ATM Sobre Ethernet Mediante el Protocolo CIF

J. M. Arco, B. Alarcos, A. M. Hellín, A. García, D. Meziat
Universidad de Alcalá
Escuela Politécnica
28871 Alcalá de Henares
Teléfono: +34 91 8856627-28
{jmarco, bernardo, hellin, antonio, meziat}@aut.alcala.es

Abstract

This paper describes the research carried out to obtain the advantages of ATM technology over legacy Ethernet networks. A protocol called "Cells In Frame" (CIF) allows to encapsulate ATM cells in Ethernet frames. We have implemented the CIF protocol and several queuing algorithms in the kernel of the Linux operating system. Measures of bandwidth allocation delay and throughput have been carried out in order to check that the developed system fulfils the requirements of the multimedia traffic. Obtained results show that it provides the necessary ATM quality of service, paying a minimum throughput decrease respect to native Ethernet. Also a queuing algorithm that allows to optimize of traffic management is being researched.

1 Introducción

El objetivo global de nuestro trabajo consiste en aportar las ventajas de la calidad de servicio de ATM a redes clásicas como Ethernet. Se pretende conseguir de esta forma llevar las ventajas de esta tecnología hasta el usuario sin necesidad de realizar las inversiones para migrar a ATM [1].

El escenario en el que se pretende aplicar este desarrollo, es el utilizado en la mayoría de los entornos de trabajo, redes LAN interconectadas mediante WAN. La tendencia actual en las redes WAN es la utilización de la tecnología ATM [2][3]. En cuanto a las redes de área local la tecnología dominante en la actualidad es Ethernet, no siendo probable que ATM se imponga en este entorno a medio plazo [4] debido a que, entre otros, presenta el inconveniente de requerir mayores inversiones, tanto en el conmutador como en el adaptador de red, además de ser una tecnología más compleja. Sin embargo, Ethernet presenta el inconveniente de no ser adecuada para la transmisión de vídeo y audio, ya que no ofrece garantía de calidad de servicio. Para superar esta dificultad se puede introducir algunas variaciones sobre la red Ethernet y dotarla de las prestaciones de ATM, sin perder las propias.

Una posibilidad para ello consiste en la utilización de un nuevo protocolo que integre la red local dentro de ATM, este protocolo es el denominado "Cells In Frame" (CIF) [5]. La implantación de CIF requiere dos actuaciones, añadir software en los sistemas finales, y el desarrollo de un conmutador especial que permita conectar ambas tecnologías. La topología genérica se ilustra en la figura 1.

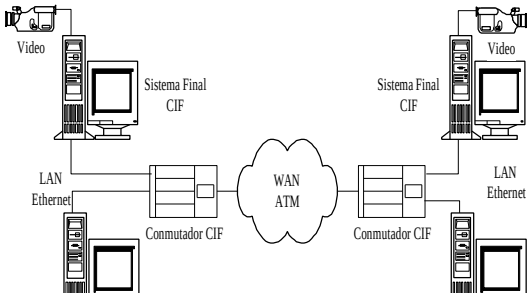

Fig. 1. Topología de red.

En el sistema final, CIF envía dentro de una trama Ethernet una cabecera específica y la carga útil de varias celdas ATM. La función del conmutador CIF consiste en extraer la carga de las celdas ATM de las tramas Ethernet, reconstruir su cabecera y enviarlas a la red ATM. Con CIF también conseguimos que la señalización ATM llegue al usuario, manteniendo la tecnología Ethernet.

2 Arquitectura de Protocolos CIF

La arquitectura de protocolos a desarrollar, representada en la figura 2, debe cumplir la especificación CIF 1.0 [6] del Forum CIF.

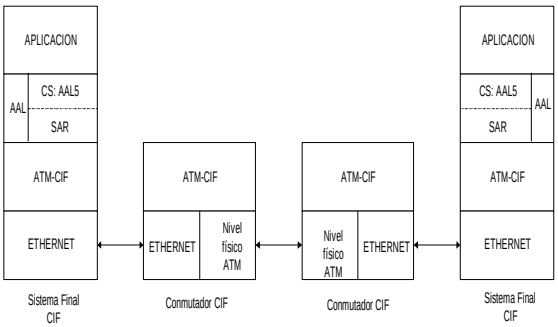

Fig. 2. Torre de protocolos.

La torre de protocolos ATM-CIF es similar a la ATM [7], con la diferencia de que el nivel físico es Ethernet. A continuación se describen brevemente

las particularidades de cada nivel de CIF respecto a ATM.

La capa de adaptación a ATM, AAL (ATM Adaptation Layer), proporciona una interfaz estándar a los niveles superiores, que pueden ser directamente aplicaciones ATM nativas, u otra pila de protocolos como TCP/IP.

En cuanto a la subcapa de convergencia CS, con CIF se puede utilizar cualquier AAL. Normalmente se utiliza AAL5 por ser la más eficiente.

La subcapa de segmentación y reensamblado (SAR), se encarga de trocear la información en los bloques de 48 bytes que forman la celda. En CIF, como se explica en el siguiente párrafo, no se realiza esta función salvo que la longitud de los datos de usuario sea mayor de 1480 bytes, límite para no superar la capacidad de la trama Ethernet.

La capa ATM-CIF se diferencia de una capa ATM en que no tiene que generar una cabecera por cada celda, si no una cabecera patrón para un grupo de celdas pertenecientes a la misma conexión, y que se encapsulan en una misma trama Ethernet. Esto se hace por motivos de eficiencia.

El conmutador CIF debe realizar varias funciones:

- Control de los Parámetros de Uso (UPC), que simula la función policía de la red ATM, para comprobar que el tráfico recibido está de acuerdo con la velocidad pactada. Para implementar esta función se utiliza el algoritmo del cubo con goteo (leaky bucket).
- Extracción de las celdas de la trama Ethernet y su inserción en la red ATM.
- Gestión de las colas de salida para garantizar la calidad de servicio.
- Mantenimiento del enlace con los sistemas finales.

La figura 3 ilustra el flujo de datos por los distintos niveles, suponiendo que la longitud máxima del mensaje es 1480 bytes.

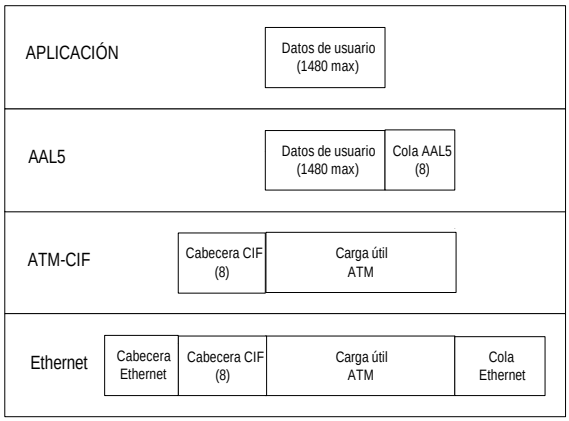

Fig. 3. Flujo de datos

La información de la cabecera CIF, determina cómo se van a realizar un serie de procedimientos, tales como el control de errores, la señalización ATM y el mantenimiento del enlace CIF. También indica si los tiene que realizar el conmutador o el sistema final CIF.

3 Implementación realizada de CIF

3.1 Escenario

Se está desarrollando la implementación de un sistema final y un conmutador CIF, de acuerdo con el esquema de la figura 1. El desarrollo se ha realizado sobre el sistema operativo Linux, de dominio público, muy difundido y que además dispone de fuentes, depuradores y documentación para desarrollo.

La primera fase de la implementación se ha realizado de acuerdo con el esquema de la figura 4, que es una simplificación de la red de la figura 1. La simplificación consiste en unir dos PCs, (que actúan de sistemas finales) mediante un único conmutador CIF. Hay que indicar que este no es el escenario final pero se puede considerar válido para garantizar la QoS.

El enlace entre el PC y el conmutador CIF es Ethernet half-duplex, esto puede producir problemas de compartición del medio si la carga es elevada. Para evitarlo habría que implementar un control de admisión de conexiones, o utilizar un Ethernet full-duplex.

Además es un escenario real que se da cuando se quiere transmitir con calidad de servicio en conexiones LAN-LAN.

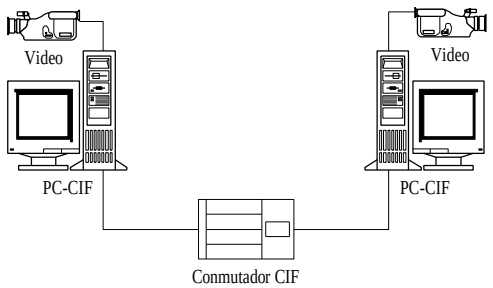

Fig. 4. Escenario considerado.

3.2 Descripción de la implementación

En el sistema final se ha desarrollado una torre de protocolos CIF dentro del núcleo, para que pueda ser usada eficientemente por varias aplicaciones [8]. El conmutador CIF consiste en un PC y sus correspondientes tarjetas de red, en el que se ha introducido una torre CIF específica.

Se han implementado los aspectos básicos de la norma CIF, no incluyendo en esta primera fase algunos aspectos, como por ejemplo el tratamiento de errores de la cabecera ATM (HEC) ya que en la práctica son muy poco frecuentes, y en caso de producirse su detección se realiza a nivel Ethernet. Esta primera fase soporta tráfico CBR y UBR.

Para facilitar las tareas de desarrollo y depuración se ha realizado la implementación de la torre de protocolos CIF, en un módulo que permite compilarlo independientemente y cargarlo dinámicamente en el núcleo [9].

3.2.1 Arquitectura de comunicaciones

Para realizar la implementación en el núcleo se ha aprovechado la arquitectura de comunicaciones utilizada en Linux, añadiendo a la familia de protocolos estándar [10], uno nuevo (que denominamos *AF_CIF)* y una estructura de direcciones específica (*sockaddr_cif).*

Cuando un proceso se comunica a través de la red [11] utiliza las funciones de la capa *socket* BSD (figura 5). Esta capa realiza tareas de administración de *sockets*, utilizando para ello unas estructuras generales de datos (*socket* y *msghdr*). El nivel *socket* BSD simplifica la portabilidad de las aplicaciones de red.

Debajo de esta capa se encuentra otra denominada *socket* INET, que hace de interfaz entre el nivel socket BSD y los protocolos específicos, en nuestro caso CIF. Esta capa utiliza la estructura *sock.*

Entre la capa CIF y los dispositivos de red se encuentra un interfaz común (*dev.c*) y un driver específico para cada dispositivo.

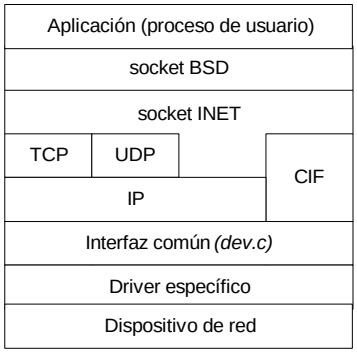

Fig. 5. Torre implementada.

En la figura 6 se muestra la secuencia de los módulos que intervienen en el proceso de comunicaciones.

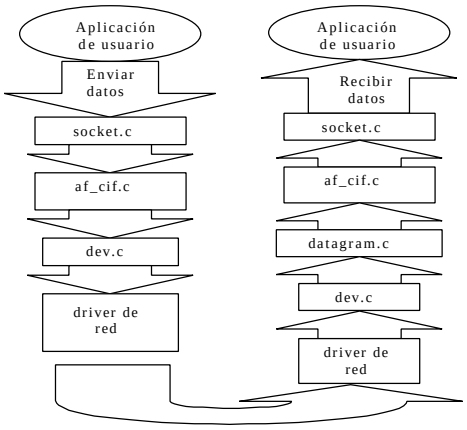

Fig. 6. Flujo de programas.

Las funciones implementadas en el conmutador CIF son el establecimiento del enlace la retransmisión de tramas y la gestión del tráfico.

3.2.2 Algoritmos de encolado

Se ha realizado un análisis y programación de diferentes algoritmos de la familia WFQ (Weight Fair Queuing) [12][13], para ver cual se adapta mejor a nuestro sistema. Comenzamos con un algoritmo sencillo, SCFQ (Self-Clocked Fair Queuing)[14] y continuamos con FFQ (Frame-Base Fair Queuing) y SPFQ (Starting Potential-Based Fair Queuing) [15].

En los algoritmos WFQ cuando llega una trama se calcula y asocia una etiqueta (Time Stamp, *TS*) que va a determinar el orden de salida y se envía a la cola de su conexión. La siguiente trama a transmitir será la que tenga el valor de *TS* más pequeño. *TS* se calcula de acuerdo a la fórmula:

$$TS_i^k = max(v(t), TS_i^{k-1}) + \frac{L_i^k}{f_i r}$$

donde:

v(t) es la función de tiempo virtual que representa el reparto de ancho de banda realizado por el sistema.

TS_i^{k-1} corresponde a la trama anterior de esa sesión.

L_i^k es el tamaño de la trama en bits.

f_i es el peso normalizado de la sesión.

r es la velocidad de transmisión.

Cuando una sesión tenga tráfico pendiente se cumple que $v(t) \leq TS_i^{k-1}$ por lo que *v(t)* no influye en *TS*. En caso contrario, al recibir la primera trama después de un periodo sin tráfico, se tiene en cuenta *v(t)* para actualizar la sesión. La dificultad del planificador WFQ radica en calcular *v(t)*, por lo que se han propuesto otros algoritmos que simplifican su cálculo.

El algoritmo SCFQ aproxima *v(t)* utilizando el valor de la etiqueta de la última trama transmitida.

Los algoritmos FFQ y SPFQ son del tipo RPS (Rate Proportional Server). Estos algoritmos se basan en el cálculo y calibración de la función potencial, cuyo incremento para una sesión durante el periodo de tiempo [τ,t), se define como:

$$P_i(t) - P_i(t) = \frac{W_i(t,t)}{r_i}$$

donde:

$W_i(t,t)$ es el servicio recibido por la sesión *i* durante el periodo [τ,t).

r_i es la velocidad mínima de la sesión *i*.

Se define la función potencial del sistema como:

$$P(t) = F(P_1(t-), P_2(t-), \ldots, P_N(t-), t)$$

es decir, depende de las funciones potenciales de cada sesión y del tiempo. En los algoritmos RPS, *P(t)* realiza una función similar a *v(t)*.

P(t) tiene pendiente 1 y debe calibrase periódicamente para reflejar el comportamiento real del sistema. En FFQ la calibración se realiza después de transmitir *F* bits y en el SPFQ cada vez que se transmite una trama.

3.2.3 Implementación en el conmutador

En la figura 7 puede verse el esquema básico desarrollado. En el funcionamiento estándar de Linux, todos los mensajes seguirían el mismo tratamiento a través de una única cola. De este modo, si enviásemos un mensaje CIF, pasaría a través de las funciones *dev_queue_xmit()* y *do_dev_queue_xmit()*, que encolan la trama y a través de la función *dev->hard_start_xmit()* que llama al driver específico del dispositivo de red.

Para obtener QoS se ha modificado el esquema anterior (contenido en el fichero *dev.c*) añadiendo tantas colas como conexiones haya, e implementando el algoritmo de encolado para gestionarlas.

Conviene destacar, que se han modificado campos de la estructura *device*, que es la que aporta las funciones necesarias para la comunicación con los drivers de red [16]. La modificación fundamental ha ido orientada al incremento del número de colas mediantes el campo *DEV_NUMBERS*. El array de estructuras *sk_buff_head buffs [DEV_NUMBERS]* permite acceder a las mismas.

Cada trama está contenida en una estructura *sk_buff*. Una cola está formada por estructuras *sk_buff* doblemente enlazadas, que son gestionadas por la estructura *sk_buff_head*. Para manejar las colas se utilizan las funciones *skb_dequeue()* y *skb_queue_tail()* para encolado y desencolado respectivamente [17].

El proceso de entrada del algoritmo clasifica la trama y calcula la etiqueta de salida. El de salida selecciona la trama con etiqueta de salida más baja, para su transmisión.

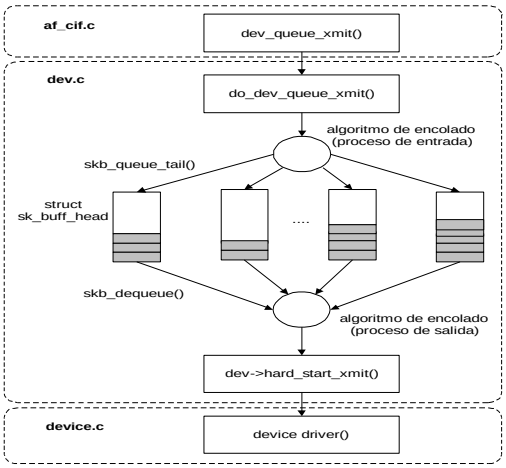

Fig. 7. Gestión de tráfico.

Inicialmente se ha realizado la señalización para el establecimiento de la conexión y la especificación de sus parámetros, de forma manual. Hay una serie de circuitos permanentes y desde la aplicación, antes de enviar datos, se envían al conmutador CIF los parámetros de cada conexión.

4 Pruebas

Las pruebas se han realizado sobre dos escenarios distintos como el de la figura 4, uno con PCs a 133 MHz y otro a 266 MHz, para comprobar la evolución de las características. Las pruebas

permiten comparar el rendimiento de los algoritmos implementados (SCFQ, SPFQ, FFQ).

Se han realizado medidas de:

- Reparto de ancho de banda.
- Retardo.
- Throughput [18].

Todas las pruebas se han realizado con un tamaño de datos de 1480 bytes y con las sesiones transmitiendo a la máxima velocidad.

Para la prueba de reparto de ancho de banda, hay que tener en cuenta que los planificadores de tráfico deben garantizar a las sesiones el ancho de banda asignado. La capacidad sobrante, ya sea por ausencia de tráfico o por no estar asignado, se reparte en proporción a sus reservas. Las sesiones sin QoS sólo recibirán servicio cuando no haya tráfico de sesiones con QoS.

Para medir el retardo que introduce el algoritmo, se mide el RTT (Round Trip Time) entre sistemas finales.

Para la medida del reparto de ancho de banda y throughput se ha empleado dos PCs (uno de ellos el usado como conmutador) como transmisores con un proceso de dos sesiones y un PC como receptor con un proceso de cuatro sesiones.

La pérdida de throughput debida al planificador se ha medido repitiendo la prueba y aumentando sucesivamente el número de tramas transmitidas, hasta que se empiezan a perder.

4.1 Resultados

4.1.1 Asignación de ancho de banda.

La tabla 1 indica el peso de las cuatro sesiones. Los resultados se han obtenido tomando medidas de tiempo en cada trama transmitida a la salida del interfaz común (*dev.c*).

Tabla 1: Asignación de ancho de banda.

BW reservado	BW teórico	FFQ	SPFQ	SCFQ
0,5	0,625	0,628	0,626	0,615
0,2	0,250	0,248	0,250	0,262
0,1	0,125	0,124	0,123	0,123
0	0	0	0	0

No se observan diferencias en los resultados entre el escenario con PCs a 133 y a 266 MHz.

4.1.2 Retardo

Se ha medido el RTT durante un periodo de 20 segundos, el cual es suficientemente grande para evitar el efecto de la ejecución de algún demonio. Con el tamaño de 1480 bytes se mide el retardo máximo [19].

Tabla 2. Retardo en PC-133

	RTT (μsec.)	Diferencia (μsec.)	Diferencia en %
FIFO	8523	0	
SCFQ	8536	13	0,15
SPFQ	8561	38	0,44
FFQ	8558	35	0,41

Tabla 3. Retardo en PC-266

	RTT(μsec.)	Diferencia (μsec.)	Diferencia en %
FIFO	5163		
SCFQ	5169	6	0,12
SPFQ	5173	10	0,19
FFQ	5171	8	0,15

4.1.3 Throughput

Dado que el throughput depende de numerosos factores (número de sesiones, número de procesos, pesos de las sesiones, modelo de tráfico, tamaño de los mensajes, tamaño de las colas, sentido del tráfico, etc), estos resultados deben interpretarse como una simplificación válida para la finalidad perseguida.

Tabla 4. Throughput en PC-133

	Throughput (Mbps.)	Diferencia (Kbps.)	Diferencia en %
FIFO	9,7248		
SCFQ	9,707	8	0,08
SPFQ	9,70	24	0,25
FFQ	9,680	44	0,45

Tabla 5. Throughput en PC-266

	Throughput (Mbps.)	Diferencia (Kbps.)	Diferencia en %
FIFO	9,72749		
SCFQ	9,7055	1	0,01
SPFQ	9,724	3	0,03
FFQ	9,7245	3	0,03

4.2 Análisis de resultados

No se aprecian grandes diferencias de rendimiento entre algoritmos, observándose que a mayor velocidad del procesador estas se reduce.

El FFQ tiene mejores resultados que el SPFQ ya que en el segundo las actualizaciones de la función potencial son más frecuentes. La ventaja del SPFQ es que transmitirá tramas de manera más justa que el FFQ.

Los resultados de la distribución del ancho de banda se corresponden con los calculados teóricamente.

Los valores de retardo son análogos a los medidos en otro escenario similar [20]. En cuanto al throughput las diferencias medidas son menores que en dicho escenario, lo cual puede ser debido a la utilización de algoritmos diferentes.

5 Conclusiones y líneas de futuros trabajos

La introducción del protocolo CIF permite que se puedan respetar parámetros garantizados del tráfico para distintos tipos de servicios. Estas garantías no son posibles utilizando Ethernet pura, al no poder especificar las características de cada tipo de tráfico.

Se ha realizado una implementación en el núcleo del sistema operativo, que permite ser usada por varios procesos. También se ha implementado un conmutador que soporta calidad de servicio.

Seguimos investigando en la búsqueda del algoritmo de encolado más eficaz para nuestro objetivo. Actualmente trabajamos en un nuevo algoritmo basado en el DRR que aporte una mejora en la uniformidad de la distribución del tráfico, para lo cual nos estamos apoyando en técnicas de simulación.

Para completar el escenario real, queda pendiente la incorporación de interfaces ATM en los conmutadores. También se completará la especificación CIF y el soporte de IP clásico que permita junto a las aplicaciones ATM nativas, la comunicación con otros ordenadores TCP/IP.

Referencias

[1] J. M. Arco, A. Martínez, B. Alarcos, A. García, D. Meziat. “Quality of service over Ethernet”. Online Educa Berlin, 2-4 December 1998, pp. 42-46.

[2] V. Castelo, “Informe de la gestión de RedIRIS durante el año 1997”. Boletín de la Red Nacional de I+D RedIris, número 41-42, páginas 5-7, diciembre 1997.

[3] A. Tanenbaum "Computer Networks, third edition", Prentice Hall, 1997.

[4] J. McQuillan, “Convergencia de routes y conmutadores”. Global Communication, número 10, páginas 42-43, enero 1998.

[5] L. Robers "CIF: Affordable ATM, At last" Data Communication, April, 1997.

[6] Cells In Frames Version 1.0, specification, http://www.ziplink.net/~lroberts/Atmf-961104.html

[7] Prycker M. "Asyncronous Transfer Mode, Solution for Broadband ISDN, Third Edition", Prentice Hall, 1997.

[8] Armitage, J. "The Application of Asynchronous Transfer Mode to Multimedia and Local Area Network" PhD thesis, January 1994. University of Melbourne, Australia.

[9] weikku.tkey.hut.fi/LDP/HOWTO/Module_HOWTO.html

[10] J.M. Arco, B. Alarcos, A. Domingo, “Programación de aplicaciones en redes de comunicaciones bajo entorno Unix”. Editado por la Universidad de Alcalá, 1997.

[11] M. Beck, H Bohme, M Dziadzka, U Kunitz, R Magnus, D Verworner. “Linux kernel internals” Second Edition, Addison-Wesley, 1998.

[12] Hui Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks" Proceeding of the IEEE, Vol 83, Nº 10 October 1995, pp. 1374-1396.

[13] W. Stalling, “High-speed networks TCP/IP ATM design principles”, Prentice Hall, 1998.

[14] S Golestani, "A self-clocked fair queuing scheme for broadband applications" Proceeding of the IEEE INFOCO ’94 Toronto, CA, June 1994, pp. 636-646.

[15] D. Stiliadis and A. Varma, "Hardware Implementation on Fair Queuing Algorithms for Asynchronous Transfer Mode Networks," IEEE Communications Magazine, Dec. 1997.

[16] Alessandro Rubini, “Linux device drivers”. O’Really & Associates, Inc.1998.

[17] Alan Cox, “Network Buffers and memory Management” Linux Journal, September 1996.

[18] S. Branner, “Bechmarking Terminology for Network Interconnection Devices”, RFC 1242.

[19] J. M. Arco, A. Martínez, B. Alarcos, A. García D. Meziat. “Implementación de ATM sobre Ethernet para aplicaciones de Teleenseñanza”, Actas IV Jornadas de Informática, Las Palmas de Gran Canaria, Julio 1998, pp. 459-465.

[20] Kenjiro Cho. "A Framework for Alternate Queueing: Towards Traffic Management by PC-UNIX Based Routers." In Proceedings of

USENIX 1998 Annual Technical Conference,
New Orleans LA, June 1998.