarrollo de los Servicios Diferenciados.

Tabla 4.7 Caudal de paquetes *in* (Mbps) e índices de justicia en los cinco escenarios del caso 3a (todos los nodos son RIO)

Fuente	Escenario A	Escenario B	Escenario C	Escenario D	Escenario E
0	2,49	1,00	2,49	1,00	3,16
1	2,50	1,00	2,50	1,00	4,00
2	2,49	2,00	2,49	2,00	2,99
3	2,49	1,99	2,49	2,00	2,99
6	2,50	2,99	2,49	2,96	1,99
7	2,49	2,99	2,49	2,93	1,99
8	2,49	3,98	2,49	2,99	1,00
9	2,49	3,99	2,49	2,60	1,00
Índice Justicia	0,843	0,732	0,710	0,615	0,784

Consideremos una situación extrema, en la que además de mezclar dos tipos de tráfico distintos (asegurado y *best-effort*) en una misma cola, uno de los tres nodos no dispone de herramientas para implementar diferenciación de servicios (caso 3b). El nodo que no opera con DiffServ será el E_2. La Fig. 4.27 describe la topología empleada para las simulaciones. De los resultados extraídos, vemos como incluso en esta situación las fuentes con un servicio asegurado logran alcanzar sus respectivos contratos (Tabla 4.8). De nuevo, experimentamos ciertas dificultades en el escenario D, donde el caudal de paquetes *in* de las fuentes s_8 y s_9 permanece por debajo del contrato establecido. Así como en el escenario E, donde la fuente s_0 no alcanza el 100% del contrato con el caudal de paquetes *in*. A pesar de las inhóspitas características de esta topología (caso 3b), hay que remarcar que en general los resultados obtenidos son buenos. Las simulaciones realizadas bajo las mismas circunstancias, pero

empleando RED en el nodo E_1 y RIO en los nodos E_2 y E_3 conducen a los mismos resultados en términos de garantías de contrato.

Nos detenemos ahora a examinar cómo se realiza el reparto del ancho de banda no contratado entre las distintas fuentes cuando coexisten tráfico asegurado y *best-effort*. El uso conjunto del acondicionador CBM para las fuentes con un servicio asegurado y RIO en los nodos de la red, favorece la generación de menos paquetes *out* por parte de las fuentes *best-effort*. Es por este motivo, por el que ante condiciones adversas como las planteadas en esta sección el índice de justicia se mantiene generalmente por encima de 0,7 puntos (véanse las últimas filas de las Tablas 4.7 y 4.8). Lo que significa que las fuentes *best-effort* consumen sólo una porción del ancho de banda no contratado, permitiendo que las fuentes aseguradas también obtengan una porción del mismo. En la Fig. 4.28 se muestra una comparativa de los índices de justicia obtenidos en los cinco escenarios (A, B, C, D y E) de cada uno de los cinco casos de estudio (caso1, caso 2a, caso 2b, caso 3a y caso 3b).

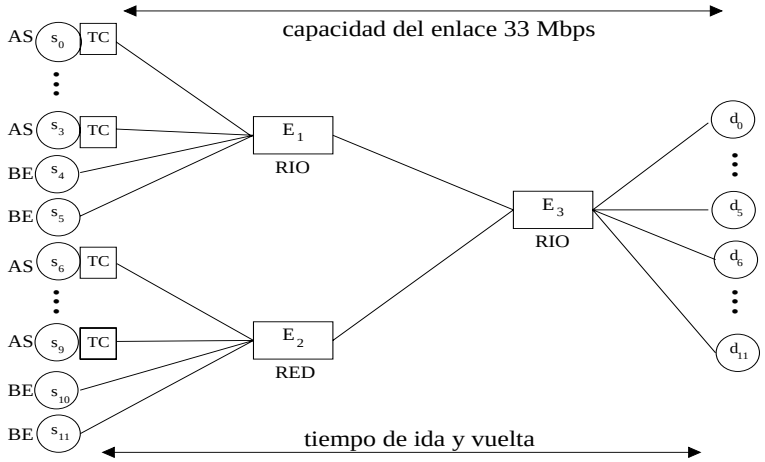


Fig. 4.27 Topología de tres nodos en el caso 3b (RED en nodo E_2) empleando CBM

Tabla 4.8 Caudal de paquetes *in* (Mbps) e índices de justicia en los cinco escenarios del caso 3b (RED en el nodo E_2)

Fuente	Escenario A	Escenario B	Escenario C	Escenario D	Escenario E
0	2,50	1,00	2,49	1,00	3,30
1	2,49	0,99	2,50	1,00	3,99
2	2,49	1,99	2,49	1,99	2,99
3	2,49	1,99	2,50	2,00	2,99
6	2,49	2,99	2,49	2,90	1,99
7	2,49	2,99	2,49	2,70	1,99
8	2,50	3,97	2,50	2,70	1,00
9	2,49	3,99	2,49	2,60	1,00
Índice Justicia	0,845	0,730	0,710	0,615	0,784

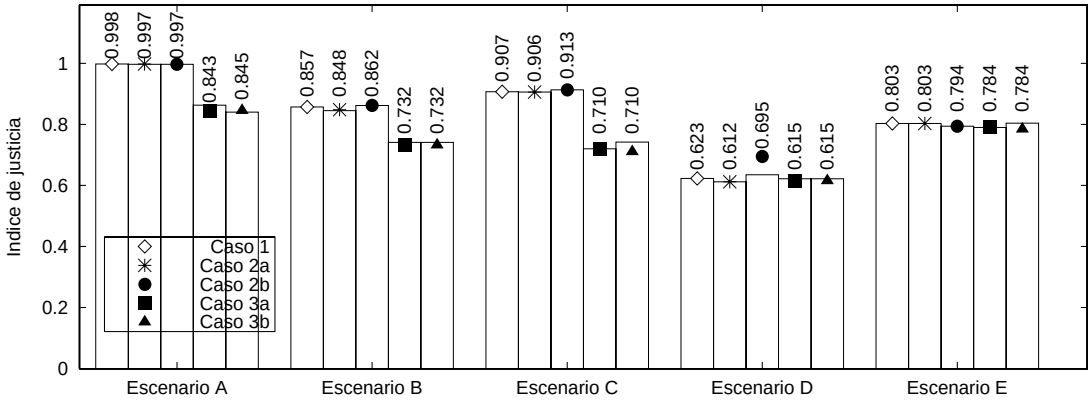


Fig. 4.28 Cuadro comparativo de los índices de justicia de todos los escenarios para cada uno de los casos de estudio analizados en la topología de tres nodos

4.4.2.4. Reconfiguración de los parámetros de RIO

Como hemos visto en el apartado anterior, la robustez de CBM se veía afectada en condiciones extremas en las que dos tipos de tráfico compartían una misma cola, viéndose reflejado en la imposibilidad de lograr los contratos fijados con una garantía del 100% en todos los casos. También apuntamos que esta reducción de prestaciones era sólo palpable en escenarios con grandes variaciones de RTT y de contratos entre las fuentes del agregado. En definitiva, siendo necesario adoptar medidas alternativas si deseábamos implementar un servicio asegurado robusto para toda la Internet.

Una de esas posibles medidas a llevar a cabo es reconfigurar los parámetros de RIO si las condiciones de la red así lo requieren, por ejemplo ante el hecho de mezclar dos tipos de tráfico en una misma cola. Como prueba, modificamos los parámetros de configuración de RIO del nodo E_3 con el fin de acomodarlo a las características de la topología ilustrada en la Fig. 4.29. Ésta consta de seis fuentes TCP Reno con un servicio asegurado que confluyen en E_1, y seis fuentes *best-effort* que convergen en el nodo E_2. El motivo de emplear esta topología, distinta de las empleadas hasta el momento, es evaluar el comportamiento del nodo E_3 con la nueva configuración de parámetros para un caso peor.

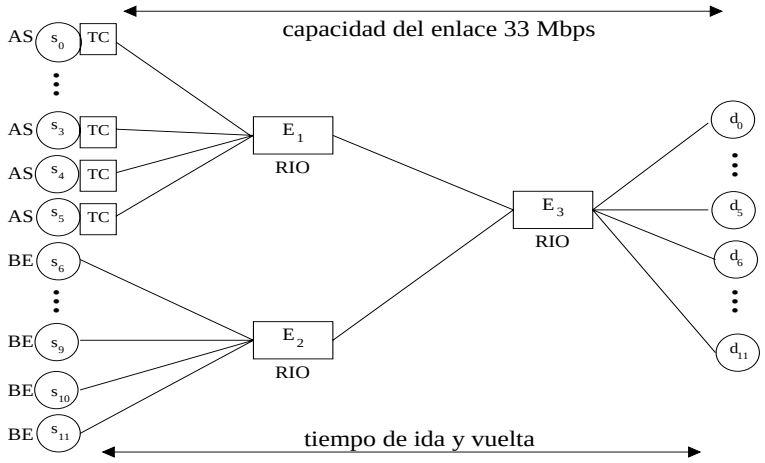


Fig. 4.29 Topología de tres nodos en la que E_1 sólo recibe tráfico asegurado, E_2 sólo gestiona tráfico *best-effort* y los parámetros de RIO de E_3 han sido reconfigurados

Los cinco escenarios simulados se resumen en la Tabla 4.9. Los agregados de ambos nodos, E_1 y E_2, se combinan en el nodo E_3, cuyos nuevos parámetros son [110/140/0,02] para los paquetes *in* y [40/80/0,8] para los paquetes *out*. Al aceptar más paquetes *in* en la cola RIO e incrementar la probabilidad de descarte de los paquetes *out* intentamos garantizar los contratos de todas las fuentes y evitar que las fuentes *best-effort* consuman más ancho del banda del que les corresponde en un reparto justo (equitativo).

En la Tabla 4.10 presentamos los resultados obtenidos tras reconfigurar el nodo E_3. La Tabla 4.11 incluye los resultados que se obtienen sin realizar ninguna variación en los parámetros de RIO, es decir, manteniendo los parámetros de simulaciones previas. Según la Tabla 4.10, vemos como en el caso peor, el caudal de paquetes *in* garantiza un 65% del contrato. En lo que concierne a la distribución del ancho de banda en exceso, los índices de justicia alcanzados son superiores a los que se obtienen si no se reconfigura el nodo E_3. Al actualizar los parámetros de RIO, conseguimos que el nodo E_3 sea capaz de gestionar la gran cantidad de paquetes *out* que le llegan desde el nodo E_2.

Tabla 4.9 Contratos, tiempos de ida y vuelta y límites max-min, si aplica, de las fuentes TCP Reno para la topología de la Fig. 4.29 (los datos se corresponden con las fuentes s0 a s11 respectivamente)

Escenarios	Contrato excepto para *best-effort* (Mbps)	RTT(ms)	E_1 s_0 a s_6		E_2 s_7 a s_{11}	
			max	*min*	*max*	*min*
A	2,5	50	13	7	-	-
B	2-2-3-3-4-4	50	11	6	-	-
C	2,5	10 a 120	9	5	-	-
D	2-2-3-3-4-4	10 a 120	8	4	-	-
E	4-4-3-3-2-2	10 a 120	8	4	-	-

Tabla 4.10 Caudal de paquetes *in* (Mbps) e índices de justicia en los cinco escenarios correspondiente a la topología de la Fig. 4.29 con actualización de parámetros RIO en el nodo E_3

Fuente	Escenario A	Escenario B	Escenario C	Escenario D	Escenario E
0	2,50	2,00	2,50	2,00	2,60
1	2,50	2,00	2,50	2,00	4,00
2	2,49	3,00	2,50	2,99	3,00
3	2,50	2,99	2,50	3,00	3,00
4	2,50	3,87	2,49	3,37	2,00
5	2,50	3,95	2,49	3,00	1,99
Índice Justicia	0,972	0,800	0,869	0,781	0,838

Tabla 4.11 Caudal de paquetes *in* (Mbps) e índices de justicia en los cinco escenarios correspondiente a la topología de la Fig. 4.29 sin actualización de parámetros RIO en el nodo E_3

Fuente	Escenario A	Escenario B	Escenario C	Escenario D	Escenario E
0	2,49	2,00	2,50	1,99	2,80
1	2,49	1,99	2,49	2,00	3,94
2	2,49	2,98	2,50	2,99	2,99
3	2,49	2,98	2,48	2,98	2,98
4	2,49	3,71	2,50	3,61	1,99
5	2,50	3,45	2,49	3,25	1,99
Índice Justicia	0,684	0,597	0,750	0,628	0,660

Queda en manos del ISP seleccionar cuál es la mejor opción. Podemos considerar que el uso complementario de los servicios expedito y asegurado de DiffServ es suficiente para hacer frente a las demandas de QoS. Esto es, el servicio expedito es el encargado de aquellas aplicaciones con requisitos estrictos de QoS, mientras que el servicio asegurado se emplea como una mejor solución que el servicio *best-effort* pero sin restricciones fuertes, por ejemplo, siendo aceptable una garantía de lograr el 75% de los contratos con el caudal de paquetes y un índice mínimo de justicia de 0,8 en situaciones imprevistas. En este caso, el acondicionador de tráfico CBM en combinación con RIO, juega un papel fundamental para conseguir los objetivos del servicio asegurado.

La otra alternativa es seguir trabajando para conseguir un servicio asegurado suficientemente robusto, que garantice los contratos al 100% y un índice de justicia cercano a 1, como para hacer frente a todas las situaciones que nos podamos encontrar en la red. Aquí, el acondicionado del tráfico se puede realizar con CBM, pues hemos comprobado que es capaz de controlar el número de paquetes *out* que las fuentes inyectan a la red. Sin embargo, hay que escoger un esquema distinto de prevención de la congestión en las colas de los nodos, pues RIO no es capaz de abordar este tipo de servicio con las suficientes garantías.

4.5. Conclusiones

En este capítulo hemos estudiado las prestaciones del acondicionador de tráfico *Counters Based Modified* (CBM). La demostrada habilidad de CBM para controlar el número de paquetes *out* que cada fuente introduce en el agregado, es la responsable de lograr una distribución más justa (más equitativa) del ancho de banda en exceso. CBM alcanza este objetivo descartando paquetes *out* en el acondicionador de tráfico, situado junto a las fuentes de tráfico pero fuera del alcance del usuario final. Los paquetes *out* se desechan con una probabilidad que depende de la cantidad de ancho de banda no contratado, de los contratos de los usuarios y de una estimación del retardo medio de las conexiones. A través de un extensivo estudio de simulación en ambientes TCP heterogéneos, hemos comprobado que el acondicionador de tráfico propuesto mejora notablemente a sus predecesores (TSW, LB y CB). En concreto, garantiza de un modo más riguroso una caudal mínimo a todas las fuentes TCP, el contrato que establecen con el ISP, y además consigue realizar un reparto justo del ancho de banda en exceso entre las distintas fuentes que componen el agregado.

En una topología de un único cuello de botella, y para diferentes escenarios con contratos y retardos variados o con coexistencia de tráfico asegurado y *best-effort*, la pareja CBM-RIO garantiza un índice de justicia en la distribución del ancho de banda en exceso por encima de 0,9. Comparado con otros acondicionadores de tráfico como TSW-RIO o LB-RIO, bajo los mismos escenarios de simulación, CBM los supera notablemente. Sin olvidar que los contratos de los usuarios se garantizan prácticamente al 100% mediante los respectivos caudales de paquetes *in*.

En un contexto más realista compuesto por tres nodos de red, desarrollamos nuevas simulaciones también para diferentes escenarios con características heterogéneas (distintos contratos, retardos variables entre las fuentes, número de fuentes variable, efecto de nodos *non-DiffServ compliant* y compartición de recursos entre tráfico asegurado y *best-effort*). Los resultados obtenidos reflejan que CBM continúa garantizando los contratos de los usuarios, donde en un caso peor se garantiza como mínimo el 70% del contrato establecido incluso cuando uno de los nodos de la red no dispone de herramientas para implementar DiffServ. Mientras que los índices de justicia alcanzados manifiestan que se realiza un reparto del ancho de banda en exceso más justo que con cualquiera de sus predecesores.

Concluimos por tanto que el acondicionador de tráfico CBM es un elemento clave a la hora de ofrecer un servicio asegurado robusto incluso ante las circunstancias más complejas, como encontrarnos con un nodo dentro del dominio DiffServ que no implementa diferenciación de

servicios, permitir a fuentes *best-effort* competir con fuentes aseguradas por los recursos de la red, etc. Por ende, constituyendo una elección factible de acondicionador de tráfico para el operador de una red que desee implementar una QoS basada en la arquitectura DiffServ.

Capítulo 5
Nuevas técnicas para el reparto proporcional del ancho de banda en exceso

5.1. Introducción

En los últimos capítulos de esta tesis nos hemos centrado en el desarrollo de acondicionadores de tráfico para el servicio asegurado AF, aportando diversas propuestas que pudieran hacer frente a los dos objetivos del AF. Estas propuestas han conseguido el primer objetivo, asegurar un caudal mínimo o CIR (normalmente el ancho de banda contratado). En lo que concierne al segundo objetivo, permitir el uso de ancho de banda en exceso si la carga de la red es baja, las soluciones planteadas hasta el momento han considerado únicamente un reparto justo como un reparto equitativo, llegando a buenos resultados en los estudios realizados al respecto (véase Capítulo 4). No obstante, como ya mencionamos en el Capítulo 1, existe cierta discrepancia en el método de distribución de este ancho de banda sobrante. Algunos autores coinciden en definir un reparto justo del ancho de banda en exceso como la distribución equitativa de éste entre las fuentes que componen el agregado. Por el contrario, otros autores entienden que un reparto justo del ancho de banda en exceso consiste en distribuir éste de modo proporcional al ancho de banda contratado por cada fuente. En este capítulo intentaremos ofrecer una solución para los objetivos del servicio AF siguiendo la segunda propuesta.

Los estudios presentados en [KUSMI00] [GEND02] [GEND03] y [SU03] presentan nuevos algoritmos con el objetivo común de ofrecer un reparto proporcional del ancho de banda en exceso dentro del servicio asegurado AF. Estas tres propuestas tienen en común el uso de tres colores para cada clase AF. En [KUSMI00], los autores proponen un esquema de marcado aleatorio por agregados, es decir, el marcado se realiza en el nodo frontera de un dominio DiffServ, y el contrato se especifica para el agregado de flujos, no para flujos individuales. La probabilidad de marcar un paquete como verde, amarillo o rojo es función de la tasa de transmisión del agregado con respecto a la tasa contratada por el agregado y la velocidad de pico. En este estudio el índice de justicia se evalúa mediante simulación con el siguiente criterio: existe justicia si ninguno de los flujos es discriminado ni recibe menos ancho de banda que los otros. El-Gendy *et al.* proponen en [GEND02] [GEND03] el algoritmo de marcado *basado en ecuación* (*Equation-Based Marking*, EBM). Este algoritmo analiza el estado actual de la red y adapta las condiciones de marcado de modo acorde. Emplea un mecanismo de control de realimentación basado en el modelo TCP, de modo que comprueba el nivel de congestión en la red observando los paquetes perdidos y entonces ajusta la tasa de envío. Aunque los resultados de simulación muestran un reparto proporcional del ancho de banda en exceso, este esquema presenta un alto nivel de complejidad, debido en gran medida a la necesidad de calcular una estima del RTT, las probabilidades de pérdidas y una probabilidad de marcado por cada paquete. El estudio desarrollado en [SU03] muestra cómo conseguir un reparto proporcional entre agregados en una red DiffServ. Al igual que en [KUSMI00], los

autores se centran en el trabajo con agregados (sólo dos en este estudio), y los resultados de simulación muestran un buen nivel de justicia para redes con una carga entre el 20% y el 70%.

Existe un único trabajo en el que se intenta conseguir un reparto proporcional del ancho de banda en exceso utilizando sólo dos niveles de precedencia. En [NANDY00], los autores presentan el acondicionador de tráfico *conocedor del tráfico* (*Traffic Aware Traffic Conditioner*, TATC). Este algoritmo transforma paquetes marcados previamente como *out* en paquetes *in*, de modo proporcional a los contratos establecidos. Lo que presumiblemente conlleva a un mayor ancho de banda para aquellas fuentes con mayor contrato. Por otra parte, algoritmos como TSW [CLARK98] o ETSW [LIN99] se utilizaron para comparar las prestaciones de EBM. Aunque ni TSW ni ETSW fueron diseñados para realizar un reparto proporcional del ancho de banda en exceso, su amplio uso los ha convertido en referencias clásicas. Esta falta de contribuciones que usen únicamente dos niveles de precedencia nos motivó a emplear la versión mejorada de TSW introducida en el Capítulo 2 como comparación con la propuesta que presentamos en este capítulo.

En este capítulo introducimos un nuevo algoritmo que ofrece un reparto proporcional del ancho de banda no contratado dentro del servicio asegurado AF. Mediante esta nueva propuesta, los usuarios finales obtienen un ancho de banda en exceso proporcional a sus contratos. El acondicionador de tráfico se sitúa junto a la fuente de tráfico (donde se establece el contrato) pero fuera del alcance del usuario final. Básicamente, este acondicionador marca paquetes IP con dos niveles de precedencia (*in* o *out*, lo que simplifica el esquema), utilizando el marcador CB (introducido en el Capítulo 2), para posteriormente aplicar la nueva función policía denominada PETER (*Proportional Excess Traffic conditionER*). La función PETER ajusta el *throughput* de las fuentes de tráfico, adaptándolas a las condiciones de la red mediante descarte de paquetes si fuera necesario. Para realizar esta tarea es necesario utilizar señalización entre el nodo frontera y los acondicionadores de tráfico. Recordemos que el nodo frontera es el más cercano al usuario final, lo que supone que esta señalización no sea un problema dado que se corresponde con el bucle de abonado (distancias cortas). Evaluaremos las prestaciones del acondicionador de tráfico PETER mediante simulaciones, en las que los resultados demostrarán que los dos objetivos del servicio AF quedan totalmente cubiertos.

El resto del capítulo se organiza de la siguiente manera. La Sección 5.2 describe el algoritmo PETER. La Sección 5.3 detalla la topología y escenarios de simulación. En la Sección 5.4 presentamos los resultados obtenidos. La Sección 5.5 da cabida a las dificultades que pueden aparecer a la hora de implementar PETER. Finalmente la Sección 5.6 resume las conclusiones más destacables.

5.2. La función policía PETER

Como se mencionó en el Capítulo 1, son cuatro las clases o instancias que presenta el servicio asegurado AF. Si se desea utilizar este servicio, los paquetes han de marcarse para pertenecer a una de estas cuatro instancias. Dentro de cada instancia hay hasta tres niveles de precedencia. En este caso nuestra función policía PETER sólo va a emplear dos niveles.

Llamemos c a la capacidad del enlace y b a la suma de todos los contratos de aquellas fuentes que se multiplexan en un mismo nodo frontera. Con tráfico de sólo dos colores, podemos definir α_{ideal} como indica la ecuación (5.1). Observe que la parte superior de esta fracción representa el ancho de banda en exceso (c-b), siendo n el número de fuentes que componen el agregado.

$$\alpha_{ideal} = \frac{capacidad\ del\ enlace - \sum_{i=1}^{n} contrato_i}{\sum_{i=1}^{n} contrato_i} = \frac{c-b}{b} \tag{5.1}$$

Suponga que en un intervalo de tiempo (t_1, t_2) medimos el ratio paquetes *out* entre paquetes *in* que salen del nodo frontera hacia el interior del dominio. Llamemos α_m (ec. 5.2) a este valor. Por simplicidad vamos a suponer que todos los paquetes tienen un tamaño similar, pero podríamos calcular α_m sumando el tamaño de los paquetes. Si la utilización de la red ronda el 100%, α_m y α_{ideal} deben ser iguales. La razón es sencilla. El dividendo de α_{ideal} representa el ancho de banda en exceso, y de la misma manera, la suma de todos los paquetes *out* representa el ancho de banda no contratado, es decir, el ancho de banda en exceso. Del mismo modo, el divisor de α_{ideal} es el total del ancho de banda del enlace contratado, y la suma de todos los paquetes marcados como *in* debe ser este ancho de banda garantizado. Por lo tanto, en una situación ideal α_m debe ser igual a α_{ideal}. Nótese que α_{ideal} no varía a menos que nuevos usuarios contraten ancho de banda para acceder a la red a través de este nodo frontera, o bien que un usuario se dé de baja del servicio o quiera modificar su contrato.

$$\alpha_m = \frac{\sum_{t_1}^{t_2} paquetes\ out}{\sum_{t_1}^{t_2} paquetes\ in} \tag{5.2}$$

Si ambas ecuaciones no coinciden significa que no se está consumiendo todo el ancho de banda disponible. Son varios los motivos por los que puede que no se esté empleando todo el ancho de banda. Puede ocurrir que una fuente se pare, intencionadamente o por algún fallo, y que las demás fuentes no se aprovechen de ese *nuevo* ancho de banda disponible. Así, al nodo frontera le bastaría con comparar α_m y α_{ideal} para conocer el estado actual de las fuentes, es decir, para saber si se está usando todo el ancho de banda. Por simplicidad, en este capítulo trabajamos con fuentes *long-lived* y suponemos que no se produce ningún fallo. Dejamos para trabajos posteriores el estudio y monitorización por parte del nodo frontera del estado de las fuentes.

Por otro lado, si medimos el ratio α_m en cada acondicionador de tráfico, situados junto a las fuentes pero fuera del alcance del usuario final, podríamos usar α_{ideal} con el objetivo de alcanzar un reparto proporcional del ancho de banda en exceso. Nótese que α_{ideal} es un valor único, es decir, el mismo α_{ideal} se envía a todos los acondicionadores de tráfico. A los valores α_m que obtenemos en cada acondicionador de tráfico los llamaremos α_m^i, donde el superíndice *i* indica el número de fuente. Cada acondicionador de tráfico debe comparar el valor de α_m^i que ha obtenido con el α_{ideal}:

- Si tenemos la relación $\alpha_m^i < \alpha_{ideal}$ entonces esta fuente no está consumiendo la porción total de ancho de banda que le corresponde (el contratado más el exceso de modo proporcional).
- Si tenemos la relación $\alpha_m^i = \alpha_{ideal}$ entonces esta fuente está usando exactamente el ancho de banda que le corresponde.
- Por último, si tenemos la relación $\alpha_m^i > \alpha_{ideal}$ entonces esta fuente está consumiendo más ancho de banda del que debería. Por lo tanto, si detectamos esta situación ésta fuente deberá ser penalizada para que disminuya su *throughput*. La Fig.5.1 muestra esquemáticamente el funcionamiento de PETER.

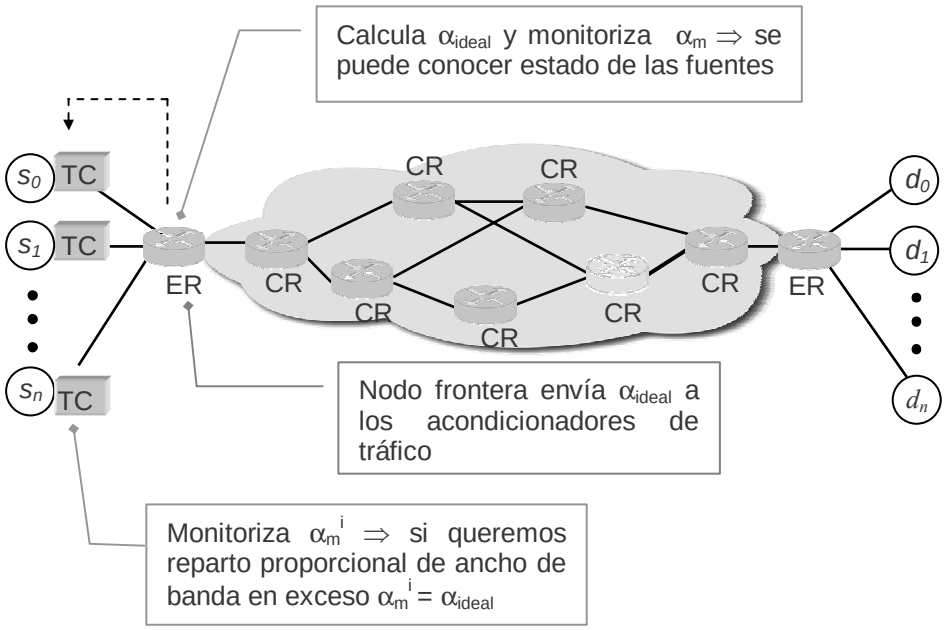


Fig. 5.1 Funcionamiento general de PETER. Nodos frontera (ER) calculan α_{ideal} y observan cada ciertos intervalos de tiempo α_m. α_{ideal} se envía a los acondicionadores de tráfico (TC) donde se aplica la función policía PETER: monitorizan α_m^i para controlar el *throughput* de la fuente. Los nodos interiores (CR) solo implementan el PHB correspondiente

Veamos un ejemplo que ilustre lo indicado con anterioridad. Supongamos que sólo hay dos fuentes de tráfico s_1 y s_2, con contratos de 1 y 10 Mbps respectivamente (véase la Fig. 5.2). La capacidad del enlace de salida del nodo frontera es de 33 Mbps, por lo que:

$$\alpha_{ideal} = \frac{capacidad\ del\ enlace - \sum_{i=1}^{n} contrato_i}{\sum_{i=1}^{n} contrato_i} = \frac{33-(1+10)}{(1+10)} = 2$$

Si queremos que se realice un reparto proporcional del ancho de banda en exceso, la fuente s_1 debería obtener 2 Mbps del ancho de banda no contratado, mientras que la fuente s_2 debería obtener 20 Mbps. Ambos valores en proporción a sus contratos:

- Ancho de banda en exceso = 33-(1+10)=22 Mbps
- Relación entre contratos 1/10, de modo que la fuente s_2 debe conseguir 10 veces más ancho de banda en exceso que s_1
- Si llamamos x a la porción de ancho de banda en exceso de la fuente s_2 tenemos que $x+x/10=22;\ x\ (1+1/10)=22;\ x= 22/\ (1+1/10)=20$ *Mbps*
- En consecuencia la fuente s_1 debe obtener 2 Mbps y la fuente s_2 20 Mbps de ancho de banda en exceso

Cada acondicionador de tráfico irá calculando α_m^i, en este caso α_m^1 y α_m^2. Si tanto uno como otro son iguales a 2, significa que el número de paquetes *out* es el doble que de paquetes *in*. En otras palabras, si el contrato de la fuente s_1 es de 1 Mbps entonces está consumiendo 2 Mbps con paquetes *out*, y lo mismo para s_2. Es decir, si en el acondicionador de tráfico el ratio α_m^i coincide con el ratio α_{ideal} entonces cada fuente está utilizando sólo el ancho de banda que le corresponde.

Por el contrario, si α_m^1 fuese igual a 3, entonces significaría que el número de paquetes *out* que está generando s_1 es el triple que los que genera como *in*. Lo que significa que en total estaría consumiendo 1 Mbps de contrato más 3 Mbps de ancho de banda en exceso. Podemos afirmar entonces que si el ratio α_m^i es mayor que α_{ideal}, entonces la fuente está usando ancho de banda que realmente pertenece a otra y deberemos penalizarla por ello para que disminuya su *throughput*.

Como último caso, si α_m^1 fuese igual a 1, entonces s_1 estaría generando el mismo número de paquetes *in* que de paquetes *out*. Traducido a Mbps, s_1 consumiría 1 Mbps de contrato más 1 Mbps de ancho de banda en exceso. En consecuencia, si el ratio α_m^i es menor que α_{ideal}, esta fuente está por debajo de lo que en realidad le corresponde.

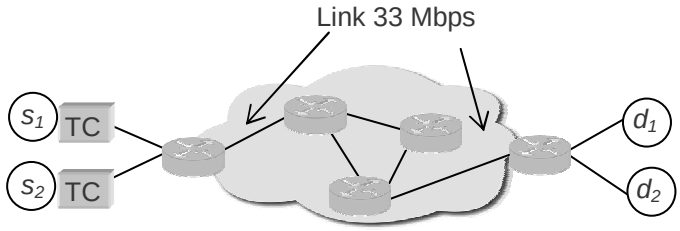


Fig. 5.2 Topología de ejemplo. Dos fuentes s_1 y s_2. Capacidad de los enlaces de salida de los nodos frontera 33 Mbps

A partir de este ejemplo concluimos que es posible conocer si las fuentes de tráfico están usando el ancho de banda que les pertenece (incluido el reparto proporcional del ancho de banda en exceso), o si nos encontramos ante una situación en la que determinadas fuentes están consumiendo más ancho de banda del que deben, dejando al resto por debajo de sus expectativas. La clave está en que cada acondicionador de tráfico compare el α_m^i que ha medido con el α_{ideal}, calculado en el nodo frontera y enviado por éste a los acondicionadores de tráfico. Recordemos que el valor de α_{ideal} no varía a menos que un usuario se dé de baja en el servicio o contrate un nuevo ancho de banda. Hemos visto como la relación $\alpha_m^i = \alpha_{ideal}$ nos indica que esta fuente está usando el ancho de banda correcto, pero la desigualdad identifica un problema de injusticia en el consumo del ancho de banda disponible. De hecho, si un acondicionador detecta que su $\alpha_m^i > \alpha_{ideal}$ deberá llevar a cabo alguna acción para que esta fuente reduzca su *throughput*. Puesto que estamos trabajando con fuentes TCP Reno, un paquete perdido reduce la velocidad de las fuentes. Así, cuando un acondicionador detecte la relación $\alpha_m^i > \alpha_{ideal}$ descartará paquetes para conseguir que disminuya la velocidad de la fuente.

Hay diferentes formas de realizar descarte de paquetes. Una de ellas sería descartar paquetes, independientemente del tipo (*in* o *out*), siempre que se detecte la relación $\alpha_m^i > \alpha_{ideal}$. El posible problema que puede arrastrar este tipo de descarte es que eliminar paquetes *in* puede afectar a la hora de asegurar los contratos. La solución por la que optamos es descartar únicamente paquetes *out* cuando la condición $\alpha_m^i > \alpha_{ideal}$ sea cierta. Nuestra propuesta operaría como muestra el pseudo-código de la Fig. 5.3.

```
α^i_m = ratio paquetes out/paquetes in
if (α_m^i ≤ α_ideal)
       no se descarta paquete
else
       if paquete in
              no se descarta paquete
       else
              descartar paquete
```

Fig. 5.3 Pseudo-código del algoritmo PETER aplicado en los acondicionadores de tráfico

Es obvio que antes de aplicar la función policía PETER, los paquetes han sido marcados a la entrada del acondicionador de tráfico. En este caso, y para continuar experimentando con las propuestas de capítulos anteriores, el marcado se realiza con el algoritmo CB. Aunque cualquier otro algoritmo de marcado sería válido, hemos visto ya como CB presenta mejores prestaciones que TSW o LB, y además aporta una simplicidad de la que no disponen sus homólogos. El pseudo-código de CB se muestra en la Fig. 2.14 del Capítulo 2.

5.3. Descripción de los escenarios de simulación

En esta sección se detallan las características de los escenarios escogidos para el análisis por simulación. La topología se muestra en la Fig. 5.4. Disponemos de *n* fuentes TCP Reno (s_1, s_2, ..., s_n) que transmiten a la velocidad del enlace, establecida en 33 Mbps. Todas las fuentes envían tráfico a sus destinos (d_1, d_2, ..., d_n) a través del nodo frontera E_1. El cuello de botella queda por tanto situado entre los nodos E_1 y E_2. α_{ideal} se calcula en el nodo E_1 con dos datos (ec. 5.1): la suma de los anchos de banda contratados por cada una de las fuentes que confluyen en E_1 (s_1 a s_n) y la capacidad del enlace de salida al siguiente nodo de la red, E_2. Para el AF PHB, los nodos utilizan RIO[1]. Los parámetros de RIO que hemos empleado en las simulaciones son [40/70/0,02] para los paquetes *in* y [10/40/0,2] para los paquetes *out*. Los parámetros de RED *weight_in* y *weight_out* han sido escogidos igual a 0,002 siguiendo las recomendaciones de [FLOY93].

Evaluaremos las prestaciones de la nueva propuesta en cinco escenarios distintos. Además, variaremos el nivel de la carga de la red desde un 20% a un 90% en todos los casos de estudio. Las características de cada escenario se detallan a continuación:

- Caso 1. Como primer caso no hay variabilidad de contratos ni de tiempos de ida y vuelta entre las fuentes que componen el agregado. Éste será el escenario más simple. El tiempo de ida y vuelta queda fijado a 50 ms. La topología consta de ocho fuentes cuyo contrato depende de la carga de red, por ejemplo para una carga del 20% el contrato será de 0,825 Mbps para todas las fuentes.
- Caso 2. En este caso evaluaremos el efecto de la diversidad de contratos entre las distintas fuentes. Los contratos dependen de nuevo de la carga de red, por ejemplo para una carga del 60% y ocho fuentes los contratos serán de 1, 1, 2, 2, 3, 3, 4 y 4 Mbps. El tiempo de ida y vuelta es el mismo para todas las conexiones e igual a 50 ms.
- Caso 3. En este caso estudiaremos el efecto de la diversidad de tiempos de ida y vuelta entre las distintas fuentes. En una topología de ocho fuentes TCP Reno este tiempo queda establecido a 10, 20, 30, 40, 50, 60, 70 y 80 ms de la fuente s_1 a la s_8 respectivamente. El contrato es el mismo para todas las conexiones y depende de la carga de red.
- Caso 4. Como situación más realista, en este caso tenemos variabilidad de contratos y de tiempos de ida y vuelta entre las fuentes. En una topología de ocho fuentes el tiempo de ida y vuelta queda establecido a 10, 20, 30, 40, 50, 60, 70 y 80 ms de la fuente s_1 a la s_8 respectivamente. Los contratos dependen de nuevo de la carga de red.
- Caso 5. En este último caso analizaremos el efecto de incrementar el número de fuentes. En concreto, la topología consta de dieciséis fuentes de tráfico. El tiempo de ida y vuelta es el mismo para todas las conexiones e igual a 50 ms. Las cuatro primera fuentes contratan un ancho de banda fijo de 1 Mbps independientemente de la carga

[1] Para el AF PHB también se han realizado simulaciones utilizando el algoritmo DQ introducido en el Capítulo 3. Los resultados se incluyen en la Sección 5.4.6.

de red. El resto de fuentes tienen el mismo contrato, pero será de un valor u otro en función de la carga. Por ejemplo, para una carga del 20% estas fuentes contratan 0,216 Mbps.

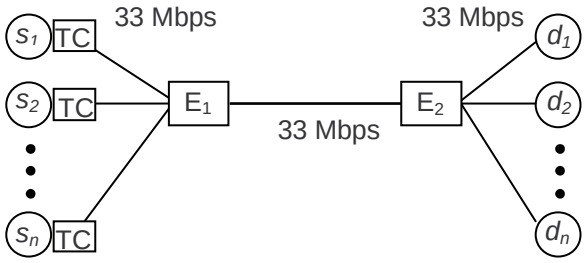


Fig. 5.4 Topología para el estudio mediante simulación

Los resultados de las simulaciones tienen un intervalo de confianza del 95% que ha sido calculado con una función de distribución normal utilizando 30 muestras, con un valor aproximado de ±0,002 para los cálculos del índice de justicia y de ±0,01 para los valores del *goodput*.

5.4. Resultados

En esta sección se discuten los resultados obtenidos. Evaluaremos las prestaciones del acondicionador de tráfico junto con la nueva función policía PETER en términos de garantías de alcanzar los contratos y justicia en el reparto del ancho de banda no contratado. Estos resultados se van a comparar con el clásico TSW, pero siguiendo para las simulaciones la guía de configuración incluida en el Capítulo 2. De manera que podríamos considerarlo un TSW mejorado. A diferencia de capítulos anteriores donde buscábamos un reparto equitativo del ancho de banda en exceso, el grado de justicia se valorará mediante el índice de justicia para distribución proporcional mostrado en las ecuaciones (1.1) y (1.2) del Capítulo 1.

5.4.1. Caso 1: igualdad en los contratos y en los tiempos de ida y vuelta

En este primer caso de estudio, la topología consta de ocho fuentes TCP Reno (s_1 a s_8) todas con el mismo ancho de banda contratado. El tiempo de ida y vuelta es de 50 ms para todas las conexiones. Con estas características, el caso 1 se presenta como el más simple de los que vamos a estudiar. La Tabla 5.1 ilustra la gran exactitud con la que nuestra propuesta consigue garantizar los contratos de los usuarios para una carga de red del 60%. La principal diferencia observable con TSW es que las tasas de paquetes *in* de éste último no son capaces de garantizar los anchos de banda contratados. En este primer caso este hecho no representaría un problema grave, puesto que se alcanzarían los contratos con el *goodput* total. Sin embargo, en una situación más compleja en la que fuera necesario descartar paquetes *out* en nodos intermedios de la red o ante un incremento de la carga de la red, sí que se convertiría en un problema.

Podemos observar en la Fig. 5.5 el *goodput* alcanzado por todas las fuentes para una carga de red del 60% y empleando PETER. Puesto que todas las fuentes contratan el mismo ancho de banda (2,5 Mbps), el reparto del ancho de banda en exceso de modo proporcional hace que todas las fuentes obtengan la misma porción de ancho de banda. El descarte de paquetes *out* en el propio acondicionador de tráfico cuando se detecta la desigualdad $\alpha^i_m > \alpha_{ideal}$, hace a las

fuentes TCP disminuir su velocidad en su justa medida. Para un abanico más amplio de valores de carga de red, desde el 20% hasta el 90%, la Fig. 5.6 representa en su eje de ordenadas el índice de justicia alcanzado con ambos métodos. En este caso, ambos obtienen valores superiores a 0,9 puntos en todo el rango, aunque manteniéndose PETER ligeramente por encima.

Tabla 5.1 *Goodputs* obtenidos en el caso 1 con ocho fuentes TCP Reno con contratos de 2,5 Mbps (carga de la red del 60%). El tiempo de ida y vuelta queda fijado a 50 ms para todas las fuentes

Fuente	Contrato (Mbps)	*Goodput* de paquetes *in* (Mbps)		*Goodput* total (Mbps)	
		PETER	TSW	PETER	TSW
1	2,5	2,48	2,15	3,84	4,19
2	2,5	2,48	2,08	3,85	4,04
3	2,5	2,49	2,11	3,87	3,91
4	2,5	2,48	2,19	3,84	4,16
5	2,5	2,49	2,13	3,86	3,94
6	2,5	2,49	2,13	3,85	3,94
7	2,5	2,49	2,16	3,88	4,17
8	2,5	2,49	2,19	3,86	4,20

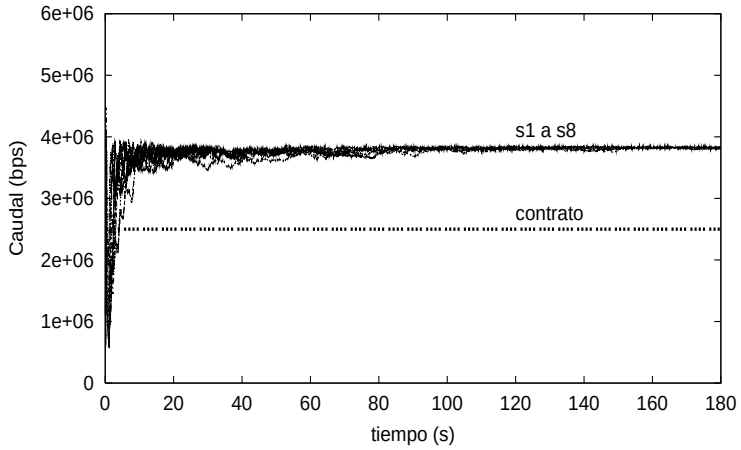


Fig. 5.5 *Goodputs* en el caso 1 con PETER y una carga de red del 60%

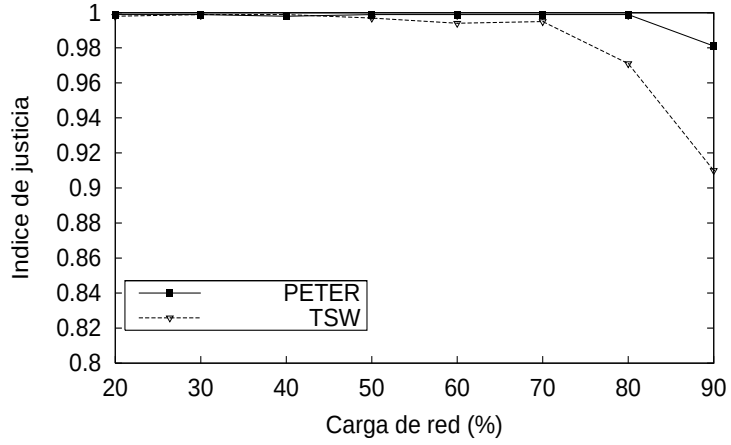


Fig. 5.6 Índice de justicia vs. carga de red en el caso 1. Comparativa entre PETER y TSW

5.4.2. Caso 2: diversidad de contratos e igualdad en los tiempos de ida y vuelta

En esta sección, la topología mantiene ocho fuentes de tráfico TCP Reno pero con diferentes contratos, que dependerán de la carga de red. Por ejemplo, para una carga del 60%, las fuentes s_1 a s_8 contratan 1, 1, 2, 2, 3, 3, 4 y 4 Mbps respectivamente. El tiempo de ida y vuelta queda establecido a 50 ms para todas las fuentes. Como caso particular, la Tabla 5.2 incluye el *goodput* obtenido con PETER y con TSW para una carga del 60%. Claramente se desprende de estos resultados que la variación de contratos no afecta a las prestaciones de PETER, pues continúa garantizando con gran exactitud los contratos de los usuarios. Con TSW, los contratos se alcanzan gracias a los paquetes *out* (es decir, considerando el *goodput* total) como ocurría en el caso 1.

Con respecto a la justicia en el reparto proporcional del ancho de banda no contratado, la Fig. 5.7 nos detalla los altos niveles de justicia que alcanzamos con PETER. Asimismo, observamos como TSW queda por debajo de estos valores. Es importante remarcar la importancia de encontrarnos con índices de justicia por encima o cercanos a 0,8. De la Fig. 5.7 se evidencia que TSW sólo alcanza este valor en un estrecho margen (carga de red entre 35% y 45%), mientras que PETER mantiene unos valores de índice de justicia por encima de 0,8 en prácticamente todo el rango (20-75%). Lo que significa que con PETER, todas las fuentes obtienen su correspondiente parte proporcional del ancho de banda en exceso. Al revés de lo que podía pensarse en un principio, el descarte de paquetes en el acondicionador de tráfico no provoca una caída de la fuente, sino que induce a las fuentes a adaptarse a las condiciones de la red a partir de la información proporcionada por α_{ideal} y α_m^i.

Tabla 5.2 *Goodputs* obtenidos en el caso 2 con ocho fuentes TCP Reno con contratos de 1-1-2-2-3-3-4 y 4 Mbps (carga de la red del 60%). El tiempo de ida y vuelta queda fijado a 50 ms para todas las fuentes

Fuente	Contrato (Mbps)	*Goodput* de paquetes *in* (Mbps)		*Goodput* total (Mbps)	
		PETER	TSW	PETER	TSW
1	1	0,97	0,80	1,25	2,79
2	1	0,96	0,79	1,22	2,80
3	2	1,98	1,64	2,91	3,56
4	2	1,97	1,67	2,88	3,52
5	3	2,99	2,65	4,65	4,43
6	3	2,99	2,63	4,76	4,35
7	4	3,98	3,85	5,21	5,50
8	4	3,99	3,79	5,25	5,30

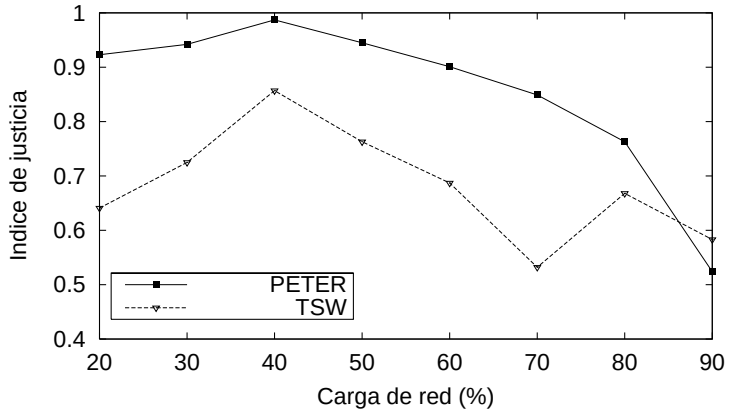


Fig. 5.7 Índice de justicia vs. carga de red en el caso 2. Comparativa entre PETER y TSW

5.4.3. Caso 3: igualdad de contratos y variación en los tiempos de ida y vuelta

Para evaluar el efecto del tiempo de ida y vuelta, presentamos un escenario con ocho fuentes TCP Reno, donde todas contratan el mismo ancho de banda pero los tiempos de ida y vuelta varían entre 10 y 80 ms (a intervalos de 10 ms). Es conocida la influencia que el RTT tiene sobre el *throughput* final [SEDD99]. En escenarios heterogéneos como éste, suele producirse una clara tendencia a favor de las fuentes con menor RTT, a menos que este efecto sea contrarrestado de algún modo. Según los resultados incluidos en la Tabla 5.3, vemos como los contratos se garantizan al 100% con PETER. De lo que podemos deducir que la diversidad en los tiempos de ida y vuelta no afecta a las prestaciones de nuestra propuesta.

Tabla 5.3 *Goodputs* obtenidos en el caso 3 con ocho fuentes TCP Reno con contratos de 2,5 Mbps (carga de la red del 60%). El tiempo de ida y vuelta queda fijado en 10-20-30-40-50-60-70 y 80 ms para las conexiones s_1 a s_8 respectivamente

Fuente	Contrato (Mbps)	*Goodput* de paquetes *in* (Mbps)		*Goodput* total (Mbps)	
		PETER	TSW	PETER	TSW
1	2,5	2,49	2,12	3,89	3,31
2	2,5	2,47	2,01	3,65	4,35
3	2,5	2,48	2,23	3,73	4,47
4	2,5	2,48	2,28	3,72	4,26
5	2,5	2,49	2,39	3,81	4,23
6	2,5	2,50	2,52	3,92	4,02
7	2,5	2,50	2,55	3,55	4,02
8	2,5	2,49	2,54	3,22	3,82

Queda manifiesto según la Fig. 5.8, que con PETER obtenemos un índice de justicia por encima de 0,8 para todo el rango de valores (20%-90%). TSW también muestra buenos valores para el índice de justicia pero sólo hasta el 70% de carga de red y siempre por debajo de PETER.

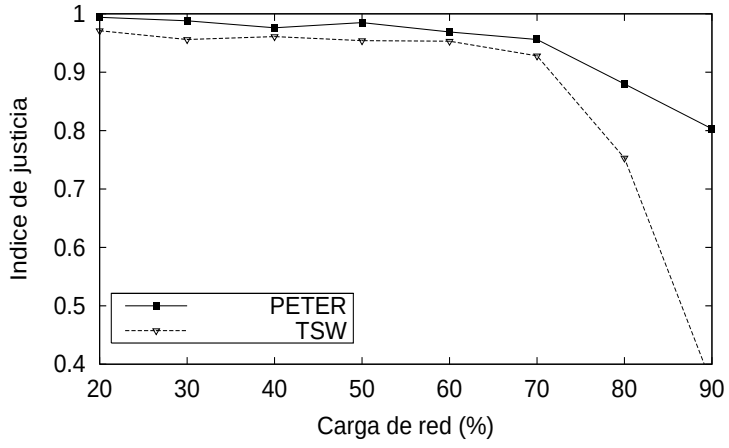


Fig. 5.8 Índice de justicia vs. carga de red en el caso 3. Comparativa entre PETER y TSW

5.4.4. Caso 4: diversidad de contratos y de tiempos de ida y vuelta

Evaluamos en este caso las prestaciones de PETER en un ambiente más realista. El escenario consta de ocho fuentes TCP Reno con diferentes contratos y variedad en los tiempos de ida y vuelta. Los RTT serán iguales a los del caso 3, variando entre 10 ms y 80 ms. Los contratos de cada fuente variarán según la carga de la red. Por ejemplo, para una carga del 60% los contratos irán de 1 a 4 Mbps. Como se desprende de la Tabla 5.4, PETER consigue garantizar de nuevo los contratos de un modo riguroso a todas las fuentes. TSW, por su parte, es capaz de alcanzar los contratos pero sólo con la ayuda de los paquetes *out* como ocurría en los casos anteriores. A pesar de la heterogeneidad presente, la Fig 5.9 muestra la clara superioridad de PETER para ofrecernos un reparto justo del ancho de banda en exceso. En contra, las prestaciones de TSW empeoran conforme aumenta la diversidad.

Tabla 5.4 *Goodputs* obtenidos en el caso 4 con ocho fuentes TCP Reno con contratos de 4-4-3-3-2-2-1 y 1 Mbps (carga de red del 60%). El tiempo de ida y vuelta se establece a 10-20-30-40-50-60-70 y 80 ms para las conexiones s_1 a s_8 respectivamente

Fuente	Contrato (Mbps)	*Goodput* de paquetes *in* (Mbps)		*Goodput* total (Mbps)	
		PETER	TSW	PETER	TSW
1	4º	3,99	3,70	4,58	4,19
2	4	3,99	3,75	5,82	5,77
3	3	2,99	2,77	4,68	5,01
4	3	2,99	2,95	4,25	4,92
5	2	1,97	1,84	2,96	3,85
6	2	1,98	1,86	3,04	3,57
7	1	0,98	0,86	1,44	2,77
8	1	0,98	0,91	1,42	2,57

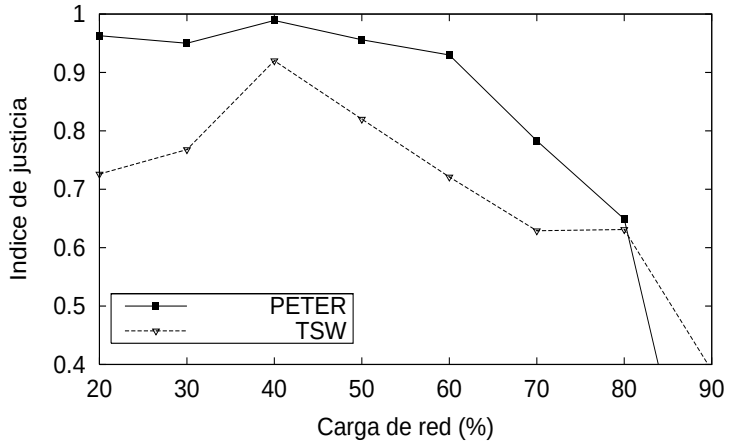


Fig. 5.9 Índice de justicia vs. carga de red en el caso 4. Comparativa entre PETER y TSW

5.4.5. Caso 5: incremento en el número de fuentes

Para finalizar el estudio de prestaciones, analizaremos el efecto de incrementar el número de fuentes. Simularemos un escenario con dieciséis fuentes TCP Reno, en el que las cuatro primeras fuentes (s_1 a s_4) tendrán en todos los casos un contrato de 1 Mbps. Las doce fuentes

restantes (s_5 a s_{16}) dispondrán del mismo contrato para completar una carga de red entre el 20% y el 90%. Por ejemplo, para una carga del 60% las fuentes s_5 a s_{16} contratan 1,32 Mbps cada una. Para esta carga de red la Tabla 5.5 ilustra las buenas prestaciones de nuestro acondicionador de tráfico a la hora de garantizar los contratos.

La Fig. 5.10 representa el índice de justicia frente a la carga de red para los dos esquemas bajo estudio, PETER y TSW. Este escenario beneficia a ambos esquemas, puesto que apenas hay variedad en los contratos y el RTT es el mismo para todas las fuentes. Aún así, se aprecia la superioridad de PETER frente a TSW para todas las cargas de red. Queda por tanto patente la robustez de PETER, ya que el incremento del número de fuentes no supone una degradación en las prestaciones finales.

Tabla 5.5 *Goodputs* obtenidos en el caso 5 con dieciséis fuentes TCP Reno con contratos de 1 Mbps (s_1 a s_4) y 1,32 Mbps (s_5 a s_{16}) para una carga de red del 60%. El tiempo de ida y vuelta se establece a 50 ms para todas las conexiones

Fuente	Contrato (Mbps)	*Goodput* de paquetes *in* (Mbps)		*Goodput* total (Mbps)	
		PETER	TSW	PETER	TSW
1	1,00	0,93	0,77	1,18	1,85
2	1,00	0,92	0,77	1,19	1,95
3	1,00	0,93	0,78	1,25	1,91
4	1,00	0,92	0,81	1,16	1,89
5	1,32	1,26	1,05	1,79	2,14
6	1,32	1,27	1,01	1,85	1,99
7	1,32	1,28	1,02	1,83	2,06
8	1,32	1,27	1,01	1,81	2,07
9	1,32	1,26	1,03	1,75	2,05
10	1,32	1,26	1,02	1,81	2,07
11	1,32	1,28	1,02	1,82	2,09
12	1,32	1,26	1,03	1,82	2,08
13	1,32	1,26	1,00	1,79	2,05
14	1,32	1,26	1,03	1,82	2,05
15	1,32	1,28	1,03	1,87	2,13
16	1,32	1,27	1,03	1,80	2,02

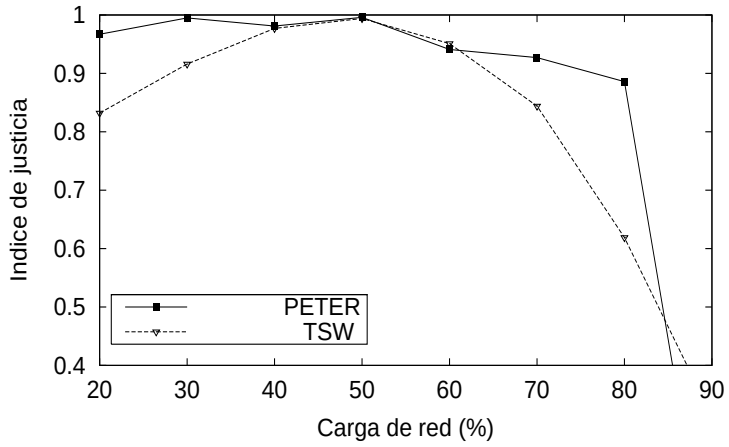


Fig. 5.10 Índice de justicia vs. carga de red en el caso 5. Comparativa entre PETER y TSW

5.4.6. Diferencias entre PETER-RIO y PETER-DQ

En este apartado mostraremos los resultados obtenidos mediante simulación al emplear la función policía PETER con el esquema de gestión de doble cola (DQ) introducido en el Capítulo 3, y haremos hincapié en las diferencias respecto al sistema PETER-RIO. Recordemos que un nodo que implementa DQ básicamente crea dos colas virtuales, una para servir paquetes *in* y otra para servir paquetes *out* (véase Fig. 5.11), con el fin de evitar interferencias entre ambos tipos de paquetes. Las dos colas son FIFO y por simplicidad, se sirven mediante una política *round-robin*. El servidor de las colas sirve la cola de paquetes *in* con una probabilidad igual a la carga total de la red ρ (ec. 3.1 del Capítulo 3), mientras que la cola de paquetes *out* se sirve con probabilidad 1-ρ. El máximo número de paquetes que se pueden almacenar está limitado por el valor HBO_{in} para la cola de paquetes *in* y HBO_{out} para la cola de paquetes *out*. En las simulaciones estos valores quedan establecidos a 20 y 10 paquetes respectivamente.

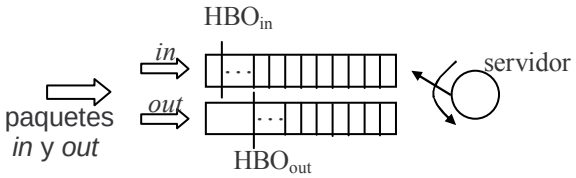


Fig. 5.11 Un nodo DQ

5.4.6.1. Caso 1: igualdad de contratos y retardos

Comenzaremos con el caso 1, donde todas las fuentes contratan el mismo ancho de banda y el retardo para todas ellas es el mismo (50 ms). Bajo estas circunstancias, los contratos se garantizan en su totalidad si utilizamos la función policía PETER. La Tabla 5.6 refleja este resultado para una carga de red del 60%. Al igual que en el caso 1 previo, TSW no es capaz de asegurar los contratos con el *goodput* de paquetes *in*. Lo que puede convertirse en un problema en topologías más complejas. En cuanto al reparto del ancho de banda en exceso, se aprecia en la Fig. 5.12 cómo el índice de justicia permanece por encima de 0,95 para todo el abanico de cargas de red. No existen por tanto diferencias apreciables entre las prestaciones de PETER-DQ y PETER-RIO en este caso.

Tabla 5.6 *Goodputs* obtenidos en el caso 1 utilizando DQ. Las ocho fuentes TCP Reno tienen contratos de 2,5 Mbps (carga de la red del 60%). El tiempo de ida y vuelta queda fijado a 50 ms para todas las fuentes

Fuente	Contrato (Mbps)	*Goodput* de paquetes *in* (Mbps)		*Goodput* total (Mbps)	
		PETER	TSW	PETER	TSW
1	2,5	2,49	2,19	3,95	3,96
2	2,5	2,45	2,25	3,92	4,18
3	2,5	2,48	2,22	3,94	4,03
4	2,5	2,38	2,18	3,89	3,96
5	2,5	2,46	2,28	3,85	4,10
6	2,5	2,42	2,17	3,85	3,91
7	2,5	2,42	2,20	3,90	4,17
8	2,5	2,40	2,20	3,91	3,97

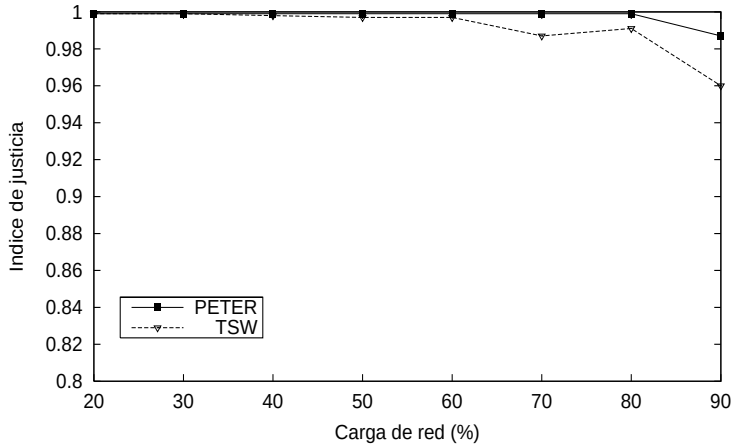


Fig. 5.12 Índice de justicia vs. carga de red en el caso 1 utilizando DQ. Comparativa entre PETER y TSW

5.4.6.2. Caso 2: variación de contratos e igualdad de retardos

Al igual que hicimos anteriormente, la topología del caso 2 consta de ocho fuentes TCP Reno con diferentes contratos. Por ejemplo, para una carga de red del 60% lo contratos quedarán fijados a 1, 1, 2, 2, 3, 3, 4 y 4 Mbps para las fuentes s_1 a s_8 respectivamente. El tiempo de ida y vuelta es el mismo para todas las fuentes, 50 ms. Como muestra la Tabla 5.7 (carga de red del 60%), PETER garantiza de modo riguroso los contratos aún cuando éstos son variables entre las distintas fuentes. De nuevo, las prestaciones de TSW están por debajo de las de PETER al no poder garantizar los contratos mediante el *goodput* de paquetes *in*. Según se observa en la Fig. 5.13, PETER presenta unos índices de justicia muy por encima de los obtenidos para TSW. Al igual que en el caso anterior, apenas existen diferencias entre el comportamiento de PETER con los esquemas RIO y DQ.

Tabla 5.7 *Goodputs* obtenidos en el caso 1 utilizando DQ. Las ocho fuentes TCP Reno tienen contratos de 1-1-2-2-3-3-4 y 4 Mbps (carga de la red del 60%). El tiempo de ida y vuelta queda fijado a 50 ms para todas las fuentes

Fuente	Contrato (Mbps)	*Goodput* de paquetes *in* (Mbps)		*Goodput* total (Mbps)	
		PETER	TSW	PETER	TSW
1	1	0,98	0,84	1,21	3,08
2	1	0,98	0,85	1,22	3,02
3	2	1,97	1,72	2,85	3,50
4	2	1,97	1,69	2,86	3,53
5	3	2,99	2,69	4,54	4,13
6	3	2,99	2,73	4,66	4,24
7	4	3,98	4,02	5,07	5,40
8	4	3,99	4,01	5,01	5,49

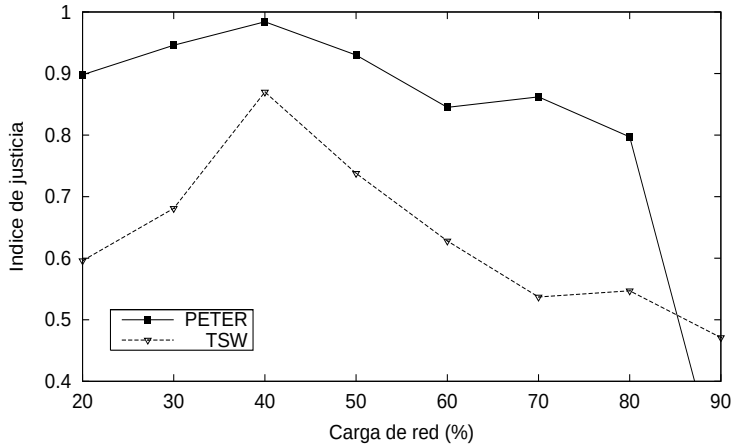


Fig. 5.13 Índice de justicia vs. carga de red en el caso 2 utilizando DQ. Comparativa entre PETER y TSW

5.4.6.3. Caso 3: igualdad de contratos y variación de retardos

En este caso estimaremos el efecto de las variaciones en el retardo sobre las prestaciones de PETER y TSW con el esquema de gestión de doble cola DQ. Trabajaremos en un escenario con ocho fuentes TCP Reno con contratos iguales y retardos que van de 10 a 80 ms para las fuentes s_1 a s_8. Los resultados que se incluyen en la Tabla 5.8 muestran claramente que PETER garantiza el contrato de todas las fuentes. Sin embargo, con TSW nos encontramos con el mismo problema que en casos anteriores. Así, podemos concluir que la variabilidad del retardo no influye en las prestaciones de PETER aún cuando utilizamos un esquema de gestión de cola diferente como es DQ.

En la siguiente figura (Fig. 5.14) podemos observar como PETER presenta un índice de justicia por encima de 0,9 para una carga de red entre el 20% y el 76%. A diferencia del caso 3 anterior en el que empleábamos RIO en vez de DQ, TSW también muestra buenos valores del índice de justicia en el rango 20%-70%. Desde nuestro punto de vista, el buen comportamiento de TSW en este caso se debe a la interacción con el esquema DQ.

Tabla 5.8 *Goodputs* obtenidos en el caso 3 utilizando DQ. Las ocho fuentes TCP Reno tienen contratos de 2,5 Mbps (carga de la red del 60%). El tiempo de ida y vuelta queda fijado a 10-20-30-40-50-60-70 y 80 ms para las fuentes s_1 a s_8

Fuente	Contrato (Mbps)	*Goodput* de paquetes *in* (Mbps)		*Goodput* total (Mbps)	
		PETER	TSW	PETER	TSW
1	2,5	2,49	1,85	3,93	3,58
2	2,5	2,48	1,85	3,57	4,26
3	2,5	2,48	2,00	3,68	4,25
4	2,5	2,49	2,18	3,76	4,21
5	2,5	2,50	2,25	3,85	4,00
6	2,5	2,49	2,40	3,76	4,27
7	2,5	2,49	2,47	3,47	4,15
8	2,5	2,49	2,44	3,15	3,55

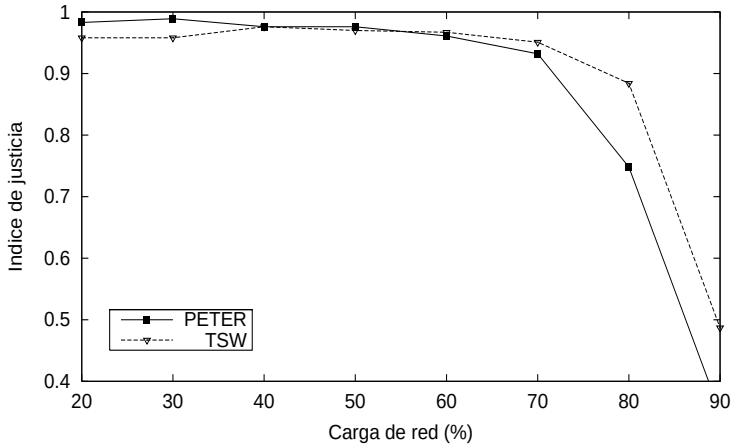


Fig. 5.14 Índice de justicia vs. carga de red en el caso 3 utilizando DQ. Comparativa entre PETER y TSW

5.4.6.4. Caso 4: variedad de contratos y de retardos

Al igual que hicimos previamente, vamos a estudiar las prestaciones de PETER y TSW con DQ en un ambiente más realista, en el que tanto los contratos como los retardos son distintos para cada fuente. El escenario consta de ocho fuentes TCP Reno cuyos retardos van de 10 a 80 ms. Los contratos variarán en función de la carga de red con la que estemos trabajando. Por ejemplo, para una carga de red del 60% los contratos irán de 1 a 4 Mbps. Según los resultados de la Tabla 5.9, PETER continúa garantizando los contratos de los usuarios. Aunque no ocurre lo mismo con TSW, que siguiendo la misma tónica, no alcanza los contratos con el *goodput* de paquetes *in*. Además, la Fig. 5.15 demuestra la clara superioridad de PETER frente a TSW a la hora de ofrecer un reparto del ancho de banda proporcional a los contratos en escenarios más heterogéneos.

Tabla 5.9 *Goodputs* obtenidos en el caso 4 utilizando DQ. Las ocho fuentes TCP Reno tienen contratos de 4-4-3-3-2-2-1 y 1 Mbps (carga de la red del 60%). El tiempo de ida y vuelta queda fijado a 10-20-30-40-50-60-70 y 80 ms para las fuentes s_1 a s_8

Fuente	Contrato (Mbps)	*Goodput* de paquetes *in* (Mbps)		*Goodput* total (Mbps)	
		PETER	TSW	PETER	TSW
1	4	3,99	2,91	4,87	4,43
2	4	3,99	3,17	5,80	5,68
3	3	2,99	2,42	4,70	4,49
4	3	3,00	2,58	4,27	4,40
5	2	1,98	1,72	3,00	3,84
6	2	1,98	1,82	3,10	3,83
7	1	0,99	0,88	1,40	2,98
8	1	0,98	0,90	1,33	2,47

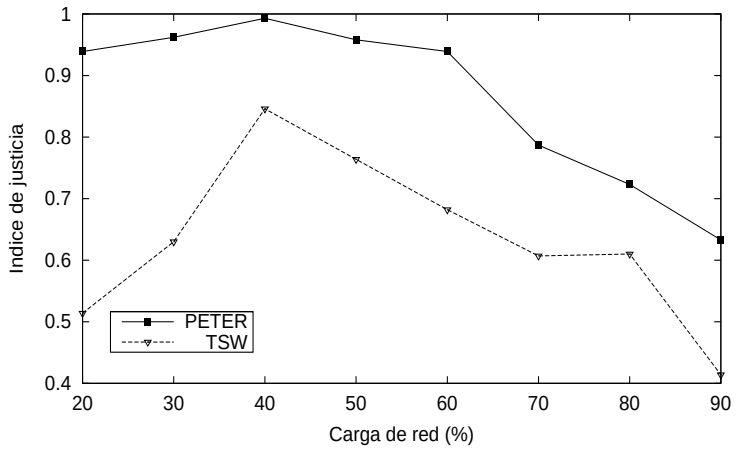


Fig. 5.15 Índice de justicia vs. carga de red en el caso 4 utilizando DQ. Comparativa entre PETER y TSW

5.4.6.5. Caso 5: incremento en el número de fuentes

Para finalizar con el estudio de PETER en combinación con el esquema DQ, incrementamos el número de fuentes para analizar el efecto que este hecho pueda tener sobre las prestaciones finales. El escenario escogido consta de dieciséis fuentes TCP Reno, en el que las cuatro primeras fuentes (s_1 a s_4) tienen un contrato fijo a 1 Mbps. El resto de fuentes (s_5 a s_{16}) contratan el mismo ancho de banda para abarcar distintas cargas de red entre el 20 y el 90%. Por ejemplo, para una carga de red del 60% las fuentes s_5 a s_{16} contratarían 1,32 Mbps cada una. La Tabla 5.10 ilustra el buen funcionamiento de PETER en este escenario. Queda así de nuevo demostrado que un incremento en el número de fuentes no supone una degradación de prestaciones para nuestra función policía PETER.

Tabla 5.10 *Goodputs* obtenidos en el caso 5 utilizando DQ. Las dieciséis fuentes TCP Reno tienen contratos de 1 Mbps (s_1 a s_4) y 1,32 Mbps (s_5 a s_{16}) (carga de la red del 60%). El tiempo de ida y vuelta queda fijado a 50 ms

Fuente	Contrato (Mbps)	*Goodput* de paquetes *in* (Mbps)		*Goodput* total (Mbps)	
		PETER	TSW	PETER	TSW
1	1,00	0,94	0,88	1,19	1,86
2	1,00	0,93	0,88	1,20	1,81
3	1,00	0,94	0,90	1,17	1,84
4	1,00	0,93	0,88	1,17	1,81
5	1,32	1,28	1,15	1,81	2,09
6	1,32	1,27	1,16	1,80	2,07
7	1,32	1,27	1,16	1,79	2,13
8	1,32	1,27	1,18	1,77	2,15
9	1,32	1,29	1,15	1,86	2,10
10	1,32	1,26	1,17	1,82	2,12
11	1,32	1,27	1,15	1,78	2,08
12	1,32	1,28	1,16	1,79	2,17
13	1,32	1,26	1,16	1,78	2,09
14	1,32	1,28	1,16	1,78	2,11
15	1,32	1,27	1,16	1,82	2,14
16	1,32	1,27	1,18	1,75	2,08

En la Fig. 5.17 tenemos representado el índice de justicia obtenido con PETER y con TSW para distintas cargas de red. Al igual que ocurría en el caso 5 que vimos con anterioridad, el hecho de no tener variedad de retardos y poca variedad de contratos provoca que ambos métodos presenten resultados aceptables. No obstante, los índices de justicia obtenidos con PETER son ligeramente superiores en todo el rango, apreciándose diferencias mínimas para cargas de nivel medio.

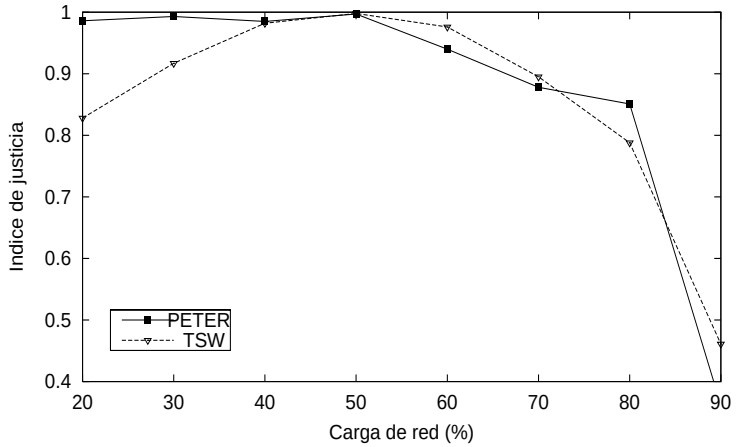


Fig. 5.16 Índice de justicia vs. carga de red en el caso 5 utilizando DQ. Comparativa entre PETER y TSW

De esta comparación entre PETER con RIO y PETER con DQ obtenemos principalmente dos conclusiones. La primera es que se pueden obtener los mismos resultados con un esquema de gestión de colas más simple como es DQ, en el que sólo hay que indicar el número máximo de paquetes que se aceptan en cada cola, puesto que dentro de cada una de ellas se sigue un régimen FIFO. En segundo lugar, podemos extraer la conclusión de que independientemente del esquema que utilicemos en el nodo frontera de la red (u otros nodos), las prestaciones de PETER se mantienen en todos los aspectos, confiriéndole la escalabilidad necesaria para esta clase de algoritmos.

5.5. Implementación

En este apartado intentaremos resolver algunas dudas que pueden aparecer sobre PETER a la hora de su implementación.

La primera cuestión a la que debemos responder es cómo puede conocer un nodo frontera la capacidad del enlace y los contratos de las fuentes a las que da entrada en la red troncal para poder calcular α_{ideal}. El ISP está a cargo de configurar todos los equipos de enrutamiento de su red (direcciones IP, protocolos de enrutamiento, listas de acceso, etc.). Esta configuración incluye también la configuración de los parámetros de QoS del enrutador. En el escenario en el que estamos trabajando, todos los nodos de la red deberían implementar el AF PHB [RFC2475][RFC2597][RFC3260]. Lo que significa que deberían utilizar un algoritmo de planificación y de gestión de cola apropiado tal como DQ, RIO o alguna de sus variantes. Los nodos conocen la capacidad de salida de cada una de sus interfaces, bien sean interfaces serie, RDSI (Red Digital de Servicios Integrados), Frame Relay, etc. De modo que la única nueva tarea que el ISP debe incluir es introducir el valor de la suma total del ancho de banda contratado que este nodo sirve. Con estos dos valores (capacidad del enlace de entrada a la red troncal y ancho de banda total contratado), el valor de α_{ideal} quedaría establecido.

Otra posible duda es cómo actuar en el caso en el que el nodo frontera tenga más de un enlace de entrada a los nodos interiores. En este estudio no hemos considerado esta situación. El motivo es que en las topologías actuales los enlaces de los usuarios se concentran en un nodo frontera (el que calcula α_{ideal}), y este nodo se conecta a través de un enlace al nodo interior que le da entrada en la red troncal. No obstante, en el caso de que el nodo frontera estuviese conectado a más de un nodo interior sería necesario un estudio más exhaustivo para utilización global de PETER.

5.6. Conclusiones

En este capítulo presentamos un nuevo acondicionador de tráfico para el servicio AF PHB, que ofrece a los usuarios unas garantías rigurosas de alcanzar los contratos establecidos y de distribuir el ancho de banda en exceso de modo justo. En este capítulo, entendemos que una distribución es justa si el uso del ancho de banda en exceso es proporcional a los contratos de cada usuario. La clave para lograr un reparto justo es la función policía a la que hemos denominado PETER. Una vez que los paquetes han sido correctamente marcados con uno de dos niveles de precedencia posibles (*in* o *out*), se aplica la función policía PETER en el acondicionador de tráfico. El nodo frontera calcula el ratio ancho de banda en exceso entre ancho de banda total contratado al que este nodo da servicio (α_{ideal}). Este valor se envía a los acondicionadores de tráfico, situados junto a las fuentes TCP pero fuera del alcance del usuario final. Los acondicionadores de tráfico medirán por su parte el ratio número de paquetes *out* entre número de paquetes *in* que han inyectado en la red (α_m^i). Como ha quedado demostrado en este capítulo, para que haya un reparto justo del ancho de banda en exceso la relación $\alpha_m^i = \alpha_{ideal}$ ha de ser verdadera, de lo contrario PETER actúa llevando la relación entre α_m^i y α_{ideal} a la igualdad.

Hemos estudiado de modo extenso las prestaciones de este nuevo acondicionador de tráfico bajo diferentes condiciones de la red: contratos varios, tiempos de ida y vuelta diferentes, variación de contratos y retardos de modo simultáneo, incremento en el número de fuentes que componen el agregado y uso de diferentes algoritmos de gestión de colas en los nodos de la red. Tras el estudio hemos observado que nuestro esquema es capaz de dar las máximas garantías a los usuarios de que alcanzarán sus contratos en todos los casos y a la vez, dispondrán del ancho de banda en exceso de modo proporcional a los mismos para cargas de red entre el 20% y el 80%, en el que la mayoría de las redes operan. Además, presenta unas prestaciones comparativamente superiores a las del clásico TSW. Concluimos por tanto que es posible contratar un servicio asegurado AF que satisface las inquietudes de los usuarios al garantizar los objetivos de este tipo de servicio en Internet.

Conclusiones

Entre las razones que deben motivar a los proveedores de servicios para emplear la arquitectura de Servicios Diferenciados destacan su gran escalabilidad y su sencilla implementación. La arquitectura DiffServ responde a las necesidades de QoS en las redes IP actuales trasladando toda la complejidad a la frontera de la red y en mejor caso, hasta las propias fuentes de tráfico. Dejando así los mecanismos a utilizar en nodos interiores tan sencillos como sea posible. De las distintas clases de servicio definidas dentro de la arquitectura DiffServ, destaca el servicio asegurado AF. Éste congrega a un mayor número de aplicaciones y en consecuencia más tráfico. Un usuario que contrate un servicio asegurado espera que se le garantice el ancho de banda que ha contratado y además, tiene la convicción de que el tráfico que genere en exceso también se transmitirá si hay recursos disponibles. Para poder ofrecer a los usuarios esta clase de servicio, habrá que poner especial atención en dos puntos clave: las técnicas empleadas para acondicionar el tráfico y los métodos de planificación y gestión de las colas en los nodos de la red.

Los objetivos de esta tesis doctoral han sido:

- En primer lugar, la evaluación comparativa de prestaciones de las técnicas existentes para la implementación del servicio asegurado: técnicas de acondicionado de tráfico y de planificación y gestión de las colas de los nodos DiffServ.
- En segundo lugar, la propuesta y evaluación de nuevos métodos de acondicionado y de gestión de colas que suplan los defectos de los métodos conocidos y sean capaces de ofrecer un servicio asegurado con todas las garantías al usuario final.

El trabajo comenzó con una revisión del estado de la técnica. Las primeras propuestas de acondicionadores de la literatura especializada presentan carencias notables. Cabe destacar el algoritmo TSW como una referencia clásica. Otros algoritmos como ETSW, el acondicionador *inteligente*, el acondicionador *justo* o el acondicionador de marcado aleatorio, intentaron aliviar los efectos que producían sobre las prestaciones finales parámetros como la heterogeneidad de contratos entre los usuarios, la variabilidad de los tiempo de ida y vuelta, etc. En algunos casos llegaron a garantizar los contratos de forma rigurosa, pero la mayoría de las veces a cambio de presentar un esquema no escalable o de requerir una señalización demasiado elevada. En el tema de la gestión de colas de los nodos, la propuesta más importante fue el esquema RIO. Con este algoritmo, paquetes con niveles de precedencia distintos se trataban de modo diferente en las colas de los nodos de la red. Aunque este esquema supuso un avance en el desarrollo del servicio asegurado, sus prestaciones eran muy mejorables. De este primer estudio concluimos que no existía ningún algoritmo de acondicionado del tráfico que pudiese garantizar los contratos de los usuarios y a la vez distribuir el ancho de banda en exceso de modo justo entre las distintas fuentes de tráfico que componen un agregado. Estas conclusiones han sido el punto de partida para el trabajo desarrollado.

En el Capítulo 2 analizamos en detalle las prestaciones de dos acondicionadores de tráfico clásicos, TSW y LB. Previo a este estudio hallamos las pautas de configuración de ambos, puesto que una de las principales dificultades que presentan es la configuración de parámetros. De los resultados del estudio se desprende que, con una configuración correcta, LB ofrece al usuario un contrato garantizado. No obstante, la distribución el ancho de banda entre las fuentes no se realiza de modo justo (en sentido equitativo) cuando hay diversidad de tiempos de ida y vuelta. TSW, por su parte, aún con una configuración óptima no garantiza de un modo preciso los contratos. Como ventaja frente a LB, TSW reparte de un modo más justo el ancho de banda

no contratado cuando hay variabilidad de tiempos de ida y vuelta. Ante la duda de escoger uno u otro, abogamos por el primero puesto que el primer objetivo a cumplir debe ser asegurar el contrato del usuario.

En este mismo capítulo introdujimos un nuevo acondicionador de tráfico denominado *Counters Based*. Su principal característica es su sencilla implementación. En comparación con los dos acondicionadores anteriores, CB garantiza estrictamente el ancho de banda contratado por los usuarios en situaciones heterogéneas. Además, es el que distribuye de modo más justo (equitativo) el ancho de banda en exceso cuando hay poca variabilidad de tiempos de ida y vuelta entre las fuentes. Sin embargo, no es capaz de realizar un reparto justo en una situación de gran variabilidad de tiempos de ida y vuelta.

En el Capítulo 3 presentamos una nueva propuesta para implementar el comportamiento por saltos del servicio asegurado. La idea de la que partimos fue eliminar cualquier tipo de interferencias entre los distintos paquetes pertenecientes a este servicio. Así nace la gestión de doble cola *Dual Queuing*. Aplicando DQ, los paquetes con mismo nivel de precedencia se sitúan en la misma cola. En nuestro caso, dado que sólo trabajamos con dos niveles tendremos dos colas. La probabilidad de servir una cola, la de más precedencia, es igual a la carga de red que sirve ese nodo. Mientras la otra cola se sirve con probabilidad complementaria. Es importante remarcar que este esquema se aleja de la definición dada inicialmente para el AF PHB, ya que según nuestro esquema los paquetes se pueden servir fuera del orden de llegada. Desde nuestro punto de vista, este hecho no es un inconveniente porque cualquier pérdida de secuencia en paquetes TCP queda resulta por el propio funcionamiento de este protocolo en el destino. El estudio de prestaciones lo realizamos utilizando CB para acondicionar el tráfico, dando lugar a la pareja CBDQ. Tras realizar simulaciones en diferentes escenarios con diversidad de contratos y tiempos de ida y vuelta, se demostró que la pareja CBDQ presenta unas prestaciones muy superiores a TSW-RIO, LB-RIO y CB-RIO. Los índices de justicia obtenidos para CBDQ en todos los escenarios son muy elevados y al mismo tiempo, los contratos de los usuarios quedaron plenamente garantizados.

El trabajo desarrollado en el Capítulo 4 se centró en buscar una mejora al acondicionador CB de manera que lograse un reparto justo, en sentido equitativo, del ancho de banda en exceso. El resultado fue el acondicionador *Counters Based Modified*. CBM penaliza a las fuentes que generan demasiado tráfico en exceso descartando paquetes de menor precedencia en el propio acondicionador. El algoritmo de descarte es del tipo RED. En concreto, cada vez que un paquete se marca como *out* se comprueba el número de paquetes *out* entre paquetes *in* consecutivos. Si este número está por debajo de un umbral mínimo el paquete *out* se transmite a la red. Si este número está por encima de un umbral máximo el paquete *out* se descarta. Por último, si este número está entre ambos umbrales el paquete *out* se descarta con probabilidad *p*. Después de detallar el método para configurar estos parámetros, pasamos al estudio de prestaciones. CBM fue analizado en topologías de uno y dos cuellos de botella. En ambos casos, se realizaron simulaciones en diferentes escenarios: variedad de contratos, variedad de tiempos de ida y vuelta, efecto de fuentes *best-effort* sobre fuentes con tráfico asegurado y por último, repercusión sobre las prestaciones si un nodo del dominio no implementa DiffServ (sólo para la topología de dos cuellos de botella), con el fin de evaluar la robustez del sistema.

El estudio de prestaciones mediante simulación reportó los siguientes resultados. En una topología de un único cuello de botella, CBM no sólo asegura los contratos de modo estricto sino que además distribuye el ancho de banda en exceso de modo justo entre las fuentes de tráfico. Lo que viene representado por unos sumamente elevados índices de justicia. Estas notables prestaciones no se ven afectadas por la variabilidad de los contratos, de los tiempos de ida y vuelta o por la competencia entre fuentes *best-effort* y fuentes con servicio asegurado por los recursos de la red. Para la topología de dos cuellos de botella, los resultados muestran que se siguen asegurando los contratos. En un caso peor, por ejemplo cuando uno de los nodos del dominio no implementa DiffServ, los contratos se garantizan al 70%. Por su parte, el reparto del ancho de banda en exceso se realiza en media de modo justo e incluso en las situaciones más adversas, presenta unas prestaciones comparativamente superiores a las de sus antecesores. En

resumen, un buen acondicionado de tráfico con el acondicionador CBM, situado junto a las fuentes de tráfico pero fuera del alcance del usuario final, demostró ser pieza clave para garantizar un servicio asegurado.

Para afrontar el problema del reparto proporcional del ancho de banda en exceso, presentamos en el Capítulo 5 el nuevo acondicionar de tráfico PETER (*Proportional Excess Traffic conditionER*). En concreto, PETER es una función policía que se aplica en el acondicionador tras marcar los paquetes IP. Para el funcionamiento de PETER es necesario definir el valor α_{ideal} como el ratio entre la capacidad del enlace de entrada al troncal del nodo frontera y la carga de red que este nodo sirve. El valor de α_{ideal} es fijo a menos que un usuario se dé de alta, se dé de baja o modifique su contrato. Este valor se calcula en el nodo frontera y, mediante señalización en el bucle de abonado, se envía a los acondicionadores de tráfico situados junto a las fuentes pero fuera del alcance de los usuarios finales. Cada acondicionador observa el ratio número de paquetes *out* entre número de paquetes *in*, denominado α_m^i. Como se demuestra en este capítulo, si α_{ideal} es igual a α_m^i, entonces cada fuente consume el ancho de banda en exceso en proporción al ancho de banda que ha contratado. Por otro lado, si $\alpha_m^i > \alpha_{ideal}$, la fuente *i* está consumiendo más ancho de banda en exceso del que le corresponde. Así, el funcionamiento de PETER en cada uno de los acondicionadores se basa en comprobar si la igualdad se cumple y si no, descartar paquetes para que la fuente disminuya el caudal.

El estudio de prestaciones mediante simulación se desarrolló en una topología de un cuello de botella. Como en capítulos anteriores, los escenarios utilizados presentan variedad de contratos, variedad de tiempos de ida y vuelta e incremento en el número de fuentes. Además, se analizaron las prestaciones de PETER para cargas de red entre el 20% y el 90%. El estudio demostró que los contratos quedaban plenamente garantizados en todos los casos y para todas las cargas de red. Además, el reparto del ancho de banda en exceso se realizaba de modo proporcional al contrato de cada fuente. Este hecho vino representado por un índice de justicia notablemente alto para cargas de red entre el 20% y el 80%. Desde un punto de vista innovador, PETER consigue de un modo práctico y más sencillo que el de otras propuestas alcanzar los objetivos de un servicio asegurado con reparto proporcional del ancho de banda excedente.

Para finalizar, el Anexo A presenta un breve estudio experimental de las soluciones disponibles en la actualidad en enrutadores comerciales para ofrecer calidad de servicio en las redes IP. El trabajo se desarrolló sobre una plataforma de prueba compuesta por cuatro enrutadores Cisco 2600 y dos PC con sistema operativo Linux y dos tarjetas de red cada uno. Se llevaron a cabo pruebas para tres implementaciones del AF PHB: una única cola FIFO con WRED, CBWFQ sin WRED y CBWFQ con WRED. Para cada uno de ellos se utilizaron tres escenarios: igualdad de contratos, variabilidad de contratos y aumento del número de fuentes. El acondicionado del tráfico (marcado) se realizó en los nodos frontera con el algoritmo *token bucket*, pues es el único que existe en equipos comerciales. El estudio proporcionó los siguientes resultados. Para todas las implementaciones se garantizan los contratos. Para los dos primeros casos (FIFO con WRED y CBWFQ sin WRED), el ancho de banda en exceso no se distribuye de forma justa (ni en sentido equitativo ni proporcional) si hay variedad de contratos. En el último caso (CBWFQ con WRED) se logra garantizar los contratos y distribuir el ancho de banda en exceso de modo justo. Sin embargo, sería imposible seguir este esquema en una red real por su demostrada falta de escalabilidad.

En consecuencia, es necesario adaptar los equipos existentes a las nuevas tendencias en el ámbito de la calidad de servicio en redes IP como las mostradas en capítulos anteriores. Si se opta por ofrecer una diferenciación de servicios con garantías a los usuarios de Internet, es fundamental el acondicionado del tráfico, a ser posible en las propias fuentes. Las propuestas presentadas en esta tesis han demostrado su validez al respecto, y probablemente su aplicación pueda extenderse a nuevos entornos como las redes 3G/4G o los nodos frontera de las redes ópticas.

Anexo A

Estudio experimental de las soluciones disponibles en enrutadores comerciales para implementar DiffServ

A.1. Introducción

La actual tendencia a utilizar DiffServ para proveer a las redes IP de QoS, nos ha motivado para estudiar DiffServ de un modo experimental. En este anexo analizaremos mediante pruebas experimentales diferentes soluciones disponibles en enrutadores comerciales para la implementación del servicio asegurado AF y de su PHB. Como parámetros de medición de QoS observaremos el caudal que obtienen las fuentes de tráfico TCP en escenarios sin variación de contratos, con variación de contratos y el efecto del número de fuentes. Para completar el estudio, consideraremos también el reparto del ancho de banda en exceso entre las distintas fuentes.

El resto del Anexo queda organizado del siguiente modo. La siguiente sección define brevemente la técnica de gestión de colas y de planificación que utilizaremos para implementar el AF PHB. La Sección A.3 describe la topología experimental y los escenarios sobre los que se realizarán las pruebas. En la Sección A.4 analizaremos los resultados obtenidos. El Anexo finaliza con las conclusiones más importantes en la Sección A.5.

A.2. Técnicas de gestión de colas

En el Capítulo 1 se introdujeron algunas técnicas básicas para la implementación de los PHB. En esta sección vamos a ver dos nuevas técnicas: WRED, muy similar al esquema RIO, y CBWFQ, una variante del conocido WFQ.

La técnica de gestión de cola RED por pesos (*Weighted RED*, WRED) [CISC04a] es una de las variantes de RED empleadas en la actualidad para ofrecer calidad de servicio a los usuarios de Internet. Además de la anticipación en la detección de la congestión, WRED permite disponer de diferentes perfiles de descarte para distintos tipos de tráfico, lo que nos faculta para ofrecer diferentes niveles de QoS. Cuando un paquete llega a una cola con WRED se producen los siguientes eventos. En primer lugar, se calcula la longitud media de la cola basándose en la longitud media previa y la longitud media actual. Después, si la estimación de la longitud media calculada está por debajo del límite inferior, el paquete se introduce en la cola. Si la longitud media calculada está entre los límites máximo y mínimo, el paquete puede descartarse o no en función de la probabilidad de descarte. Finalmente, si la longitud media calculada está por encima del límite máximo el paquete se descarta.

Por otro lado, la técnica de gestión de cola WFQ basada en clases (*Class Based WFQ*, CBWFQ) se presenta como una variante del conocido mecanismo de planificación WFQ. WFQ asocia pesos a las colas, de manera que cada cola obtenga una porción del ancho de banda

disponible. Además, acepta paquetes de distintos tamaños. CBWFQ tiene las mismas características que WFQ, con la nueva propiedad de realizar la planificación de paquetes para el acceso de éstos a las colas en función de clases de tráfico definidas previamente por el administrador.

A.3. Topología Experimental

Los medios de los que disponemos para realizar el estudio experimental son los siguientes: cuatro enrutadores Cisco de la serie 2600 y dos PCs con sistema operativo Linux. La topología se muestra en la Fig. A.1. El sistema operativo de los enrutadores de Cisco, al que se denomina Cisco IOS (*Cisco Internetworking Operating System*), incluye herramientas para la implementación de los Servicios Diferenciados en los nodos de la red. Una de estas herramientas es la interfaz de línea de comandos para QoS modular (*Modular Quality of Service Command Line Interface*, MQC). Gracias al MQC podemos configurar un marcado de paquetes basado en clases, funciones policía (hasta la fecha sólo se puede configurar la función policía *token bucket*), WRED y algunas otras facilidades de DiffServ.

Nosotros nos centraremos en el servicio AF. Los nodos frontera, en nuestro caso E_1 y E_2, llevarán a cabo las tareas de acondicionado del tráfico. Además, estos nodos también aplicarán el AF PHB adecuado. El nodo interior E_3 sólo implementará el AF PHB, mientras que E_4 sólo demultiplexará el tráfico a sus correspondientes destinos. EL mecanismo empleado para el acondicionado del tráfico es un *token bucket*. Este *token bucket* no descarta ningún paquete, sólo los marca como *in* o como *out*. En función de las características de la fuentes de tráfico, el *token bucket* tendrá diferentes parámetros de configuración. El cuello de botella se sitúa entre E_3 y E_4.

Cada PC tiene dos tarjetas de red, lo que nos ayudará a simular dos redes de área local (*Local Area Network*, LAN) diferentes. Cada tarjeta del PC_1 está conectada a un nodo frontera. El tráfico TCP se genera mediante la herramienta Netperf [NETP04]. Los destinos son las dos tarjetas de red del PC_2.

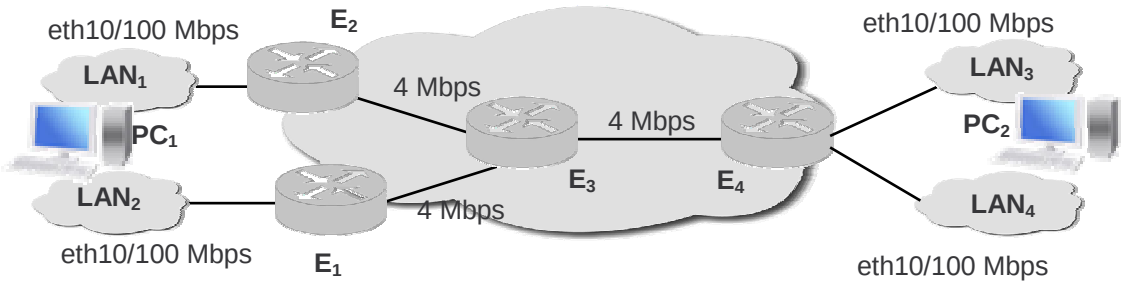


Fig. A.1. Topología experimental: 4 enrutadores Cisco 2600 y 2 PCs con sistema operativo Linux (cada PC dispone de dos tarjetas de red)

Realizaremos diferentes pruebas utilizando en cada una de ellas diferentes métodos de implementación del AF PHB. Estos métodos que vamos a emplear son los siguientes:

- Caso 1. Todo el tráfico AF (paquetes *in* y *out*) se sitúan en una misma cola FIFO con esquema de gestión WRED. Véase Fig. A.2.

o

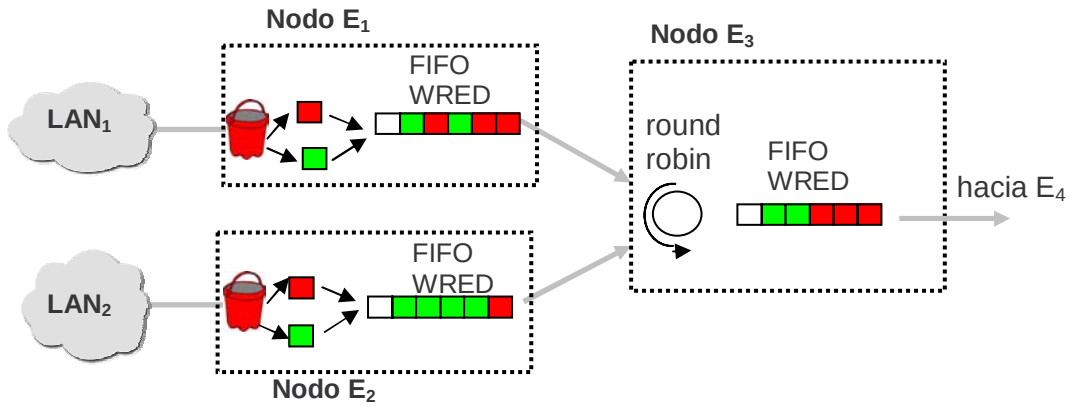


Fig. A.2 Cola FIFO con WRED para todo el tráfico AF, paquetes *in* (gris claro) y *out* (gris oscuro)

- Caso 2. Todo el tráfico AF se introduce en las colas utilizado CBWFQ, de manera que los paquetes *in* se sitúan en una cola FIFO y los paquetes *out* se sitúan en otra cola FIFO. El gestor visita la cola de paquetes *in* con una probabilidad ρ_1 que iguala la carga de red (véase ec. A.1). La cola de paquetes *out* se visita con probabilidad 1- ρ_1. Véase la Fig. A.3.

$$\rho_1 = \frac{\sum_i contrato_i}{capacidad\ del\ enlace} \qquad (A.1)$$

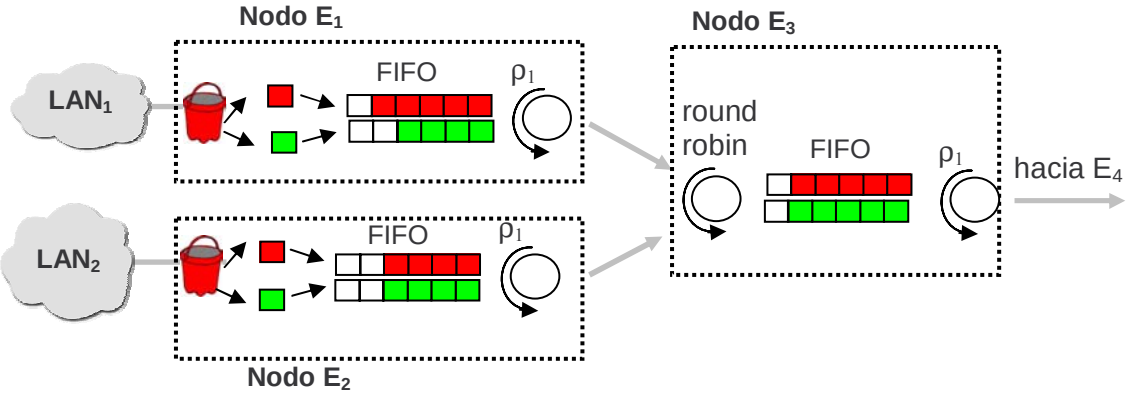


Fig. A.3 CBWFQ sin WRED para el tráfico AF (paquetes *in* y *out*). En cada nodo hay una cola para los paquetes *in* y otra para los paquetes *out*

- Caso 3. Todo el tráfico AF se introduce en las colas utilizando CBWFQ con WRED. De este modo, el tráfico proveniente de la red LAN_1 se sitúa en una cola y el tráfico proveniente de la red LAN_2 en otra cola distinta. El gestor visita la cola de la red LAN_1 con una probabilidad ρ_2 igual al ancho de banda correspondiente a esta red (contratos más ancho de banda en exceso). Véase la Fig. A.4. Por ejemplo, si las redes LAN_1 y LAN_2 han contratado 1 y 2 Mbps respectivamente, entonces el ancho de banda en exceso es de 1 Mbps. Si se realiza un reparto de este exceso de modo equitativo, la red LAN_1 debe obtener en total 1,5 Mbps y la red LAN_2 2,5 Mbps. Por lo tanto, la cola de la red LAN_1 se visita con probabilidad igual al 37,5% (véase ec. A.2) y la cola de la red LAN_2 con una probabilidad igual al 62,5%.

$$\rho_2 = \frac{contrato + porcion \quad ancho \quad de \quad banda \quad en \quad exceso}{capacidad \ del \ enlace} \tag{A.2}$$

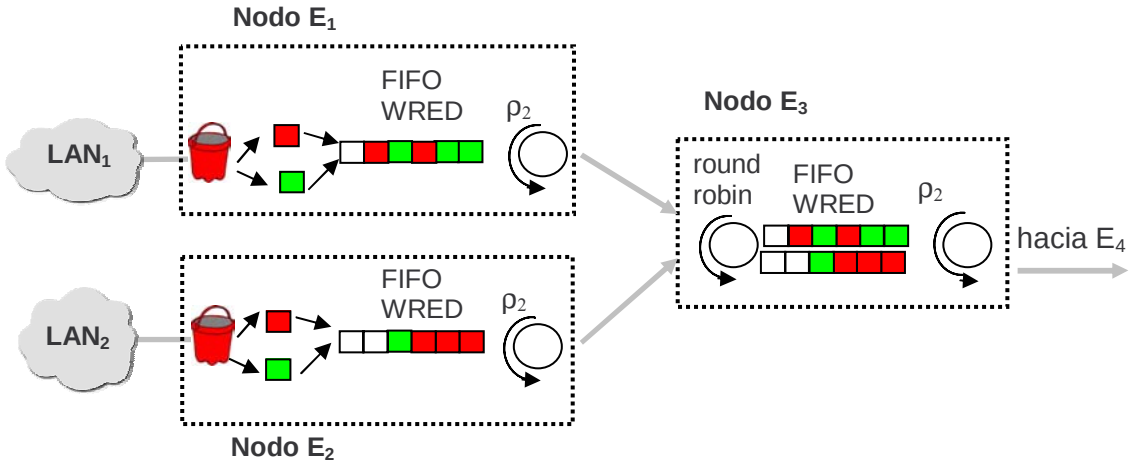


Fig. A.4 CBWFQ con WRED para todo el tráfico AF (paquetes *in* y *out*). Cada LAN tiene su propia cola. E_1 sólo trabaja con tráfico de la LAN_1 por lo que sólo implementa una cola. Lo mismo es aplicable a E_2. E_3 recibe tráfico de las dos redes LAN por lo que dispone de dos colas

Nótese que estos mecanismos de planificación y congestión se aplican a la salida de cada uno de los nodos del dominio DiffServ. A la entrada, los nodos utilizan una planificación del tipo *round robin* para comprobar si hay paquetes en las distintas interfaces de las que dispone. Para cada una de las tres implementaciones anteriores, realizaremos pruebas sobre tres escenarios distintos:

- Escenario A. Cada red LAN tiene una única fuente de tráfico TCP con el mismo contrato, que iremos incrementando desde 256 Kbps (carga de red del 12.5%) hasta 2 Mbps (carga de red del 100%). Véase la Tabla A.1.

Tabla A.1 Contratos para el escenario A

Carga de red (%)	Contratos de LAN_1 y LAN_2 (Mbps)
12,5	0,250
25,0	0,500
37,5	0,750
50,0	1,000
62,5	1,250
75,0	1,500
87,5	1,750
100	2,000

- Escenario B. Cada red LAN tiene una única fuente de tráfico TCP. La red LAN_1 tiene un contrato fijo de 256 Kbps y la red LAN_2 contrata un ancho de banda variable desde 256 Kbps (carga de red del 12.5%) a 3840 Kbps (carga de red del 100%). Véase la Tabla A.2.

Tabla A.2 Contratos para el escenario B

Carga de red (%)	Contrato de LAN_1 (Mbps)	Contrato de LAN_2 (Mbps)
12,50	0,250	0,250
18,75	0,250	0,500
25,00	0,250	0,750
31,25	0,250	1,000
37,50	0,250	1,250
43,75	0,250	1,500
50,00	0,250	1,750
56,25	0,250	2,000
62,50	0,250	2,250
68,75	0,250	2,500
75,00	0,250	2,750
81,25	0,250	3,000
87,50	0,250	3,250
93,75	0,250	3,500
100	0,250	3,750

- Escenario C. La red LAN_1 tiene cuatro fuentes de tráfico TCP con un contrato total fijo de 1024 Kbps. Por su parte, la red LAN_2 tiene un número variable de fuentes, desde 2 hasta 16, y cada una de ellas tiene un contrato de 128 Kbps (véase la Tabla A.3). Este escenario sólo lo analizaremos para los casos 2 y 3 (CBWFQ con y sin WRED).

Tabla A.3 Contratos para el escenario C

Carga de red (%)	Nº de fuentes en LAN_1 (contrato de 0,250 Mbps cada una)	Contrato de LAN_1 (Mbps)	Nº de fuentes en LAN_2 (contrato de 0,125 Mbps cada una)	Contrato de LAN_2 (Mbps)
31,25	4	1,000	2	0,250
37,50	4	1,000	4	0,500
43,75	4	1,000	6	0,750
50,00	4	1,000	8	1,000
56,25	4	1,000	10	1,250
62,50	4	1,000	12	1,500
68,75	4	1,000	14	1,750
75,00	4	1,000	16	3,000

A.4. Resultados

En esta sección vamos a estudiar mediante pruebas experimentales las prestaciones de los mecanismos descritos en la sección anterior: una única cola FIFO con WRED, CBWFQ (dos colas una para cada tipo de paquete) sin WRED y CBWFQ (tantas colas como redes LAN) con WRED. Analizaremos cada uno de los mecanismos en las tres situaciones detalladas también en la sección previa. Es importante remarcar que estos sistemas son los que en la actualidad ofrece una de las firmas más importantes del sector para proporcionar QoS. Los parámetros de medición de la QoS que vamos a observar son el caudal de los usuarios finales, sin tener en cuenta los paquetes retransmitidos, y la distribución del ancho de banda en exceso entre las fuentes. Objetivos que recordemos se corresponden con el servicio asegurado AF. Cuando una red sólo tenga una fuente de tráfico utilizaremos indistintamente los términos red o fuente de tráfico.

A.4.1. Caso 1: Una única cola FIFO con WRED

En este apartado presentamos los resultados obtenidos cuando todo el tráfico AF se dispone en una única cola FIFO con control de la congestión del tipo WRED. Los parámetros que se han utilizado para WRED han sido [40/70/0,02][1] para los paquetes *in* y [10/40/0,2] para los paquetes *out*. Hemos escogido estos valores tal y como se indica en [FLOY93]. La longitud del la cola es de 200 paquetes.

La Fig. A.5 muestra el caudal de paquetes *in* alcanzado por cada una de las fuentes de tráfico TCP en el escenario A. Los contratos quedan garantizados independientemente de la carga de la red. En general, ambas fuentes se benefician por igual del ancho de banda en exceso (véase Fig. A.6). Las mayores diferencias en cuanto al reparto del ancho de banda en exceso se producen para cargas de red en el intervalo 50-62.5%. Estos buenos resultados se deben a que ambas fuentes tienen el mismo contrato y no hay por lo tanto diferencia alguna entre ambas.

Si examinamos el efecto de dejar una fuente (LAN_1) con un contrato fijo a 256 Kbps y variar el contrato de la otra (escenario B), nos encontramos con los siguientes resultados. La Fig. A.7 muestra que el contrato queda asegurado para ambas fuentes en todo el rango de cargas de red. Sin embargo, hay una carencia de justicia en la distribución del ancho de banda no contratado como ilustra la Fig. A.8. La fuente con menor contrato (LAN_1) se beneficia notablemente de un mayor ancho de banda en exceso.

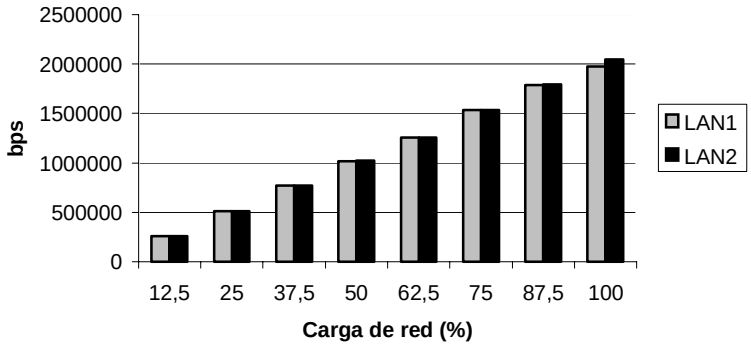


Fig. A.5 Caudal de paquetes *in* en el escenario A (ambas redes tienen el mismo contrato de 256 Kbps a 2 Mbps). El AF PHB se implementa con una única cola FIFO con WRED

[1] Se corresponde con [límite mínimo, límite máximo, probabilidad de descarte]

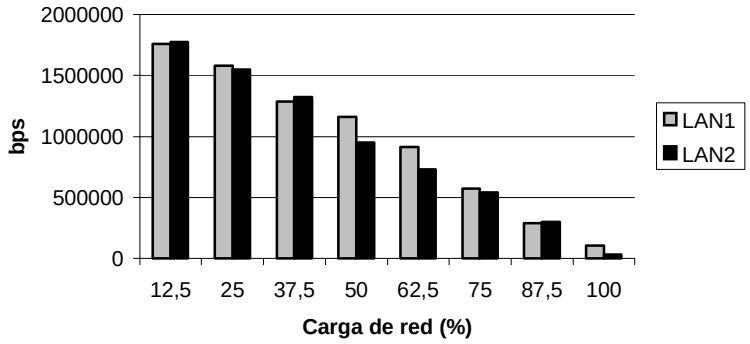


Fig. A.6 Ancho de banda en exceso en el escenario A. El AF PHB se implementa con una única cola FIFO con WRED

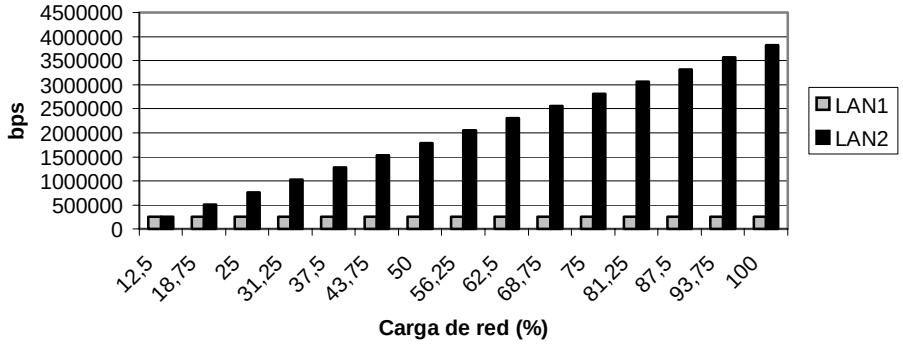


Fig. A.7 Caudal de paquetes *in* en el escenario B (las redes tienen contratos distintos). El AF PHB se implementa con una única cola FIFO con WRED

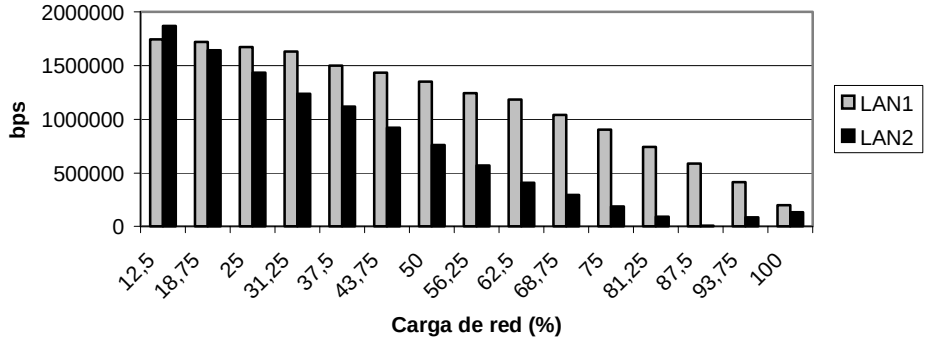


Fig. A.8 Ancho de banda en exceso en el escenario B. El AF PHB se implementa con una única cola FIFO con WRED

A.4.2 Caso 2: CBWFQ sin RED

En esta sección presentamos los resultados obtenidos cuando el tráfico AF se almacena utilizando el esquema CBWFQ sin WRED. En función del tipo de paquete (*in* o *out*), éste se sitúa en una cola FIFO distinta. Este tipo de almacenamiento se corresponde con la implementación del AF PHB introducida en el Capítulo 3, cuyo principal objetivo es eliminar interferencias entre ambos tipos de paquetes.

La Fig. A.9 representa el caudal de paquetes *in* que consigue cada fuente en el caso de que ambas contraten el mismo ancho de banda (escenario A), desde 256 Kbps a 2 Mbps. Vemos como los contratos están totalmente garantizados. Respecto al ancho de banda en exceso, la distribución se realiza de un modo justo (equitativo), obteniendo cada fuente aproximadamente la mitad del ancho de banda no contratado (véase Fig. A. 10). Estos resultados coinciden con los alcanzados con el esquema anterior, FIFO con WRED. La razón de este buen comportamiento, al igual que ocurría antes, es la homogeneidad presente en este escenario. En un escenario más heterogéneo con variedad de contratos (escenario B), éstos se siguen garantizando (Fig. A.11). Sin embargo el reparto del ancho de banda en exceso es imparcial para cargas de red superiores al 30% (véase Fig. A.12). A partir de ese valor la fuente con menor contrato obtiene más ancho de banda en exceso.

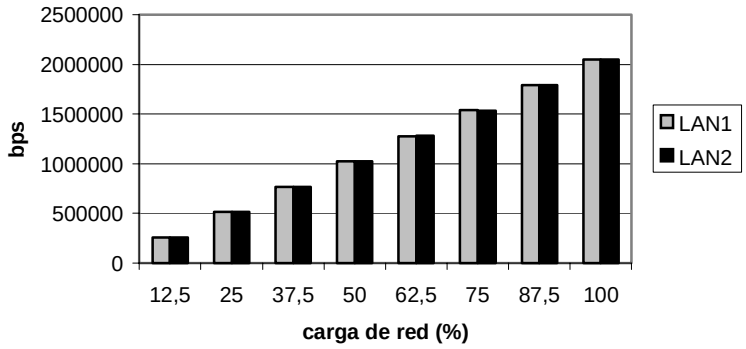


Fig. A.9 Caudal de paquetes *in* en el escenario A (ambas redes tienen el mismo contrato de 256 Kbps a 2 Mbps). El AF PHB se implementa con dos colas FIFO (una para paquetes *in* y otra para paquetes *out*) sin WRED

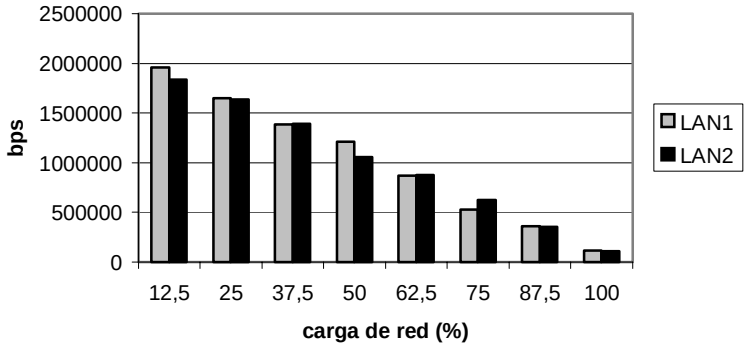


Fig. A.10 Ancho de banda en exceso en el escenario A. El AF PHB se implementa con dos colas FIFO (una para paquetes *in* y otra para paquetes *out*) sin WRED

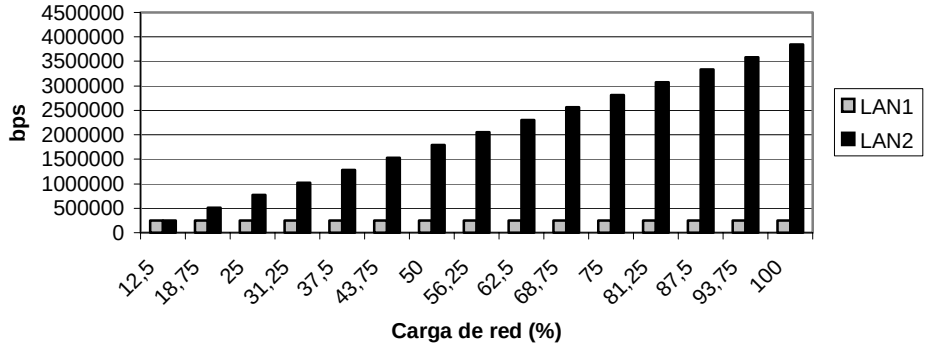


Fig. A.11 Caudal de paquetes *in* en el escenario B (las redes tienen contratos distintos). El AF PHB se implementa con dos colas FIFO (una para paquetes *in* y otra para paquetes *out*) sin WRED.

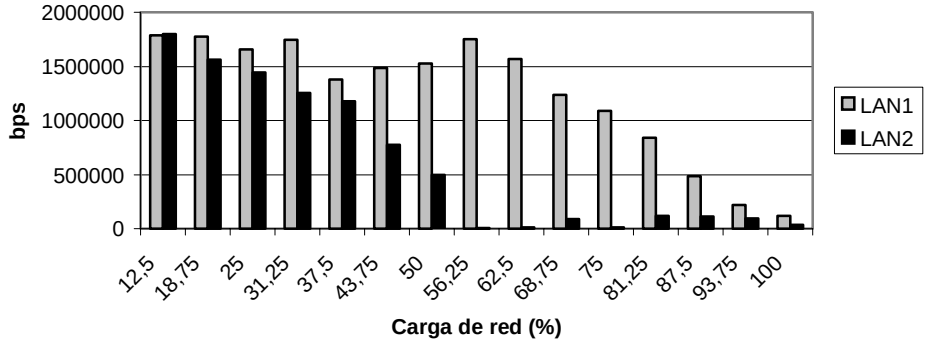


Fig. A.12 Ancho de banda en exceso en el escenario B. El AF PHB se implementa con dos colas FIFO (una para paquetes *in* y otra para paquetes *out*) sin WRED

Para este esquema también estudiamos el efecto que produce sobre las prestaciones finales un incremento del número de fuentes dentro de cada red LAN (escenario C). Recordemos que en este escenario la red LAN_1 tiene siempre cuatro fuentes de tráfico TCP y dispone de un contrato fijo de 1024 Kbps. Mientras, la red LAN_2 irá incrementando el número de fuentes TCP de 2 a 16, y cada fuente contratará 128 Kbps para abarcar una carga de red desde el 31% hasta el 75%. Tras las pruebas detectamos que independientemente del número de fuentes, cada LAN alcanza su contrato con el caudal de paquetes *in*. Dentro de cada LAN el caudal total se distribuye por igual entre las distintas fuentes. Por ejemplo, observe la Fig. A.13 en la que representamos el caudal de las fuentes de las redes LAN_1 y LAN_2 en el caso en el que la primera tiene un contrato de 1024 Kbps (cuatro fuentes) y la segunda un contrato de 1536 Kbps (12 fuentes), correspondiente a una carga de red del 62,5%. Respecto al reparto del ancho de banda en exceso, en la Fig. A.14 se aprecia claramente que prevalece la influencia del contrato menor sobre el número de fuentes, es decir, la red LAN cuyas fuentes tienen menores contratos obtiene más ancho de banda en exceso. Por ejemplo, cuando la red LAN_1 tiene cuatro fuentes con contratos de 256 Kbps y la red LAN_2 tiene sólo dos fuentes con contratos de 128 Kbps (carga de red del 31,25%), ésta última obtiene más ancho de banda en exceso aunque tenga menos fuentes.

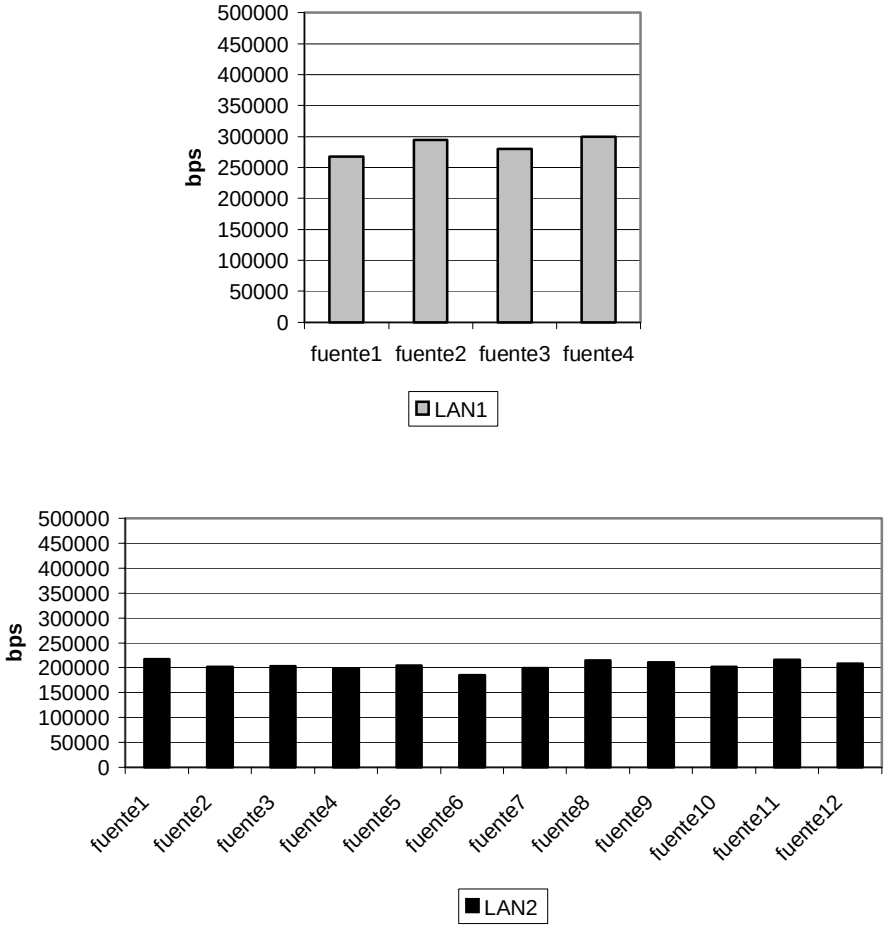


Fig. A.13 Caudal de paquetes *in* para una carga de red del 62,5% en el escenario C. La red LAN_1 tiene un contrato total de 1024 Kbps y cuatro fuentes de tráfico TCP. La red LAN_2 tiene doce fuentes de tráfico TCP, cada una de ellas con contrato de 128 Kbps. El AF PHB se implementa con dos colas FIFO (una para paquetes *in* y otra para paquetes *out*) sin WRED

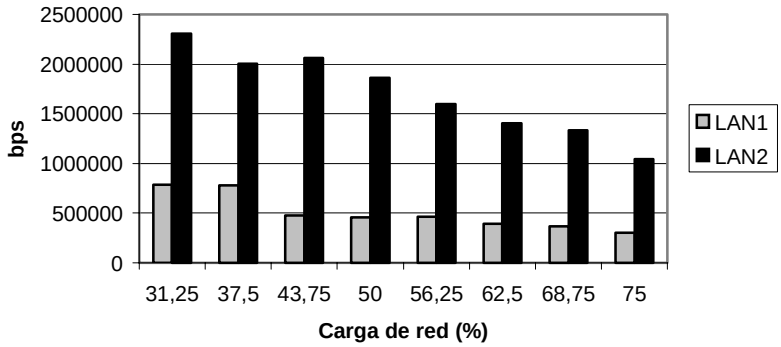


Fig. A.14 Ancho de banda en exceso que obtiene cada red LAN en el escenario C. El AF PHB se implementa con dos colas FIFO (una para paquetes *in* y otra para paquetes *out*) sin WRED

A.4.3 CBWFQ con WRED

En esta sección analizaremos los resultados obtenidos cuando empleamos CBWFQ con WRED para gestionar el tráfico AF. El tráfico proveniente de cada red LAN se almacenará en colas distintas, donde cada cola implementará WRED como mecanismo de prevención de la congestión con los mismos parámetros que empleamos en la sección A.3.1. Las colas se sirven con la probabilidad definida en la ecuación (A.2).

En este caso, se puede apreciar que se aseguran de modo estricto los contratos y además el ancho de banda en exceso se distribuye de modo equitativo entre las dos redes (véanse Fig. A.15 y A.16). Con las otras técnicas (FIFO con WRED y CBWFQ sin WRED) los resultados fueron similares cuando ambas redes contrataban el mismo ancho de banda. Sin embargo, CBWFQ con WRED es la única configuración que no sólo garantiza los contratos sino que reparte de modo justo el ancho de banda en exceso cuando los contratos de las redes son diferentes y para una carga de red hasta del 80% aproximadamente (véanse Fig. A.17 y A.18). Además, incrementar el número de fuentes no representa ningún inconveniente para esta implementación. Los contratos siguen estando garantizados y, como muestra la Fig. A.19, el ancho de banda se reparte de modo justo (equitativo) entre las dos redes.

Como contrapartida a esta implementación se pueden argumentar los siguientes inconvenientes:

- El número de colas necesarias para el tráfico AF es igual al número de redes LAN que confluyen en un nodo. Por tanto es muy probable que si el número de redes es muy elevado el nodo no sea capaz de implementar tantas colas.
- Para configurar cada nodo del dominio DiffServ, tanto los nodos frontera como los interiores, es necesario conocer el contrato de todas las redes y las capacidades de todos los enlaces. En concreto, es obligatorio para poder calcular la probabilidad con la que servir cada cola. Obviamente, se trata de un sistema muy poco escalable.

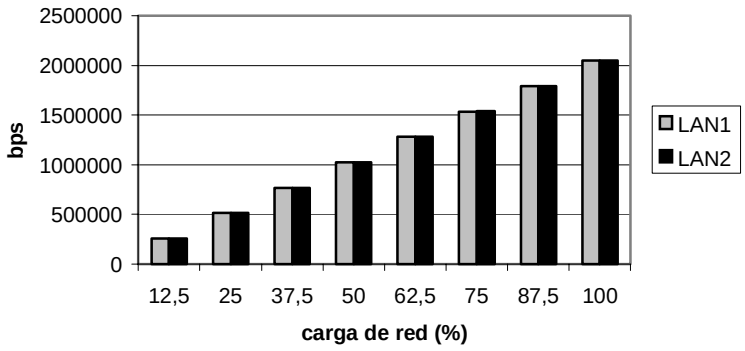


Fig. A.15 Caudal de paquetes *in* en el escenario A (ambas redes tienen el mismo contrato de 256 Kbps a 2 Mbps). El AF PHB se implementa con CBWFQ con WRED. El tráfico de cada LAN va a una cola distinta

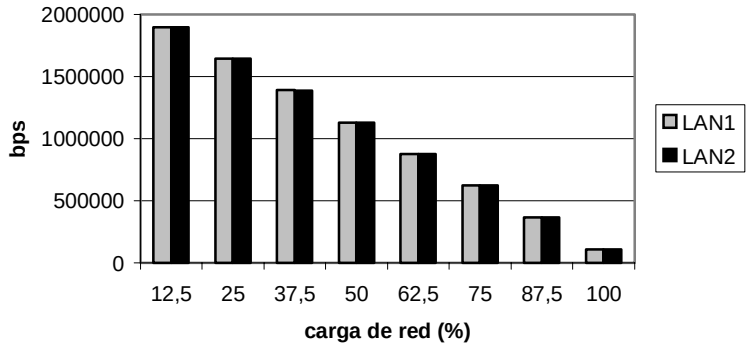


Fig. A.16 Ancho de banda en exceso en el escenario A. El AF PHB se implementa con CBWFQ con WRED. El tráfico de cada LAN va a una cola distinta

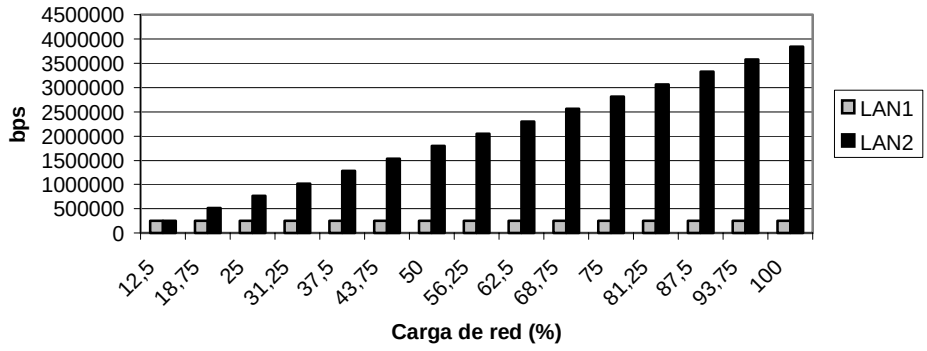


Fig. A.17 Caudal de paquetes *in* en el escenario B (las redes tienen contratos diferentes). El AF PHB se implementa con CBWFQ con WRED. El tráfico de cada LAN va a una cola distinta.

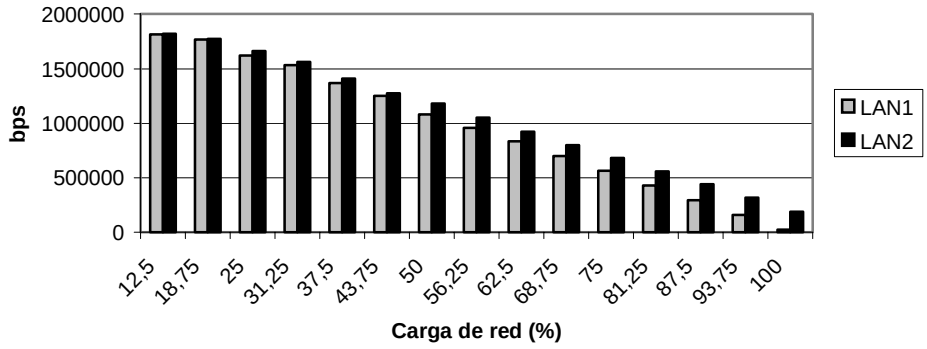


Fig. A.18 Ancho de banda en exceso en el escenario B. El AF PHB se implementa con CBWFQ con WRED. El tráfico de cada LAN va a una cola distinta

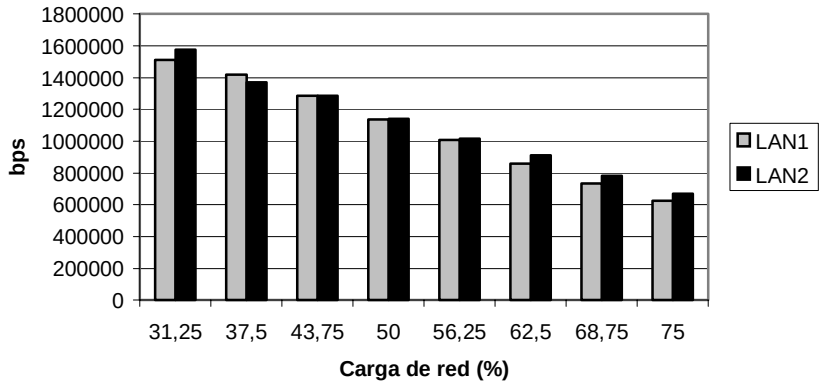


Fig. A.19 Ancho de banda en exceso en el escenario C. La red LAN_1 tiene un contrato total de 1024 Kbps y cuatro fuentes de tráfico TCP. La red LAN_2 incrementa el número de fuentes de 2 a 16, cada una de ellas con contrato de 128 Kbps. El AF PHB se implementa con CBWFQ con WRED. El tráfico de cada LAN va a una cola distinta.

A.5. Conclusiones

Hemos realizado un estudio experimental de las posibles configuraciones existentes en enrutadores comerciales actuales capaces de ofrecer QoS basada en los Servicios Diferenciados. Hemos empleado enrutadores de la firma Cisco System, en concreto de la serie 2600, ampliamente utilizados en las redes IP para crear la topología experimental. Como orígenes y destinos del tráfico utilizamos PCs con sistema operativo Linux, que usan NetPerf para generar tráfico TCP. El acondicionado del tráfico en los nodos frontera se realiza mediante el algoritmo *token bucket* (sólo marcado). Para la implementación del AF PHB probamos diferentes técnicas: una única cola FIFO con WRED, CBWFQ (dos colas una para cada tipo de paquete *in* o *out*) sin WRED y CBWFQ (una cola para cada red) con WRED. Para cada una de estas técnicas analizamos tres escenarios diferentes: sin variación de contratos, con variación de contratos y diferente número de fuentes dentro de cada red.

Los resultados demuestran que con todos los mecanismos se pueden garantizar los contratos gracias al caudal de paquetes *in* (con mayor o menor exactitud). El problema aparece a la hora de distribuir el ancho de banda excedente. En general, las fuentes con menor contrato se benefician notablemente del ancho de banda en exceso en perjuicio de las fuentes con contratos mayores. La única implementación que permite obtener un reparto justo (equitativo) es CBWFQ con WRED. No obstante, para configurar esta implementación es requisito imprescindible conocer los contratos de todos los usuarios. Además, el tráfico se separa dentro de cada nodo en función de la red LAN origen, por lo que dentro de cada nodo habrá tantas colas para el servicio AF como redes origen. Lo que claramente convierte a este esquema en poco escalable. En consecuencia, podemos concluir que con las soluciones disponibles en los equipos actuales para DiffServ, y en definitiva para QoS, no es posible ofrecer un servicio asegurado AF con todas las garantías a los usuarios finales. Siendo necesario el desarrollo de nuevas técnicas al respecto y la inclusión de éstas en equipos comerciales.

Glosario

A

AF PHB	PHB de encaminamiento asegurado (*Assured Forwarding PHB*)
AS	Servicio asegurado (*Assured Service*)
ATM	Modo de transferencia asíncrono (*Asynchronous Transfer Mode*)

B

BE	Mejor esfuerzo (*Best-Effort*)

C

CB	Acondicionador de tráfico basado en contadores (*Counters Based*)
CBDQ	CB empleado junto con una gestión de buffer de doble cola (*CB Dual Queuing*)
CBM	Acondicionador de tráfico basado en contadores modificado (*CB Modified*)
CBWFQ	WFQ basado en clases (*Class Based Weigh Fair Queuing*)
CIR	Tasa de información contratada (*Commited Information Rate*)
CR	Nodo interior (*Core Router*)

D

DiffServ	Servicios diferenciados (*Differentiated Services*)
DSCP	Código diffServ (*DiffServ Code Point*)

E

EBM	Marcado basado en ecuación (*Equation Based Marking*)
ECN	Notificación de congestión explícita (*Explicit Congestion Notification*)
EF PHB	PHB de encaminamiento expedito (*Expedited Forwarding PHB*)
ETSW	Ventana temporal deslizante mejorada (*Enhanced Time Sliding Window*)
ER	Nodo frontera (*Edge Router*)
EWND	Ventana de exceso (*Excess Window*)

F

FIFO	Primero en entrar primero en salir (*First In First Out*)
FQ	Encolamiento justo (*Fair Queuing*)
FRED	Descarte temprano aleatorio justo (*Fair Random Early Drop*)

G

GFR	Velocidad de trama garantizada (*Guaranteed Frame Rate*)

H

HBO	Ocupación alta de cola (*High Buffer Occupancy*)

I

IETF	Grupo para el desarrollo de Internet (*Internet Engineering Task Force*)
IntServ	Servicios integrados (*Integrated Services*)
IP	Protocolo de Internet (*Internet Protocol*)
ISP	Proveedor de servicios de Internet (*Internet Service Provider*)

L

LAN	Red de área local (*Local Area Network*)
LB	Cubo licuante (*Leaky Bucket*)

M

MPLS	Conmutación por etiquetas multiProtocolo (*MultiProtocol Label Switching*)
MSS	Tamaño máximo de segmento (*Maximum Segment Size*)

P

PAT	Tiempo de llegada del paquete (*Packet Arrival Time*)
PETER	Acondicionador de tráfico con reparto proporcional (*Proportional Excess Traffic conditionER*)
PHB	Comportamiento por salto (*Per Hop Behavior*)
PQ	Planificación por prioridad (*Priority Queuing*)

Q

QoS	Calidad de servicio (*Quality of Service*)

R

RDSI	Red Digital de Servicios Integrados
RED	Detección temprana de congestión (*Random Early Detection*)
RIO	RED *In* y *Out* (*RED In and Out*)
RTT	Tiempo de ida y vuelta (*Round Trip Time*)
RSVP	Protocolo de reserva de recursos (*Resource reSerVation Protocol*)
RWND	Ventana reservada (*Reserved Window*)

S

SLA	Acuerdo de nivel de servicio (*Service Level Agreement*)
SLS	Especificación de nivel de servicio (*Service Level Specification*)

T

TAT	Tiempo de llegada teórico (*Theoretical Arrival Time*)
TATC	Acondicionador de tráfico conocedor del tráfico (*Traffic Aware Traffic Conditioner*)

TC	Acondicionador de tráfico (*Traffic Conditioner*)
TCS	Especificación de acondicionado del tráfico (*Traffic Conditioning Specification*)
TCP	Protocolo de control de transporte (*Transport Control Protocol*)
TSW	Ventana temporal deslizante (*Time Sliding Window*)

U

UDP	Protocolo de datagramas de usuario (*User Datagram Protocol*)
USD	Diferenciación compartida de usuario (*User Shared Differentiation*)

V

VoIP	Voz sobre IP (*Voice over IP*)

W

WFQ	Planificación justa por pesos (*Weighted Fair Queuing*)
WRED	RED por pesos (*Weighted RED*)

Referencias bibliográficas

[ALVE00] I. Alves, J. De Rezende, L. De Moraes, "Evaluating Fairness in Aggregated Traffic Marking", *Proceedings* de IEEE Global Telecommunications Conference Globecom'00, pp. 445-449, San Francisco, EEUU, Noviembre 2000.

[ANDR00] I. Andrikopoulos, L. Wood, G. Pavlou, "A Fair Traffic Conditioner for the Assured Service in a Differentiated Services Internet", *Proceedings* de IEEE International Conference on Communications ICC'00, Vol. 2, pp. 806-810, New Orleans, LA, Junio 2000.

[BONA97] O. Bonaventure, "A Simulation Study of TCP with the Proposed GFR Service Category", *Proceedings* de High-Performance Networks for Multimedia Applications, DAGSTUHL Seminar, Germany, Junio 1997.

[BONI00] V. Bonin, F. Cerdan, O. Casals, "A Simulation Study of Differential Fair Buffer Allocation", *Proceedings* de 3rd International Conference on ATM, ICATM'00, pp. 365-372, Germany, Junio 2000.

[BONI01] V. Bonin, O. Casals, B. Van Houdt, C. Blondia, "Performance Modeling of Differentiated Fair Buffer Allocation", *Proceedings* de 9th International Conference on Telecommunications Systems, pp. 199-214, Dallas, EEUU, Marzo 2001.

[CANO01] Maria-Dolores Cano, Josemaria Malgosa-Sanahuja, Fernando Cerdan, Joan Garcia-Haro, "Internet Measurements and Data Analysis over the Spanish Regional Network Ciez@net", *Proceedings* de IEEE Pacific RIM Conference on Communications, Computers & Signal Processing PACRIM'01, Vol. II, pp. 393-396, Victoria, Canada, Agosto 2001.

[CANO01a] Maria-Dolores Cano, Fernando Cerdan, Joan Garcia-Haro, Josemaria Malgosa-Sanahuja, "Performance Evaluation of Traffic Conditioner Mechanisms for the Internet Assured Service", *Proceedings* de SPIE Quality of Service over Next-Generation Data Networks, Vol. 4524, pp. 182-193, Denver, EEUU, Agosto 2001.

[CANO01b] Maria-Dolores Cano, Josemaria Malgosa-Sanahuja, Fernando Cerdan, Joan Garcia-Haro, "Análisis y Caracterización de Tráfico IP en la Red Regional Ciez@net", *Proceedings* de Jornadas de Ingeniería Telemática JITEL'01, pp. 529-530, Barcelona, España, Septiembre 2001."

[CANO02] Maria-Dolores Cano, Fernando Cerdan, Joan Garcia-Haro, Josemaria Malgosa-Sanahuja, "A New Proposal for Assuring Services in Internet", *Proceedings* de Internet Computing IC'02, CSREA Press, Vol. II, pp. 379-384, Las Vegas, EEUU, Junio 2002.

[CANO02a] Maria-Dolores Cano, Fernando Cerdan, Joan Garcia-Haro, Josemaria Malgosa-Sanahuja, "Counters-Based Modified Traffic Conditioner", *Lecture Notes in Computer Science* (QofIS 2002), Vol. 2511, pp. 57-67, Springer-Verlag, 2002.

[CANO03] Maria-Dolores Cano, Fernando Cerdan, Joan Garcia-Haro, Josemaria Malgosa-Sanahuja, "Performance Evaluation of the Counters-Based Modified Traffic Conditioner in a DiffServ Network", *Proceedings* de International Symposium on Computers and Communications ISCC'03, Vol. I, pp. 305-311, Antalya, Turquia,

Julio 2003.

[CANO03b] María-Dolores Cano, Juan Jose Alcaraz, Pablo Lopez-Matencio, Fernando Cerdan, "CBDQ: Garantía de Calidad de Servicio en Internet", *Proceedings* de XIII Jornadas Telecom. I+D 2003, Madrid, Noviembre 2003.

[CANO03a] Maria-Dolores Cano, Fernando Cerdan, Joan Garcia-Haro, Josemaria Malgosa-Sanahuja, "Análisis de las Prestaciones del Acondicionador de Tráfico *Counters-Based Modified* en un Dominio DiffServ", *Proceedings* de Jornadas de Ingeniería Telemática JITEL'03, pp. 1-8, Las Palmas de Gran Canaria, España, Septiembre 2003.

[CANO04] Maria-Dolores Cano y Fernando Cerdan "End-to-End TCP Performance of the Couple CBM Traffic Conditioner and RIO Buffer Management in a Three-Node Topology", *Proceedings* de International Conference on Communications and Computer Networks CCN'04, MIT, Cambridge, EEUU, Noviembre 2004.

[CANO04a] Maria-Dolores Cano, Pablo Lopez-Matencio, Juan Jose Alcaraz , Fernando Cerdan, "El Papel de los Acondicionadores de Tráfico para Ofrecer Calidad de Servicio Extremo a Extremo", Revista II Teleco-Forum, Universidad Politécnica de Cartagena, pp. 80-82, Cartagena, España, Abril 2004.

[CANO04b] Maria-Dolores Cano, Fernando Cerdan, "CBM: Guarantees for a Complete Internet Assured Service", en proceso de revision en *Journal of Communications and Networks*.

[CANO04c] Maria-Dolores Cano, Fernando Cerdan, Joan Garcia-Haro "Traffic Control for an Effective Internet Service Differentiation", en proceso de revisión en *International Journal of Electronics and Communications*.

[CANO04d] Maria-Dolores Cano, Fernando Cerdan "An Experimental Study of Bandwidth Assurance in a Diffserv Network", en proceso de revisión en International Conference on Internet and Multimedia Systems and Applications EuroIMSA'05.

[CERD00] F. Cerdan, J. Malgosa, J. Garcia-Haro, F. Monzo, F. Burrull, "Providing QoS to TCP/IP Traffic: an Overview", *Proceedings* de International Workshop in Protocols for Multimedia Systems PROMS'00, pp. 91-99, Crackow, Polonia, Octubre 2000.

[CERD00a] F. Cerdan, O. Casals, "Mapping an Internet Assured Service on the GFR ATM Service", *Lecture Notes in Computer Science* (Networking 2000), Vol. 1815, pp. 398-409, Ed. Springer-Verlag, Mayo 2000.

[CERD00b] F. Cerdan, O. Casals, "Performance of Different TCP Implementations over the GFR Service Category", *ICON Journal, Special Issue on QoS Management in Wired &Wireless Multimedia Communications Network*, Vol.2, pp. 273-286, Baltzer Science, Enero 2000.

[CISC04] "Congestion Management Overview", disponible en <www.cisco.com>.

[CISC04a] WRED, "Configuring Weighted Random Early Detection CISCO IOS Quality of Service Solutions Configuration Guide", disponible en <www.cisco.com>

[CLAR98] D. Clark and W. Fang, "Explicit Allocation of Best-Effort Packet Delivery Service", *IEEE/ACM Transactions on Networking*, Vol. 6 No. 4, pp. 362-373, Agosto 1998.

[ELLO99] O. Elloumi, S. De Cnodder, K. Pauwels, "Usefulness of the Three Drop Precedences in Assured Forwarding Service", Internet draft, work in progress, Julio 1999

[FENG97] W. Feng, D. Kandlur, D. Saha, K. Shin, "Understanding TCP Dynamics in an Integrated Services Internet", *Proceedings* de ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video NOSSDAV'97, pp. 295-306, Mayo 1997.

[FLOY01] J. Padhye, S. Floyd, "On Inferring TCP Behavior", *Proceedings* de ACM International Conference of the Special Interest Group on Data Communication SIGCOMM'01, Vol. 31 Issue 4, pp. 287-298, San Diego, EEUU, Agosto 2001.

[FLOY93] S. Floyd y V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, Vol. 1 No.4, pp. 397-413, Agosto 1993.

[GEND02] Mohamed A. El-Gendy y Kang G. Shin, "Equation-Based Packet Marking for Assured Forwarding Services", *Proceedings* de IEEE Infocom 2002, Vol. 2, pp. 845-854, New York, EEUU, Junio 2002.

[GEND03] Mohamed A. El-Gendy, Kang G. Shin, "Assured Forwarding Fairness Using Equation-Based Packet Marking and Packet Separation", *Computer Networks*, Vol. 41 No. 4, pp. 435-450, 2003.

[GOYA99] M. Goyal, A. Durresi, P. Misra, C. Liu, R. Jain, "Effect of Number of Drop Precedences in Assured Forwarding", *Proceedings* de IEEE Global Telecommunications Conference Globecom'99, Vol. 1(A), pp. 188-193, Rio de Janeiro, Brasil, Diciembre 1999.

[HABI02] Ahsan Habib, Bharat Bhargava, Sonia Fahmy, "A Round Trip Time and Time-out Aware Traffic Conditioner for Differentiated Services Network", *Proceedings* de IEEE International Conference on Communications ICC'02, Vol. 25 No. 1, pp. 981-985, New York, EEUU, Abril 2002.

[IBAN98] J. Ibanez y K. Nichols, "Preliminary Simulation Evaluation of an Assured Service", Internet draft, work in progress, draft-ibanez-diffserv-assured-eval-00.txt, Agosto 1998.

[JAIN91] R. Jain, *The Art of Computer Systems Performance Analysis*, John Wiley and Sons Inc., 1991.

[JIE99] Wu Jie, Feng Zhenming, Yuan Jian, "Differentiated Services TCP Algorithm for the Internet", *Electronics Letters*, Vol.35, pp. 1513-1515, Septiembre 1999.

[KIM99] H. Kim, "A Fair Marker", Internet draft, work in progress, Abril 1999.

[KUSMI00] E. Kusmierek y R. Koodli, "Random Packet Marking for Differentiated Services", UMN Technical Report TR-00-020, Dept. of Computer Science & Eng., University of Minnesota, 2000.

[LIN97] D. Lin y R. Morris, "Dynamics of Random Early Detection", *Proceedings* de ACM International Conference of the Special Interest Group on Data Communication SIGCOMM'97, Vol. 27 No. 4, pp. 127-137, Octubre 1997.

[LIN99] W. Lin, R. Zheng, J. Hou, "How to Make Assured Service more Assured", *Proceedings* de 7th International Conference on Network Protocols ICNP'99, pp. 182-191, Toronto, Canadá, Octubre 1999.

[NAND00] B. Nandy, N. Seddigh, P. Pieda, J. Ethridge, "Intelligent Traffic Conditioners for Assured Forwarding Based Differentiated Services Networks", *Proceedings* de Networking 2000, pp. 540-554, París, Francia, Mayo 2000.

[NETP04] NetPerf, disponible en <www.netperf.org>

[OPER01] Operax, "The Challenge of Enabling QoS in IP Networks", disponible en <www.operax.com/docs/whitepaper-QoS-challenge-C.pdf>, Octubre 2001.

[PARK00] Won Hyong Park, "A Modified RIO Algorithm that Alleviates the Bandwidth Skew Problem in Internet Differentiated Services", *Proceedings* de IEEE International Conference on Communications ICC'00, pp. 1599-1603, Vancouver, Canadá, Junio 2000.

[REZE99] J. F. De Rezende, "Assured Service Evaluation", *Proceedings* of IEEE Global Telecommunications Conference Globecom'99, pp. 100-104, Rio de Janeiro, Brasil, Diciembre 1999.

[RFC1633] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, Junio 1994.

[RFC2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205 (Actualizada en RFC2750), Septiembre 1997.

[RFC2474] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, Diciembre 1998.

[RFC2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC 2475, Diciembre 1998.

[RFC2597] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, Junio 1999.

[RFC2697] J. Heinanen y R. Guerin, "A Two Rate Three Color Marker", RFC 2697, Septiembre 1999.

[RFC2698] J. Heinanen y R. Guerin, "A Single Rate Three Color Marker", RFC 2698, Septiembre 1999.

[RFC3168] K. Ramakrishnan, S. Floyd, D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168 Septiembre 2001.

[RFC3246] B. Davie, A. Charny, J.C.R. Bennet, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, Marzo 2002.

[RFC3260] D. Grossman, "New Terminology and Clarifications for DiffServ", RFC 3260, Abril 2002.

[SEDD98] N. Seddigh, B. Nandy, P. Pieda, J. Hadi Salim, A. Chapman, "An Experimental Study of Assured Services in a DiffServ IP QoS Network", SPIE Symposium on Voice, Video and Data Communications (QoS Issues related to the Internet), Vol. 3529, Boston, EEUU, Noviembre 1998.

[SEDD99] N. Seddigh, B. Nandy, P. Pieda, "Bandwidth Assurance Issues for TCP flows in a Differentiated Services Network", *Proceedings* de IEEE Global Telecommunications Conference Globecom'99, Vol. 3, pp. 1792-1798, Rio de Janeiro, Brasil, Diciembre 1999.

[STEV94] W. R. Stevens, *The Protocols (TCP/IP Illustrated, Volume I)*, Addison-Wesley Professional, 1st edition, 1994.

[SU03] Hongjun Su, Mohammed Atiquzzaman, "ItswTCM: a New Aggregate Marker to Improve Fairness in DiffServ", *Computer Communications* 26, pp. 1018-1027, 2003.

[TART02] S. Tartarelli y A. Banchs, "Random Early Marking: Improving TCP Performance in DiffServ Assured Forwarding", *Proceedings* de IEEE International Conference on Communications ICC'02, Vol. 25 No. 1, pp. 970-975, New York, EEUU, Mayo 2002.

[WANG98] Z. Wang, "USD: Scalable Bandwidth Allocation for the Internet", *Proceedings* de IFIP Conference on High Performance Networking HPN'98, pp. 351-361, Viena, Austria, Septiembre 1998.