CAPÍTULO 8

MOTIVACIONES GENERALES

8.1. INTRODUCCIÓN

El principal objetivo de este capítulo es el de presentar las motivaciones generales de nuestra investigación, para poder argumentar el verdadero valor de las aportaciones realizadas. En la actualidad las redes ATM se están usando como la tecnología para soportar todo tipo de tráfico, pero con un destacable predominio de los protocolos TCP/IP. Por esto vamos a presentar los beneficios que nuestro mecanismo de recuperación de congestiones puede aportar, no sólo al trafico ATM nativo, sino también al tráfico generado por las fuentes TCP/IP.

El protocolo TCP se ha convertido en los últimos años en el estándar de comunicaciones de datos. Este es un protocolo fiable de la capa de transporte de la arquitectura TCP/IP, que usa control de error y control de flujo basados en ventana y se encarga del enrutamiento de paquetes en internet con control extremo-a-extremo [1].

Como ya hemos comentado en el *Capítulo 2*, existen numerosas investigaciones para conseguir integrar dos tecnologías tan diferentes como ATM y TCP/IP. Sin embargo, la integración de ambas se ha demostrado [2] con un pobre resultado en cuanto al comportamiento del throughput de TCP sobre ATM. Mientras ATM es, en líneas generales, una tecnología orientada a conexión, de conmutación de células de 53 octetos y de tamaño uniforme, TCP e IP se basan en mecanismos de enrutamiento de segmentos y datagramas de tamaño variable, generalmente de 1.500 octetos y mucho mayores de 48 octetos que es la unidad de datos básica de ATM. Así, cuando una fuente TCP emplea una red ATM como medio de transporte, los segmentos deben ser adaptados a las unidades de procesamiento que ATM es capaz de conmutar. Los segmentos TCP son encapsulados en PDU de la capa AAL-5, que serán luego segmentadas por la capa ATM en células de 48 octetos que son conmutadas en dirección al receptor del tráfico. Podemos decir, por tanto, que el pobre rendimiento de TCP se produce por su propio dinamismo que es a menudo empeorado cuando se implementa sobre ATM, y por la fragmentación de paquetes que ocurre cuando un paquete TCP fluye dentro de una conexión VPI/VCI a través AAL-5. La segmentación de AAL-5 es necesaria porque el tamaño típico de los segmentos TCP es mucho mayor que el de las células ATM. Además, el tráfico TCP es generalmente a ráfagas por lo que no permite su caracterización para conseguir predecir su comportamiento a priori. TCP transmite los segmentos tan rápido como es posible y usando una ventana para dar solución a las congestiones.

Las características que acabamos de comentar provocan un efecto bastante devastador en el throughput cuando los segmentos TCP atraviesan conmutadores ATM con tamaño de buffer mucho menor que el tamaño de ventana de TCP. Por esta causa las células se pierden y acaban generándose retransmisiones por timeout. Además, la pérdida de una sola célula dará como resultado la pérdida de un segmento TCP en el receptor de la comunicación, por lo que solicitará una retransmisión al emisor que debe encargarse de reenviar nuevamente el segmento perdido completo, en lugar de la célula errónea. Este mecanismo acaba degenerando el rendimiento de la red con un considerable desaprovechamiento del goodput de la misma. Como ATM no dispone de control de acceso al medio (MAC) el throughput cae considerablemente si la red comienza a experimentar congestiones.

TCP es un protocolo de transporte orientado a conexión que basa su fiabilidad en retransmisiones extremo-extremo cuando aparecen problemas. En este caso hay coincidencias con ATM, y por esto consideramos que nuestra propuesta de GoS acaba aportando ventajas no sólo al tráfico ATM nativo, sino también al predominante TCP. Para demostrar nuestra tesis vamos a presentar una serie de simulaciones que demuestran el comportamiento de TCP cuando se enfrenta a las congestiones. Para ello vamos a usar el simulador NS (Network Simulator) con el que estudiamos el mecanismo de ventana de TCP fijándonos en los efectos del umbral, de la ventana de congestión, de la probabilidad de pérdidas de segmentos, de los retardos, y en las consecuencias que todo esto acaba teniendo sobre el throughput. Veremos cómo el goodput acaba degenerándose cuando aparecen congestiones en los escenarios simulados. Si introducimos TAP en la red conseguiremos mejorar sustancialmente el goodput de las transmisiones TCP ya que logramos eliminar las retransmisiones e-e y además las labores de autoajuste de las ventanas de congestión en las fuentes TCP.

En primer lugar comentamos las características generales de TCP, para pasar después a las simulaciones con NS (Network Simulator) [3] en varios escenarios. La siguiente sección propone la inclusión de TAP en la red en un escenario IPoATM. Terminamos el capítulo con un apartado de conclusiones de las motivaciones generales de nuestro trabajo que se propone mantener el throughput en TCP sobre ATM considerando los aspectos de justicia, de ahorro del RTT, de la fragmentación de paquetes, y de la implosión, todo ello apoyándonos en el SMA-TAP.

8.2. FUNCIONAMIENTO DE TCP

El protocolo TCP es en la actualidad un conjunto de algoritmos que envían paquetes a la red sin ningún tipo de reserva previa, pero que son capaces de reaccionar ante determinados eventos en la misma. Entre esos algoritmos destaca el *Control de Congestión* y el de *Recuperación de Segmentos Perdidos*. Para poder llevar esto a cabo, cada emisor mantiene dos ventanas: RWND (*Receiver Window*) y CWND[1] (*Congestion Window*). Cuando se envían paquetes se usa la ventana más pequeña de estas dos.

El mecanismo de control de congestión en TCP tiene dos fases diferentes: *Slow Start* y *Congestion Avoidance*. Al iniciar una conexión, o al reiniciarse por el envío de un segmento perdido, el tamaño de la ventana de congestión es puesta a 1 paquete, y después es aumentada al doble en cada ACK recibido desde el receptor en el tiempo RTT. El mecanismo *Congestion Avoidance* se encarga de la violación que puede producirse cuando el tiempo de retransmisión es demasiado corto. Durante esta fase, CWND es incrementada linealmente al contrario de su crecimiento exponencial durante la fase *Slow Start*.

Se emplea un tercer parámetro que es el THRESHOLD, usado para arrancar la fase de congestión y que es inicializado a 64 Kb. Cuando aparece congestión, (por ejemplo, detectada por un timeout) el valor de THRESHOLD es puesto a la mitad del valor actual de la ventana CWND, la ventana CWND es puesta a 1 y la fase de *Slow Start* es reiniciada de nuevo.

La implementación original de TCP debida a Jacobson [4] ha dado lugar a diversas variantes que son conocidas como distribuciones Tahoe y Reno. Posteriormente han aparecido versiones más actualizadas y cuidadas de la implementación de TCP como TCP Vegas [5] o Selective Acknowledgement. En general, los algoritmos de control de congestión de TCP se han preocupado [6] por diversos aspectos concretos como: rendimiento, escalabilidad, justicia, complejidad y compatibilidad con la tecnología actual.

Por tanto, las fuentes TCP determinan la velocidad de envío de datos usando una ventana de control de flujo. El emisor envía una ventana de paquetes por cada RTT. TCP ajusta su tamaño de ventana para reflejar las condiciones de la red según lo siguiente: 1) cada vez que TCP envía una ventana de paquetes, éste incrementa el tamaño de la ventana en un paquete; 2) cada vez que la red tira un paquete, TCP reduce el tamaño de la ventana a la mitad.

TCP puede detectar la pérdida de paquetes rápidamente usando "*fast retransmit*" tan pronto como la ventana sea mayor de tres paquetes. Si falla el "*fast retransmit*" TCP cae en una retransmisión conservadora basada en timeout de un segundo o más. Así, la pérdida afecta al retardo de dos formas: 1) decrementando el tamaño de la ventana de TCP y la velocidad del envío; y 2) forzando a caer a TCP en *timeouts*. La referencia [7] muestra resultados del efecto de la perdida de paquetes en los retardos de los paquetes provocadas por estas cuestiones, de forma que puede verse claramente cómo, a medida que se incrementa la pérdida de paquetes, la eficiencia de TCP cae sustancialmente. Puede comprobarse cómo, también en el caso de elevadas velocidades de transmisión, se acaban generando comportamientos erráticos e injustos en el retardo que no se ajustan a los valores medios deseables.

[1] El control de flujo en TCP se basa en el tamaño de CWND en cada envío de segmentos. Las cabeceras de cada segmento disponen de un campo de 16 bits que indica el tamaño de ventana y lo limitan a un máximo de 65.535 bytes.

Los siguientes son los aspectos básicos de funcionamiento de TCP:

- Segmento: designa de forma genérica los paquetes TCP/IP, tanto los de datos como los ACK. Generalmente las unidades de transferencia de IP se designan con el nombre de datagramas, mientras las del protocolo TCP se conocen como segmentos.
- CWND: representa la ventana de congestión, y no es otra cosa que una variable que limita la cantidad de datos que TCP puede enviar. Su tamaño varía dependiendo de las condiciones de la red, de forma que si la red no descarta paquetes por congestión, el tamaño de esta ventana aumenta permitiendo aumentar también la velocidad de transmisión de las fuentes de tráfico.
- INITIAL_WINDOW: es el valor con que se inicia la ventana de congestión CWND.
- SMSS: expresa la cantidad máxima de datos que una fuente de tráfico TCP puede enviar.
- RWND: es la cantidad máxima de datos que puede recibir un receptor de tráfico TCP.
- RTT: es el *round trip time*, es decir el tiempo que transcurre desde que un segmento sale del emisor hasta que éste recibe la confirmación de que ha sido recibido por el receptor. En realidad, el RTT determina la velocidad de transmisión de TCP, ya que el emisor envía cada RTT el tamaño de datos determinado por la ventana CWND.
- CURRENT_WINDOW: representa la cantidad de información que envía el emisor cada RTT. Esta ventana toma como valor el más pequeño de CWND o RMSS.
- SSTHRESH: es una variable que se usa para determinar qué algoritmo de control de congestión se debe usar. Como ya se ha indicado, estos dos algoritmos son *Slow Start* y *Congestion Avoidance*. Cuando CWND < SSTHRESH se emplea el algoritmo *Slow Start*, mientras que cuando CWND ≥ SSTHRESH se aplica el control de congestión determinado por *Congestion Avoidance*.

Por tanto, una fuente TCP establece la cantidad de datos que envía usando una ventana (Window Flow Control) y envía una ventana de segmentos por cada RTT. TCP ajusta el tamaño de dicha ventana dependiendo de las condiciones de la red. Así, el tamaño de CWND se incrementa en el doble de segmentos por cada ACK recibido si estamos en *Slow Start*, y se incrementa en 1/CWND por cada ACK recibido en *Congestion Avoidance*. Todo esto ocurre en una conexión en la que no hay descartes de paquetes, y lo ilustramos con la *Figura 8.1*, donde podemos observar la ventana de congestión sin pérdidas que acabamos de comentar. La ventana de congestión aumenta exponencialmente mientras su tamaño es menor que SSTHRESH, esto es debido a que se está usando el algoritmo *Slow Start* aumentando progresivamente el número de segmentos (1, 2, 4, 8, 16...) a medida que se van recibiendo los ACK. Pero cuando el tamaño de CWND se iguala al valor de SSTHRESH entra en acción el control de congestión de *Congestion Avoidance*. A partir de este momento la ventana incrementa en 1/CWND por cada ACK, dando lugar a un crecimiento lineal de la ventana CWND. Podría decirse que el algoritmo *Slow Start* es usado por TCP para probar la capacidad de la red (de la que se desconoce su capacidad) y la cantidad de segmentos que puede soportar sin congestionarse. En cuanto se acerca una posible congestión, TCP cede control al algoritmo *Congestion Avoidance* que permite el incremento lineal de CWND hasta que la congestión es detectada.

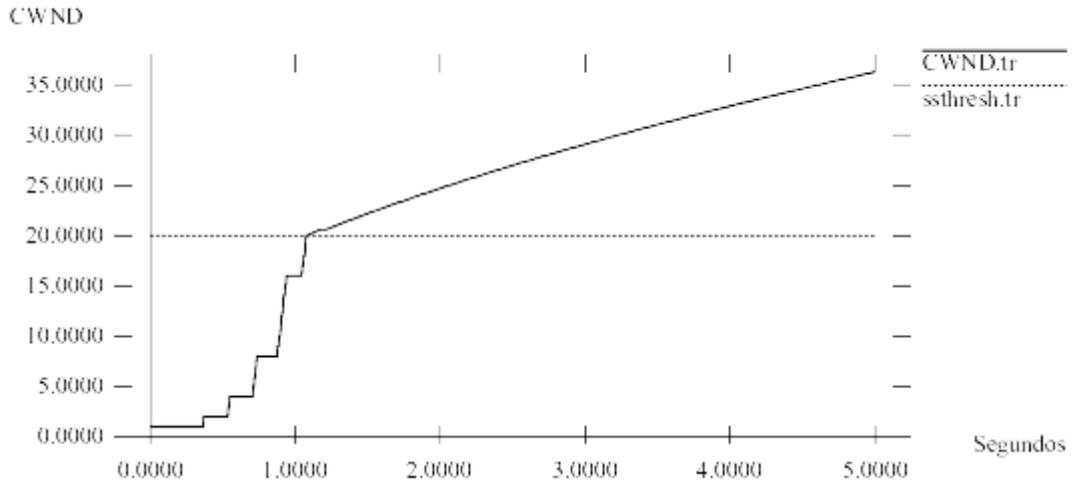

Figura 8.1. *Evolución de la ventana de congestión sin pérdidas*

Comentario aparte merecen las novedosas investigaciones aportadas por [8] donde se demuestra la naturaleza caótica del control de congestión de TCP en función de aplicaciones concretas de las redes TCP/IP. Del mismo modo es también destacable la aportación al buen funcionamiento de TCP del concepto de "*pacing*" o espaciado de los datos aportados por [9] a las fuentes TCP cuyos mecanismos de control de congestión acaban degenerando en tráfico a ráfagas que provocan la pérdida general de eficiencia de la red.

8.2.1. REACCIÓN ANTE CONGESTIONES

Cuando la capacidad de transmisión de la red es inferior a la cantidad de información que desea transmitirse, la propia red comenzará a descartar segmentos en un intento porque los emisores disminuyan la cantidad de información que generan. TCP detecta que un segmento ha sido descartado por congestión si el número de ACK repetidos recibidos es 3, o cuando el timeout de las retransmisiones ha expirado. TCP reaccionará reinicializando el valor de SSTHRESH a la mitad de la ventana, reduce CWND al valor determinado por INITIAL_WINDOW, reduciendo así la cantidad de segmentos que envía a través de la red. En suma, se opera según el siguiente algoritmo:

- Inicialización de CWND a un segmento, y del umbral SSTHRESH a 65.535 bytes.
- La rutina de salida de TCP nunca envía más datos del menor de los valores CWND y RMSS.
- Cuando se produce la congestión (timeout o ACK duplicados) SSTHRESH es reducido y toma el valor mínimo $min(\frac{CurrentWindow}{2}, 2)$ y CWND se inicializa a INITIAL_WINDOW.
- Al recibir un nuevo ACK en el emisor, se incrementa CWND dependiendo del algoritmo de control de congestión que se esté usando en ese instante y según lo ya comentado.

La *Figura 8.2* representa la situación comentada y podemos observar la evolución de la ventana de congestión con pérdidas en la red debidas a congestiones.

Tanto la *Figura 8.1* como la *Figura 8.2* se han obtenido con la simulación de la misma topología sobre NS. La topología es presentada en la *Figura 8.3*, donde pueden observarse los 7 nodos que intervienen. Cada uno de los enlaces tiene un ancho de banda de 1 Mbps, un delay de 10 ms. y usan DropTail como tipo de cola. El escenario de simulación está compuesto por los tres agentes siguientes:

- tcp (Agent/TCP) actúa como el emisor de la conexión y está asociado al nodo 0. Es el agente a monitorizar y del que se obtienen los valores de CWND y SSTHRESH.
- tcpSink (Agent/TCP) es el agente que actúa como receptor de la conexión, que envía los ACK al agente tcp. Este agente está asociado al nodo 6.
- ap (Application/Traffic/CBR), es el agente que realmente genera el tráfico, y está asociado al agente tcp (nodo 0).

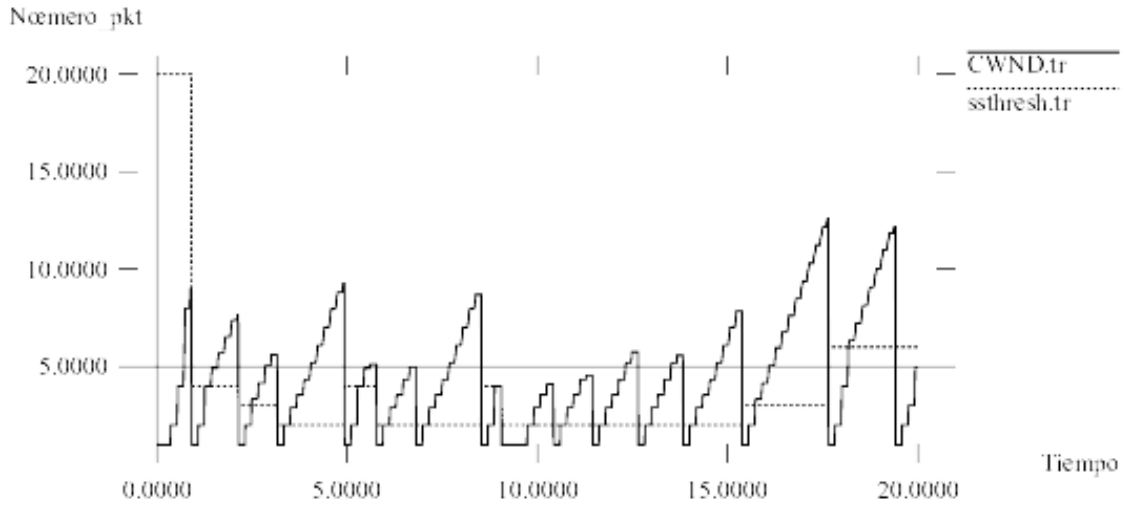

Figura 8.2. *Evolución de CWND y de SSTHRESH en una red congestionada*

La diferencia de los dos escenarios simulados está en el hecho que mientras la *Figura 8.1* muestra el comportamiento de TCP sin pérdidas de segmentos, la *Figura 8.2* introduce la pérdida de paquetes TCP. Para ello, en el segundo caso se ha introducido la probabilidad de pérdida mediante modelos de error que nos han permitido definir una probabilidad de pérdida de 0,02 en los enlaces 2-3 y 3-4. En la *Figura 8.3* puede observarse cómo el nodo 3 ha perdido un paquete que ha sido descartado por la red.

En el caso de la *Figura 8.2* se ha empleado un tiempo de simulación superior (20 s.) para poder comprobar con claridad el efecto de las pérdidas sobre la ventana de congestión que es reducida a 1 en múltiples ocasiones para solventar el problema de las congestiones. Destacamos que nuestra propuesta TAP intenta solventar estos problemas de pérdidas que afectan, tanto a la reducción de la ventana, como a la posterior retransmisión de las pérdidas extremo-a-extremo. Así, la fuente no se verá obligada a reducir su velocidad de envío tan a menudo como muestra la *Figura 8.2* y, sobre todo, cuando aparezcan las congestiones, éstas serán resueltas localmente entre los nodos afectados, siempre que sean nodos AcTMs.

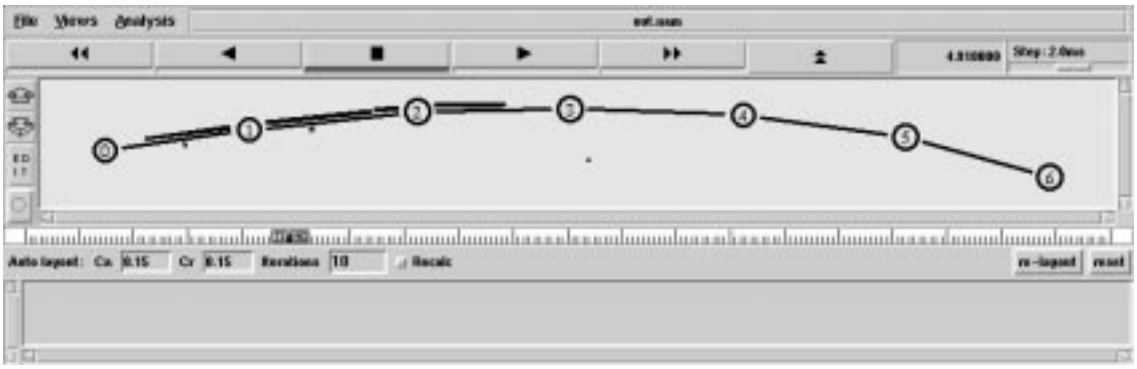

***Figura 8.3**. Topología de simulación NS para las Figuras 8.1 y 8.2*

8.2.2. THROUGHPUT Y PROBABILIDAD DE PÉRDIDAS

Puede estimarse el comportamiento de TCP considerando que el enlace sobre el que se usa cumple las siguientes características:

- El valor de *RTT* es constante.
- El ancho de banda del enlace es más que suficiente para soportar la conexión TCP.
- Existe una probabilidad de pérdidas en la red que es aleatoria y constante y la representamos por *P*. Suponemos por tanto que el emisor es capaz de enviar *1/P* paquetes consecutivos antes de que un segmento TCP sea descartado provocando la congestión y el recorte de CWND justo a la mitad de su tamaño actual.

Podemos destacar que la ventana de congestión CWND dibuja el gráfico periódico presentado en la *Figura 8.4* [10].

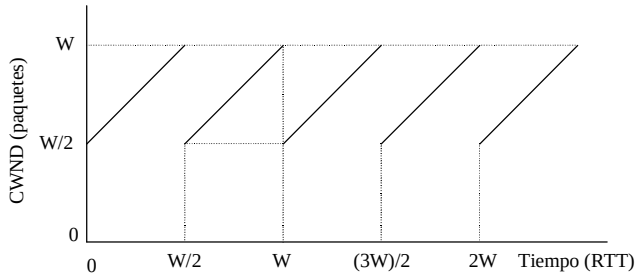

Figura 8.4. Evolución de la ventana CWND de TCP con pérdidas periódicas

A la vista de la *Figura 8.4*, y suponiendo que el tamaño máximo de la ventana CWND es de *W* segmentos, entonces según la definición que hemos hecho del algoritmo *Congestion Avoidance*, el valor mínimo de CWND es de *W/2* segmentos. Como sabemos que el receptor devuelve un ACK por cada segmento recibido, entonces CWND se incrementa en un segmento por cada RTT, o bien cada *(RTT*(W/2))*

segundos. A partir de esto se deduce que el total de datos entregados a la red en cada ciclo es precisamente el área de la superficie situada por debajo del diente de sierra perfecto que se observa en la *Figura 8.4*. Así, este volumen de datos o número de segmentos TCP entregados en cada ciclo puede ser calculado por la expresión

$$(\tfrac{W}{2})^2 + \tfrac{1}{2}(\tfrac{W}{2})^2 = \tfrac{3}{8}W^2 \tag{1}$$

Supóngase ahora una red sin pérdidas con un RTT constante porque tiene suficiente ancho de banda y con una baja carga total que nunca llene las colas. Pues bien, según [10] puede aproximarse la pérdida de paquetes aleatoria mediante una probabilidad constante *P* que asuma que el enlace entrega aproximadamente *1/P* paquetes consecutivos seguidos de un descarte, todo esto sin considerar los datos que se transmiten durante las recuperaciones de paquetes. Tenemos por tanto dos aproximaciones a la entrega de paquetes, la expresión (1) y *1/P*, por lo que uniendo ambas y despejando *W* obtenemos,

$$W = \sqrt{\tfrac{8}{3p}} \tag{2}$$

Por otro lado, podemos aplicar estos datos conocidos sobre la siguiente expresión que calcula el ancho de banda (donde *MSS* es el máximo tamaño de segmento TCP) transmitido,

$$AB = \frac{datos/ciclo}{tiempo/ciclo} = \frac{MSS * \frac{3}{8}W^2}{RTT * \frac{W}{2}} = \frac{MSS/P}{RTT * \sqrt{\frac{2}{3P}}} \tag{3}$$

Se puede reestructurar la expresión (3) agrupando el término constante $K = \sqrt{\frac{3}{2}}$; llegando a,

$$AB = \frac{MSS}{RTT}\frac{K}{\sqrt{P}} \tag{4}$$

La fórmula (4) expresa el ancho de banda de la red y nos sirve así para aproximar de forma sencilla el funcionamiento de TCP tras haber realizado algunas simplicaciones como las explicadas. El artículo [10] presenta otras referencias con diversas aproximaciones para el valor de la constante *K* que, indiferentemente de su valor, puede servir para considerar que su valor es siempre menor que 1, con lo cual podemos quedarnos finalmente con la siguiente fórmula (5) como una buena expresión del throughput de TCP,

$$AB < \left(\frac{MSS}{RTT}\right)\frac{1}{\sqrt{P}} \tag{5}$$

Por nuestra parte hemos analizado también el comportamiento de TCP en diversas situaciones y queremos plantear el comportamiento y efecto del throughput estudiado con respecto a la probabilidad de pérdida de paquetes. Para ello reorganizamos la formulación (5) del algoritmo *Congestion Avoidance* de TCP que expresa el comportamiento "*steady state*" de TCP bajo condiciones ligeras de carga y para una pérdida de paquetes moderada en su expresión general,

$$TH = \frac{MSS}{RTT\sqrt{P}} \tag{6}$$

Si en la ecuación (6) *TH* representa el throughput y consideramos como constantes los valores de *MSS* y *RTT*, podemos obtener la fórmula (7) donde comprobamos cómo el throughput es inversamente proporcional a la probabilidad de pérdida de paquetes,

$$TH = \frac{K}{\sqrt{P}} \tag{7}$$

La expresión (7) nos permite estudiar, por tanto, el comportamiento que va a experimentar el throughput a medida que se produzcan pérdidas de paquetes en routers congestionados. Dando valores a la *P* podemos obtener la representación gráfica de la *Figura 8.5* donde se observa una pendiente logarítmica negativa que indida la caida de la efectividad de la red a medida que se pierden segmentos TCP.

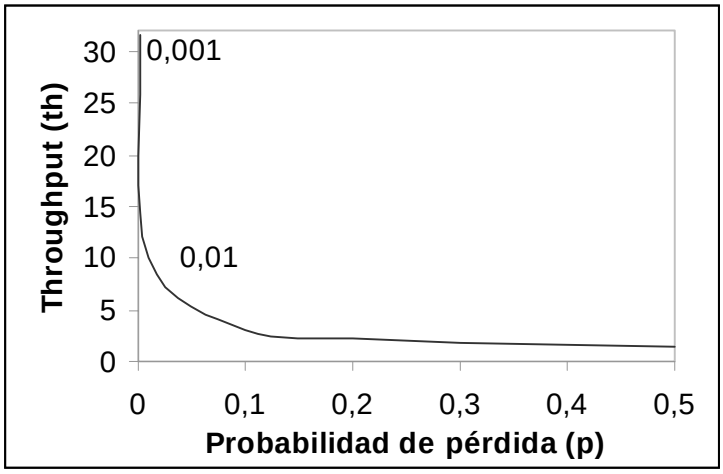

Figura 8.5. *Efecto de la probabilidad de pérdida sobre el throughput*

Como podemos observar en la *Figura 8.5*, entre una probabilidad de pérdida de 10^{-3} y 10^{-1} , la degradación del comportamiento de la red cae logarítmicamente, mientras a partir de valores estables de pérdidas de paquetes el comportamiento de la red es prácticamente constante. El problema lo encontramos en el hecho que TCP dobla los intervalos de los tiempos de retransmisión entre pérdidas sucesivas de paquetes. Es decir, si en la ecuación (6) no consideramos constante el valor de *RTT* nos encontraremos con que su efecto sobre la *Figura 8.5* será aun más negativo para la eficiencia de la red.

De la formula (7) puede obtenerse una nueva función que aproxima el tamaño de la ventana *W* que emplea TCP cuando se tiene una velocidad media de pérdidas *P*. Así, ajustando el valor de la constante *K* de la fórmula (4), obtenemos la siguiente expresión que es resultado de la adaptación de (7) en [10] y de las propuestas realizadas en [11],

$$W = \frac{0{,}866}{\sqrt{P}} \tag{8}$$

En nuestro caso particular, para el valor de la constante *K* calculamos $\sqrt{3/4}$ suponiendo que se aplica la estrategia retardada en los ACKs en lugar de aplicarla sobre cada uno de los paquetes, y según el planteamiento de pérdidas periódicas, en lugar de partir de un plantemiento de pérdidas aleatorias.

Así, y en línea con [7], podemos comprobar que la ecuación (8) puede verse desde dos puntos de vista diferentes. En primer lugar, la red descarta los paquetes a una velocidad independiente de la actividad del emisor, por lo que la fórmula expresa entonces la forma en que el emisor es capaz de reaccionar. En segundo lugar, si consideramos que la red es capaz de almacenar sólo un número determinado de paquetes, la ecuación puede entenderse como la velocidad que la red debe imponer para que la ventana de TCP quepa en esa capacidad de almacenamiento de la red. Según todo esto, podemos reorganizar la fórmula (8) y obtendremos la velocidad media de pérdidas *P* según la siguiente expresión,

$$P = \frac{0{,}75}{W^2} \tag{9}$$

La ecuación (8) puede entenderse como si la red descartase un porcentaje de segmentos independientemente de las acciones que realice la fuente. Es decir, describe la forma en que va a reaccionar el emisor. Para estudiar este comportamiento hemos simulado con NS el escenario de la *Figura 8.6*. Se han empleado enlaces de 2 Mbps de ancho de banda, con retardos de 10 ms. y la cola es *DropTail*. Además, se ha asociado a cada enlace una probabilidad de pérdida inicial de 0,001 que es incrementada el 5% cada 5 segundos en todos los enlaces.

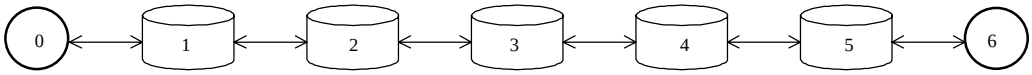

Figura 8.6. *Escenario para estudiar el funcionamiento de TCP*

El objetivo de este escenario es estudiar el comportamiento del ancho de banda con respecto a la probabilidad *P* de errores. Como resultado hemos obtenido el gráfico de la *Figura 8.7* en el que nuevamente podemos comprobar cómo se cumple el comportamiento macroscópico que indicaba la intuitiva *Figura 8.1*. Podemos ver cómo en la *Figura 8.7* la probabilidad de pérdida *P* acaba determinando el ancho de banda de la fuente TCP, como también lo indica intuitivamente la fórmula (8) que obtuvimos anteriormente. A medida que aumentamos la probabilidad de pérdida disminuye el ancho de banda logarítmicamente hasta acercarse a una evolución lineal cuando la probabilidad de pérdidas desciende, que es lo que intenta TAP.

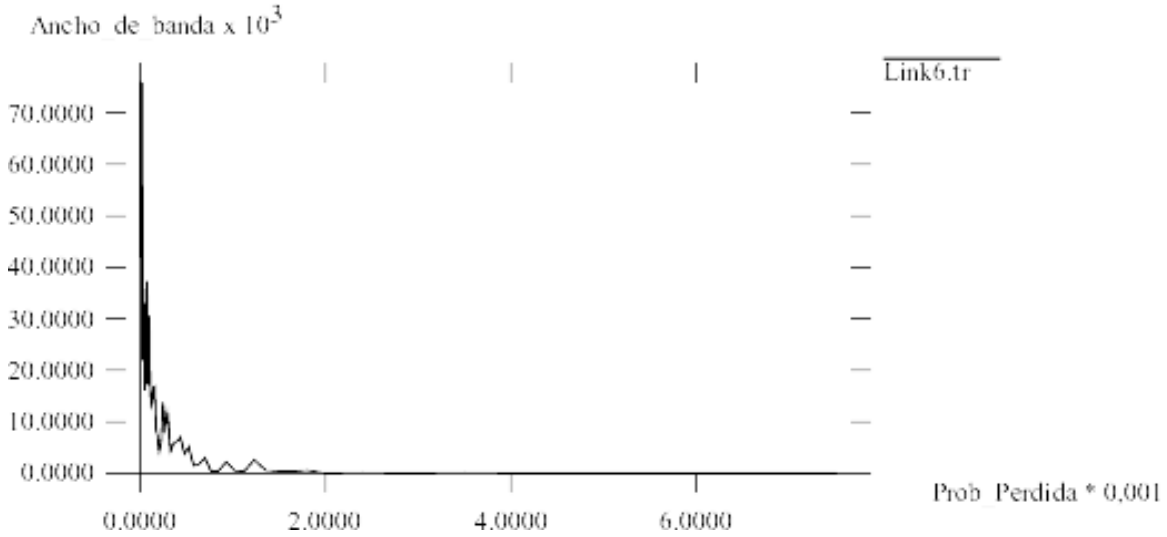

Figura 8.7. *Simulación de la ecuación (4) con NS*

8.2.3. ADAPTACIÓN DEL ANCHO DE BANDA DEL EMISOR

Una consideración importante a tener en cuenta en el caso de TCP es que el ancho de banda disponible para el emisor es adaptado por la red de acuerdo a su propia capacidad. Si consideramos que la red dispone de una capacidad limitada para la transferencia de un número concreto de paquetes, la ecuación (8) puede expresar la probabilidad de pérdida que la red debe imponer para que el emisor del tráfico TCP se adapte a la capacidad de transferencia de la red en cada momento. Así la ecuación (8) la hemos reestructurado para entender este planteamiento, dando lugar a la fórmula (9) vista anteriormente.

Hemos analizado una aplicación de la formula (9) consistente en un emisor TCP enviando tráfico a través de un router que dispone de una cola de longitud máxima de 8 paquetes. El router acaba generando una situación de congestión, por lo que la ventana CWND del emisor debe variar entre 8 y 4 paquetes con una media de 6 paquetes. Para lograr esta situación con la ecuación (9), según [7,10], el router debe descartar el 2,1% de los paquetes del emisor TCP. Este es el modo en el que la red usa la probabilidad de pérdida *P* para indicar a TCP el tamaño correcto de la ventana de congestión. Este estudio lo realizamos mediante la simulación del escenario presentado en la *Figura 8.8*.

Este escenario consta de una topología con 4 nodos, donde los enlaces 0-1 y 1-3 tienen anchos de banda de 5 Mb, un delay de 20 ms. y usan cola *DropTail*. El enlace 1-2 dispone de un ancho de banda de 0,3 Mb, con un delay de 100 ms. y también cola *DropTail*.

Para este escenario usamos los siguientes agentes:

- tcp (Agent/TCP) que actúa como el emisor de la conexión y se asocia al nodo 0.
- sink (Agent/TCP) actúa como receptor de la conexión enviando ACKs al agente tcp. Está asociado al nodo 3.
- ftp (Application/FTP) que es el que realmente genera el tráfico. Está asociado al agente tcp.

Hemos simulado el escenario (mostrado en la *Figura 8.8*) para intentar estudiar la forma en que la red adapta el ancho de banda del emisor modificando los paquetes descartados. Hemos realizado un total de cinco simulaciones del escenario variando algunos de los parámetros según lo siguiente:

- Tres simulaciones se han usado para variar el tamaño de la cola del nodo 2. De este modo se obtienen tres gráficos con la longitud de la cola a 16, 8 y 4 paquetes.
- Otras dos simulaciones en las que se mantiene fijo el tamaño de la cola a 8 paquetes, pero variando el ancho de banda del enlace 1-2 en primer lugar a 0,2 Mbps y después a 0,5 Mbps.

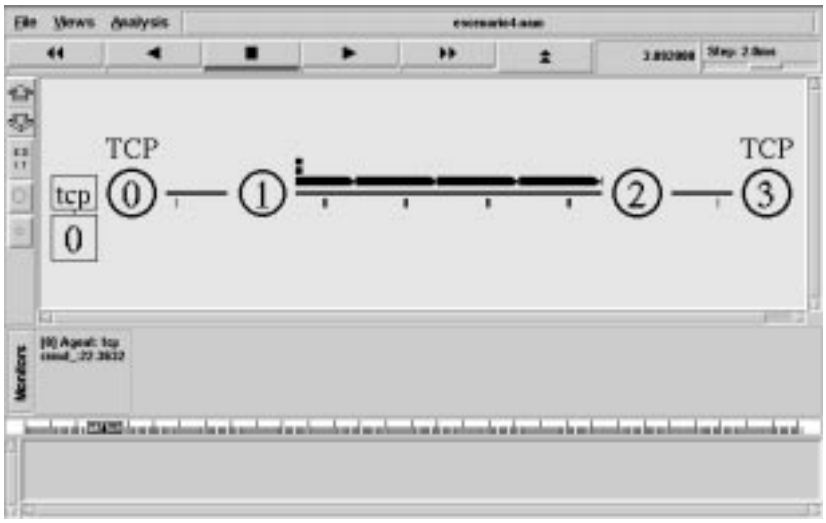

Figura 8.8. *Escenario NS para el estudio del comportamiento del tamaño de las colas y del ancho de banda*[2]

En todas las simulaciones el objetivo es analizar la evolución de la ventana de congestión en el agente emisor TCP comentado anteriormente. A continuación se comentan los resultados obtenidos en las cinco simulaciones realizadas con NS.

De la simulación del anterior escenario hemos obtenido con NS la *Figura 8.9*, donde puede observarse la evolución de CWND dependiendo del tamaño de la cola que se emplee en el router. Si la longitud de la cola es suficientemente grande (L=16) la red no necesita descartar ningún paquete en los 30 segundos estudiados, por lo que la ventana CWND evoluciona con un crecimiento lineal a partir del segundo 2, en que se alcanza el valor del umbral SSHTRESH y empieza a actuar el mecanismo *Congestión Avoidance*.

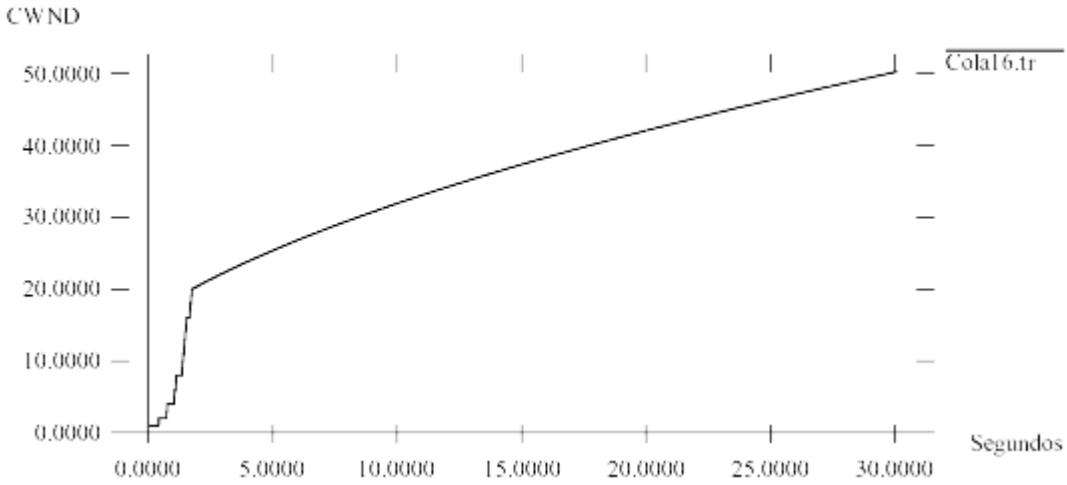

Figura 8.9. *Evolución de la ventana CWND con una cola de 16 paquetes*

[2] Puede observarse el crecimiento de la cola del nodo 1, así como los segmentos y sus correspondientes ACK

Sin embargo, observando la *Figura 8.10* donde se emplea una cola de 8 paquetes podemos comprobar cómo la red obliga al emisor TCP a ajustar su ventana de congestión a la nueva capacidad de la cola. Como puede observarse en la *Figura 8.10* y en la *Figura 8.11*, se ha seguido disminuyendo el tamaño de la cola del buffer de forma progresiva (L=8 y L=4), lo que acaba provocando el descarte de paquetes y acarrea el descenso del throughput en los enlaces. Observemos cómo a medida que decrementamos el tamaño de las colas, también se produce el descenso de la ventana CWND representada en el eje Y.

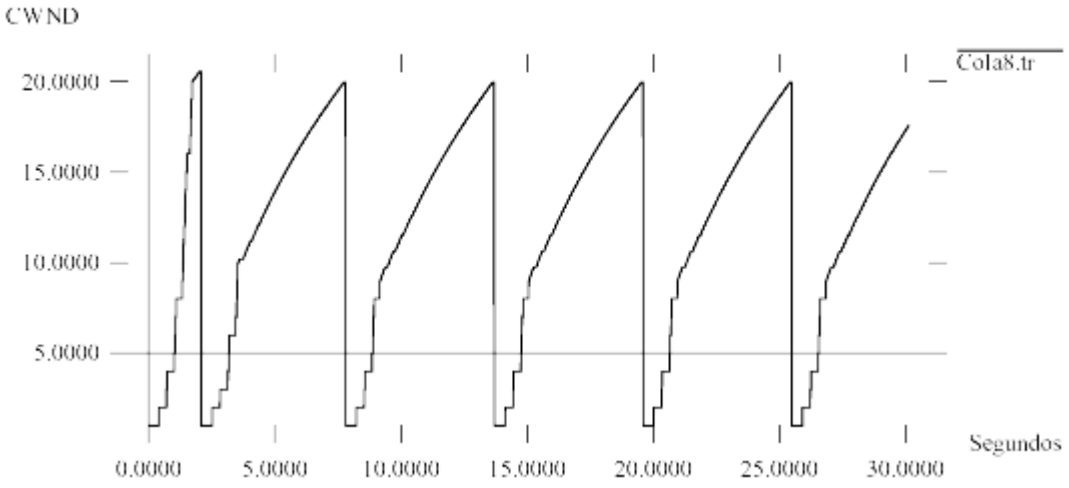

Figura 8.10. *Evolución de la ventana CWND con una cola de 8 paquetes*

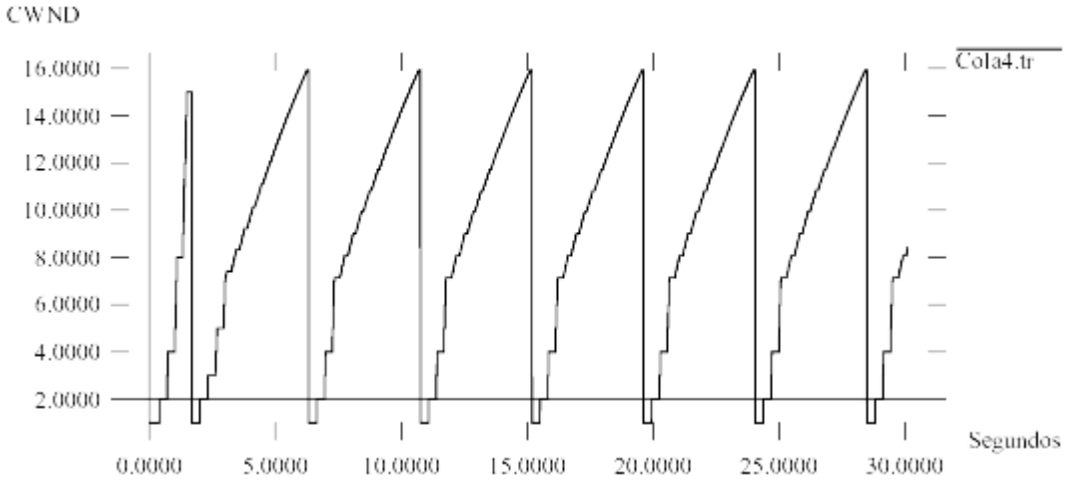

Figura 8.11*. Evolución de la ventana CWND con una cola de 4 paquetes*

En las tres simulaciones presentadas anteriormente no se han tenido en cuenta los paquetes almacenados "*in flight*"[3], porque hemos considerado que la capacidad de almacenamiento de la red está determinada solamente por la capacidad de almacenamiento del router. Sin embargo, aclaramos que la capacidad de almacenamiento de la red es la suma de la longitud de la cola del router más los paquetes almacenados "*in flight*" en el enlace, lo que está determinado por el ancho de banda de cada enlace. Las *Figuras 8.12* y *8.13* demuestran esta situación, en las que podemos observar cómo al disminuir el ancho de banda de un enlace de 0,5 Mb a 0,2 Mb, disminuye también la capacidad de almacenamiento de la red y, por tanto, también desciende el throughput en la misma. Podemos observar cómo al incrementar el ancho de banda de la red a

[3] La capacidad de almacenamiento real de la red es la suma de los paquetes contenidos en las colas de los routers más todos aquellos que en cada momento se encuentran viajando por los enlaces entre nodos.

0,5 Mbps en la *Figura 8.13* se consigue evitar que, en el tiempo de simulación de 30 segundos, la ventana de congestión de TCP no sea puesta a 1 en cinco ocasiones como ocurre en la *Figura 8.12*. Al incrementar la capacidad del enlace, la ventana CWND no es reducida a 1, sino que crece linealmente según el control de *Congestion Avoidance*.

Destacamos que si se aumenta el tiempo de simulación de la *Figura 8.12*, la congestión acabará apareciendo, por las propias características de TCP[4]. Lo que se consigue aumentando el ancho de banda del enlace es retrasar el momento de aparición de las congestiones, mejorando el goodput por tanto.

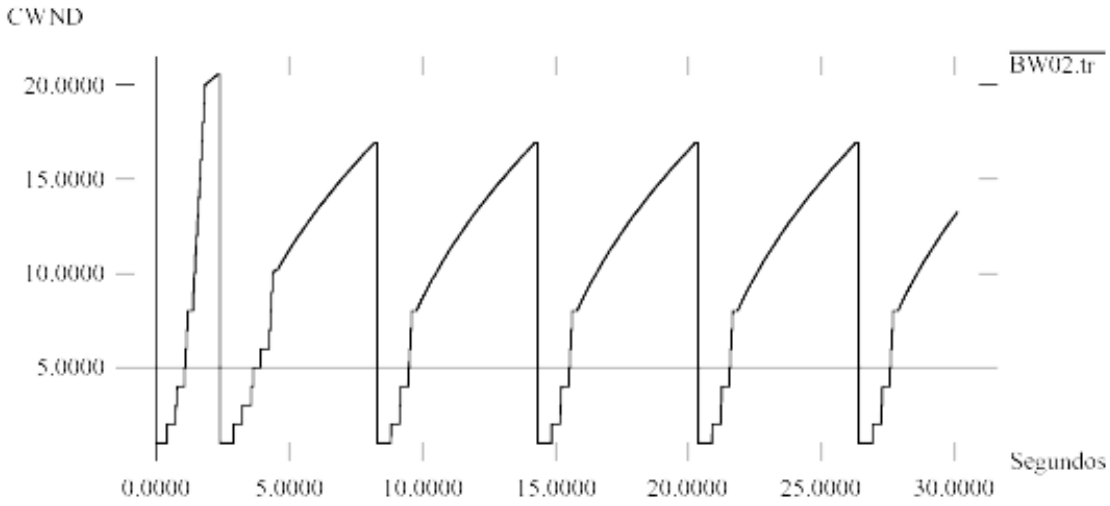

***Figura 8.12**. Evolución de CWND sobre un enlace de 0,2 Mbps*

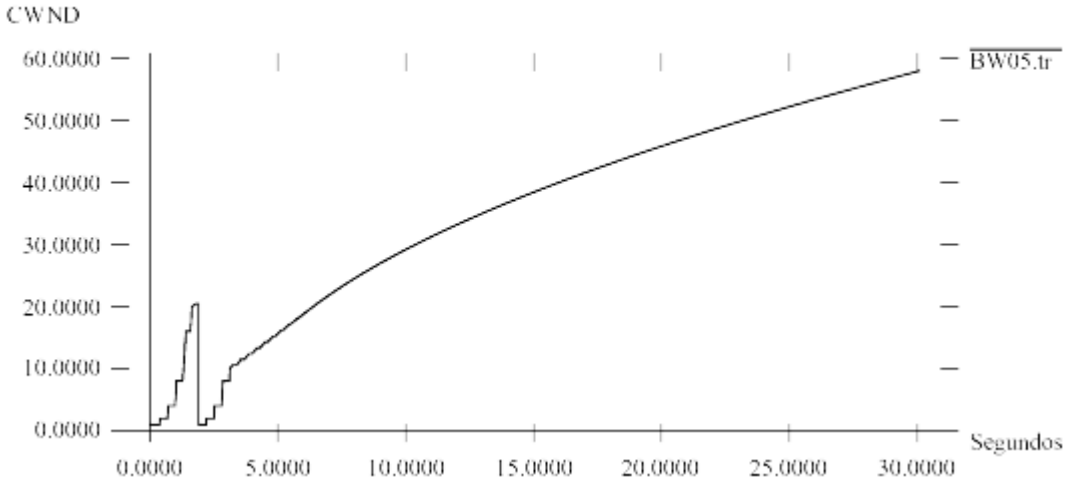

***Figura 8.13**. Evolución de CWND sobre un enlace de 0,5 Mbps*

Si deseamos considerar el efecto de *N* fuentes TCP compartiendo un enlace, debemos de partir de la posibilidad que ese enlace pueda convertirse en un posible cuello de botella determinado por el tamaño de la cola del router que podemos suponer con valor *B*. Los tamaños de las colas TCP deben sumar el tamaño *B*,

[4] Aunque una vez iniciado el algoritmo *Control Avoidance* el crecimiento de la ventana CWND es lineal, llegará un momento en que se supera la capacidad de algún enlace de la red, o bien la capacidad de alguno de los routers que la forman, por lo que CWND será reducido a 1 en el momento que aparezca la congestión.

de forma que *W=B/N*. Sustituyendo estos valores en la ecuación (9) podemos calcular una nueva aproximación a la predicción de probabilidad de pérdida de paquetes:

$$P = 0{,}75\frac{N^2}{B^2} \tag{10}$$

Puede entenderse por la fórmula anterior que cargas elevadas de la red acabarán generando pérdidas elevadas de paquetes, donde además influye directamente el número de fuentes de tráfico que intervienen compartiendo el enlace y, por tanto, también el buffer de los routers. Además, la posibilidad de gestión de la carga de tráfico depende directamente de la capacidad de paquetes del propio buffer que se comporta como un punto de congestión para el tráfico.

Si usamos *N* fuentes, la capacidad de almacenamiento de la red será entonces la suma de todas las fuentes TCP compartiendo el ancho de banda *RTT* veces [7,11]. Por tanto, si las fuentes TCP tienen igual *RTT*, la capacidad de almacenamiento del enlace es el ancho de banda *RTT* veces. De este modo, en la ecuación (10) el valor de *B* es la capacidad de almacenamiento del enlace más los buffers de las colas de los routers.

Por último, podemos entender cómo pequeños límites en la longitud de la cola provocan que en la fórmula (10) la variable *B* se convierta en constante y provoque que la velocidad de pérdidas crezca cuando se incrementa el numero de conexiones TCP compitiendo por el enlace. Como la velocidad de pérdida crece, cada ventana de TCP se ve reducida y cada fuente TCP aminorará su velocidad de envío. De este modo la velocidad de envío *V* de cada fuente puede ser obtenida de las fórmulas (4) y (8),

$$V = \frac{0{,}866}{RTT\sqrt{P}} \tag{11}$$

Llegados a este punto hemos de destacar que la inclusión de la memoria DMTE en la arquitectura TAP contribuye de forma directa a aumentar el tamaño de buffer y, a la vista de la fórmula (10), a conseguir disminuir la probabilidad de pérdida *P* que es inversamente proporcional al tamaño del buffer de las colas en los routers. Sin embargo, nuestra aportación va más allá de la simple agregación de memoria, ya que la ganacia más importante la obtenemos en la posibilidad de realizar las recuperaciones de forma local al router (conmutador) congestionado consiguiendo un valor de *RTT* más pequeño (como se explica en la sección 8.4). TAP consigue, por tanto, disminuir la probabilidad de pérdida *P* y también el *RTT*. Podemos ver que estas dos variables afectan a la fórmula (11) que muestra cómo valores bajos de *RTT* y de *P* permiten alcanzar mayores velocidades de envío en las fuentes de tráfico.

8.3. TCP SOBRE ATM

En lo relativo a las investigaciones sobre la evaluación del rendimiento de TCP sobre ATM éstas se pueden dividir en tres grandes grupos [6]: 1) las que se fijan en el dinamismo de TCP; 2) las que atienden al comportamiento de ATM; y 3) las que prestan atención a la interacción entre las ventanas de TCP y los mecanismos de control de congestión de la capa ATM. Aunque la evaluación del rendimiento de TCP sobre ATM ha sido fuente de diversas investigaciones, las propuestas resuelven sólo problemas particulares como es la fragmentación de TCP, los requerimientos de buffers, la interacción entre los esquemas de congestión de TCP y ATM, y la degradación de TCP. En cierto modo se echan en falta propuestas generales que se enfrenten a todas o varias de estas problemáticas y en esta línea se dirigen nuestras investigaciones, aportando un SMA que busca la optimización del goodput con un adecuado tratamiento de las colas de entrada, y con una cuidada política de gestión de buffer mediante la delegación de actividades concretas en los agentes que forman parte del SMA.

La mayor parte de aplicaciones de datos no son capaces de predecir sus propias necesidades de ancho de banda, por lo que se necesita de algún servicio que permita a todos los usuarios activos de la red compartir dinámicamente el ancho de banda disponible. Sabemos que en el caso de ATM las CoS ABR y UBR son la propuesta estándar para soportar el tráfico de datos. La referencia [2] presenta el estudio de congestiones de redes TCP sobre ATM comprobando cómo el throughput de TCP cae también cuando se comienzan a descartar células en los conmutadores ATM. El bajo throughput conseguido se debe al desaprovechamiento del ancho de banda en los enlaces congestionados que transmiten células de paquetes corrompidos, es decir, paquetes en los cuales se ha tirado alguna de sus células[5]. Otras investigaciones [12] han demostrado que

[5] En realidad este trabajo estudia el efecto de la fragmentación de los segmentos TCP.

TCP sobre UBR con EPD experimenta una apreciable degradación en el funcionamiento en cuanto a la justicia, requiriendo además de un tamaño de buffer relativamente grande, incluso con pocas conexiones. Sin embargo, cuando se emplea la CoS ABR con esquemas de realimentación de velocidad explícita ofrecen a TCP mejor comportamiento en cuanto a justicia y aprovechamiento de los enlaces y, todo ello, con tamaños de buffer menores que con UBR.

La literatura [13] describe también otras formas de evitar la degradación del througput de fuentes TCP sobre UBR. Para ello lo que se hace es desactivar los descartes de células ATM durante un periodo de tiempo en el que se están produciendo congestiones. De este modo se evitan los timeouts de TCP que son la principal causa de descenso del throughput de TCP, y se acortan los periodos de congestión evitando el gran retardo experimentado con el algoritmo de retransmisión rápida de TCP antes de que el emisor reciba los ACK duplicados.

Uno de los principales problemas que experimentan las fuentes de tráfico TCP cuyos segmentos son transferidos sobre ATM es que, al usarse como indicación de congestión la propia pérdida de paquetes, esto provoca que la gestión de las congestiones se realice cuando ya es demasiado tarde. Es decir, cuando el buffer se llena, el conmutador descarta los paquetes, el receptor detecta la pérdida y éste avisa al emisor que acaba reaccionando con la retransmisión de los paquetes perdidos. Se impone por tanto la necesidad de aportar mecanismos más ágiles para la solución de las congestiones del tráfico TCP que es transferido sobre tecnología ATM.

Nos encontramos con la característica que TCP es un protocolo orientado a conexión y fiable (confirmación de segmentos entregados), mientras que ATM es también orientado a conexión pero no ofrece de forma estándar ningún mecanismo de confirmación de células entregadas. Puede hablarse así de comportamientos independientes entre ambas tecnologías. Además, cuando TCP funciona sobre ATM, el control de la red es más complejo por requerirse un mecanismo de control de congestión diferente en cada una de las capas. Existen dos diferencias básicas entre los esquemas de control de congestión de TCP y la CoS ABR de ATM: 1) El mecanismo de realimentación de ABR con células RM controla la velocidad de transmisión de las células desde el emisor (control de velocidad), mientras el mecanismo de realimentación de TCP controla el tamaño de una ventana como hemos visto en el apartado anterior (control de créditos) y 2) el mecanismo de realimentación de ABR puede ser realizado por conmutadores intermedios de la red, o por el extremo receptor del tráfico, mientras en TCP el mecanismo de realimentación es realizado sólo por el nodo destino mediante ACK extremo-a-extremo que es lo que aporta la fiabilidad a TCP.

En las fuentes TCP el tráfico máximo es controlado por la ventana CWND tal como hemos demostrado en el apartado anterior. Sin embargo, en el caso de la CoS ABR de ATM el tráfico es controlado por diversos parámetros como MCR, PCR y ACR. Son también aspectos clave para TCP sobre ATM los mecanismos de gestión de tráfico usados en los nodos extremos de TCP, en los nodos extremos de ATM y en los conmutadores de la red para, entre todos ellos, aportar el adecuado goodput para reducir el retardo causado por las retransmisiones. El retardo de procesamiento de los paquetes TCP es un periodo de tiempo aleatorio que modela la media del retardo de procesamiento de paquetes con una cierta variación de retardo. Esto se aplica, tanto a paquetes de datos, como a los de confirmación entre emisores y receptores

A la vista de todas estas características diferenciadoras y, dado que ATM es siempre un protocolo situado por debajo del protocolo de la capa de transporte TCP, se requieren soluciones para resolver los problemas de rendimiento provocados por la integración de ambas tecnologías. Estas soluciones parece lógico que estén en la línea de realizar cambios en los conmutadores ATM dentro de la red, o bien en la nueva implementación de extensiones para TCP, o también en la propuesta de protocolos especializados para los nodos que están en los límites de la red ATM con la red TCP. Destacamos que en nuestro caso TAP se enfrenta a estos problemas actuando dentro de la misma red con mecanismos hardware (conmutadores AcTMs) y también software (SMA con protocolo TAP), lo que configura toda la arquitectura TAP objeto de esta tesis.

8.4. BENEFICIOS APORTADOS POR TAP

Como ya sabemos, en nuestro caso nos basamos en EAAL-5 como extensión de AAL-5 que se diseñó específicamente para la comunicación de datos a través de ATM. En el caso de TCP sobre ATM, los datagramas IP son transmitidos en la zona del campo de datos (payload) de EAAL-5, tal como podemos intuir en la *Figura 8.14* que presenta las pilas de protocolos de una fuente emisora y otra receptora basada en TCP sobre ATM con UBR.

Por otro lado, también hemos comentado ya que nuestras propuestas van dirigidas al tráfico UBR y ABR que puede servir perfectamente para el soporte del tráfico TCP. La CoS ABR soporta aplicaciones que generan y gestionan tráfico de datos. ABR tiene la capacidad de reducir su velocidad de envío de tráfico si se

producen congestiones en la red. Por esto, podemos decir que la CoS es en realidad un mecanismo evitador de congestiones (Congestion Avoidance), y el ATM Forum ha adaptado un control de flujo para ABR conocido como *Control de Congestión basado en velocidad.* Sabemos ya que este mecanismo está basado en realimentación en la propia red usando células especiales RM. Si consideramos el esquema de velocidad explícita (ER), la fuente de tráfico envía al destinatario por el VPI/VCI una célula cada N[6] células, o bien se genera una célula RM cada T[7] unidades de tiempo. Cada célula RM contiene tres campos que aportan la realimentación a la fuente: el bit de Indicación de Congestión (CI);el bit de No Incremento (NI) y el campo de Velocidad Explícita (ER). La fuente de tráfico fija el valor del campo ER de la célula RM indicando la velocidad a la que desea enviar datos y a continuación transmite el célula FRM (Forward RM). Cuando la célula FRM es recibida por el destinatario, ésta es devuelta al emisor como célula BRM (Backward RM). Cualquiera de los campos CI, NI y ER pueden ser cambiados por los conmutadores ATM que procesan el VC del destinatario, antes de que la BRM llegue a la fuente que usará esta información para conocer el estado de la red, información que usará para ajustar su tasa de envío si se detectan congestiones. Se han propuesto varios esquemas basados en velocidad como EPRCA (Enhanced Proportional Rate Control Algorithm) o el esquema elegido por el ATM Forum ERICA (Explicit Rate Indication Congestion Avoidance).

Antes de concluir estos breves comentarios sobre el control de congestión de ATM hemos de destacar que éste es un importante aspecto para esta tecnología, por lo que se ha puesto especial interés en aspectos como: la escalabilidad, la justicia, la robustez y la facilidad de implementación del control de congestión. Aspectos éstos que hemos procurado satisfacer en nuestra propuesta.

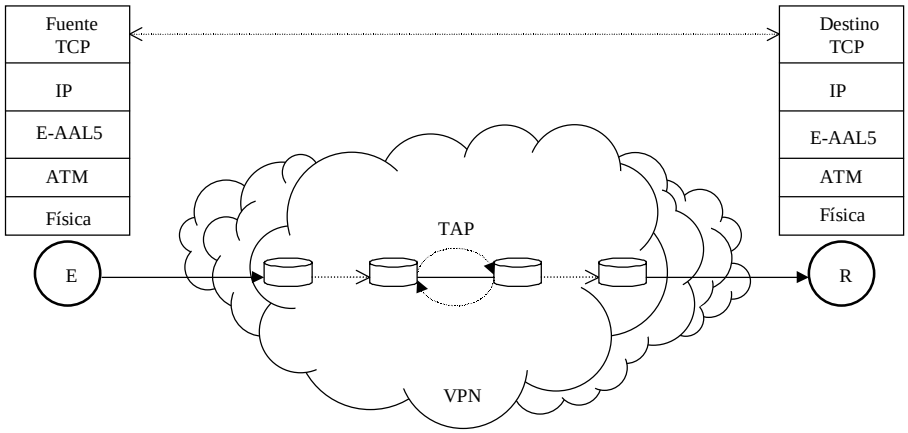

Figura 8.14. *TCP sobre ATM con TAP*

El efecto del tiempo *RTT* con respecto a la velocidad de transmisión puede comprenderse claramente observando la *Tabla 8.1* en la que se presentan diferentes portadoras ópticas, tanto de la jerarquía digital síncrona (SDH), como de la jerarquía de multiplexación SONET. La *Tabla 8.1* presenta los tiempos necesarios para la transmisión de ficheros de diversos tamaños con respecto a enlaces con diferentes velocidades de transmisión.

Nuestra intención es justificar la necesidad de mecanismos que sirvan para evitar los costes de los tiempos *RTT* en el caso de congestiones. Para ello supongamos que la VPN de la *Figura 8.14* tiene 2.000 Kms. de longitud extremo-a-extremo. Supongamos también que empleamos fibra óptica como medio físico de transmisión, en la que sabemos que se obtienen velocidades de propagación aproximadas a 2/3 la velocidad de la luz en el vacío, es decir, unos 200.000 Kms/s. En este escenario un bit tardará en recorrer la red 10 ms., por lo que, en el mejor de los casos (libre de errores y de congestiones), el *RTT* será de 20 ms.

Si estudiamos la *Tabla 8.1* con detenimiento podemos destacar que en esa misma red la transferencia de un fichero de 512 Kbytes a 155,52 Mbps supondrá un tiempo total de 3,3 ms. para llegar desde un extremo a otro de la red. Suponiendo que el fichero llegase en un solo paquete, y sin experimentar ningún problema en la transmisión, la transferencia tendrá un coste total de 23,3 ms., de los cuales 20 ms. pertenecen al coste de transferencia y al de confirmación de llegada del paquete desde el receptor, suponiendo que se trata de una fuente TCP. Por tanto, el enlace realmente está en estado de inactividad durante 20 ms., lo que supone un total del 86 % del tiempo total.

[6] El valor por defecto de N es 32. Por tanto, se genera de forma fija una célula RM de realimentación cada 32 células de datos.

[7] El valor por defecto de T es de 100 ms. Por tanto, se genera de forma constante una célula RM de realimentación cada 100 ms.

TABLA 8.1
RELACIÓN DE TIEMPOS DE TRANSMISIÓN RESPECTO A VELOCIDADES DE TRANSMISIÓN

SONET	SDH	Velocidad de transmisión	1 Mbytes	512 Kbytes	256 Kbytes	20 Kbytes
OC-1	-	51,84 Mbps	161,8 ms.	10,1 ms.	5 ms.	395 µs.
OC-3	**STM-1**	155,52 Mbps	53,9 ms.	3,3 ms.	1,6 ms.	131 µs.
OC-12	**STM-4**	622,08 Mbps	13,4 ms.	842 µs.	421 µs.	32,9 µs.
OC-24	**STM-8**	1.244,16 Mbps	6,7 ms.	421 µs.	210 µs.	16,4 µs.

La situación descrita se ve acrecentada a medida que usamos velocidades de transmisión más elevadas. Si consideramos usar un enlace OC-12 de 622,08 Mbps para la transferencia del mismo fichero de 512 Kbytes la diferencia entre el tiempo de transferencia total del fichero (0,842 ms.) y el *RTT* (20 ms.) es aún mayor que en el caso anterior, por lo que el enlace está desaprovechado el 96,3% del tiempo.

Los estudios relacionados con las transferencias realizadas a través de las redes demuestran que la mayor parte de ellas consisten en ficheros del orden de pocas decenas de Kbytes. Así, si consideramos un escenario similar al presentado en los párrafos anteriores, pero con la transferencia de un fichero de 20 Kbytes podemos comprobar a la vista de la *Tabla 8.1* que si se emplea nuevamente OC-12, el tiempo total de transferencia y de confirmación de la misma supondrá 20,032 ms. En esta situación la fuente del tráfico estará enviando tráfico durante 32,9 µs. y el resto de tiempo corresponde al *RTT*. Por esto la fuente se mantendrá en actividad el 0,16% del tiempo y el enlace quedará desaprovechado el 99,84 % del tiempo total.

Podemos comprobar que en el ejemplo propuesto en la *Figura 8.15*, a medida que se incrementan las velocidades de los enlaces, o se decrementa el tamaño de los ficheros transmitidos, el *RTT* se aproxima asintóticamente a 20 ms., mientras el porcentaje de utilización del ancho de banda del VPI/VCI usado para las transferencias se aproxima a 0.

En el escenario descrito hemos supuesto una red ideal en la que no se produce ningún problema debido a congestiones o bits erróneos. Fácilmente podemos entender que cualquier tipo de retransmisión debida a estos problemas acaba agravando aún más el problema descrito, ya que se multiplica el tiempo *RTT* mientras el tiempo de actividad de la fuente se mueve en valores cercanos a 0.

La motivación general de nuestra tesis se encuentra por tanto en encontrar soluciones para aliviar este negativo problema de retransmisiones extremo-a-extremo por lo que TAP se enfrenta a resolver las retransmisiones de forma local a donde se producen para evitar el coste del tiempo *RTT* total de la red y emplear sólo el *rtt* del enlace donde se produce la congestión. A la vez, aprovechamos los tiempos de inactividad en que se encuentran las fuentes de tráfico cuando se usan enlaces rápidos. Es decir, empleamos los tiempos de inactividad de los enlaces para realizar las retransmisiones punto-a-punto en lugar de hacerlas extremo-a-extremo.

De forma más intuitiva podemos ver también la situación descrita en la *Figura 8.15*, donde tenemos una red con 6 enlaces en la que cada uno de ellos experimenta igual retardo *d=10 ms*. En una situación de red ideal, el retardo total de la red es de *D=60 ms*, por lo que tenemos un *RTT=120 ms.* entre los dos extremos de la comunicación. Nuestro objetivo es situar el protocolo TAP entre dos enlaces de la red para conseguir amortiguar el efecto negativo de las congestiones producidas en el conmutador 3. Si suponemos que, tanto el conmutador 2 como el 3 soportan la arquitectura TAP, cuando se produzca congestión de un paquete en el conmutador 3, su retransmisión no tendrá un coste de 150 ms, sino que en realidad el coste será de 20 ms. que corresponde el retardo debido al *rtt* local del enlace que ha experimentado la congestión

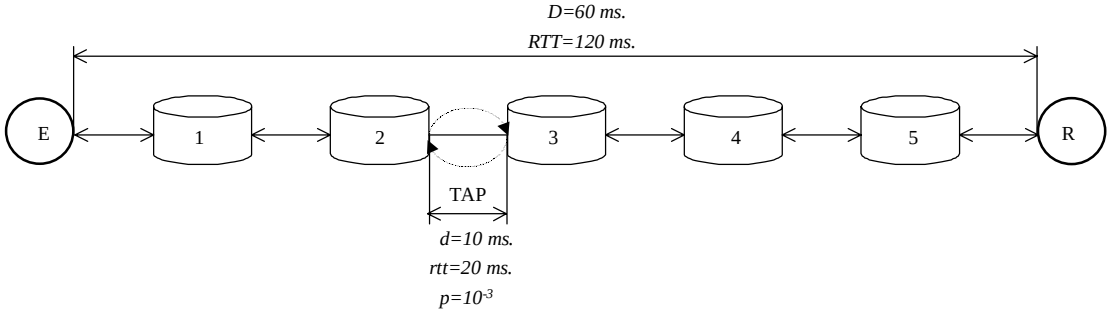

Figura 8.15. *Retransmisión local al conmutador congestionado*

Ante la *Figura 8.15* podemos retomar las fórmulas (10) y (11) donde se expresan las relaciones entre *RTT*, *B* (tamaño del buffer), *P* (probabilidad de pérdida) y *V* (velocidad de envío de las fuentes). Destacamos que en el caso ideal en que no se produzcan pérdidas de segmentos TCP en la red, la memoria DMTE de TAP aporta mayor tamaño de buffer lo que colabora a evitar las pérdidas. Por otro lado, si aparecen pérdidas, TAP las recupera en el punto donde se producen por lo que se reduce el *RTT* extremo-a-extremo al *rtt* punto-a-punto.

8.5. CONCLUSIONES

En este capítulo hemos centrado algunas de las motivaciones generales de esta tesis relacionándolas con las actuales necesidades que tiene la tecnología ATM para soportar la gran cantidad de aplicaciones (no nativas ATM) de datos existentes en las que se usa el protocolo TCP de la capa de transporte. Aunque la arquitectura TAP está pensada para soportar el tráfico ATM nativo, también puede aportar sus características intrínsecas a las transferencias de fuentes basadas en TCP.

En los protocolos de la capa de transporte como TCP sobre ATM un paquete es descartado por la red cuando se pierde una o varias células del paquete, y el nodo destino solicita la retransmisión completa del paquete corrompido o perdido. Hemos demostrado mediante simulaciones la degradación que experimenta el throughput de TCP viendo cómo éste cae logarítmicamente a medida que aumenta la probabilidad de pérdida de células ATM. Con TAP se realizan las retransmisiones de forma local por lo que se consigue decrementar la probabilidad de pérdida con el consiguiente efecto sobre el throughput de TCP que evita los retardos debidos al RTT extremo-a-extremo. Se consigue así maximizar el goodput con una mínima pérdida de paquetes en el nivel TCP y con un retardo mínimo de células en el nivel ATM.

REFERENCIAS

[1] W. Richard Stevens, "TCP/IP Illustrated, Volume 1," *Addison-Wesley Professional Computing Series*, (1994).

[2] Romanow, A. and Floyd, S., "Dynamics of TCP traffic over ATM networks," *IEEE Journal on Selected Areas in Communications*, pp. 633-641, (1995).

[3] UCB/LBL/VINT Network Simulator - ns, *http://www-mash.cs.berkeley.edu/ns/*

[4] V. Jacobson, "Congestion Avoidance and Control," *ACM Computer Communications Review, proceedings of SIGCOMM'88*, pp. 314-329 (Aug. 1988).

[5] L.S. Brakmo, S. W. A'Malley, and L.L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," *Proceedings of SIGCOMM'94* (1994).

[6] K. Djemame, and M. Kara, "Proposals for a Coherent Approach to Cooperation between TCP and ATM Congestion Control Algorithms," *Proceedings UKPEW'99*, pp. 273-284, *http://www.cs.bris.ac.uk/Events/UKPEW1999/proceedings/* (1999).

[7] Robert Morris, "Scalable TCP Congestion Control," *Proceedings IEEE INFOCOM'2000*, pp.1176-1183, (2000).

[8] András Veres, and Miklós Boda, "The Chaotic Nature of TCP Congestion Control," *Proceedings IEEE INFOCOM'2000*, pp. 1715-1723, (2000).

[9] Amit Aggarwal, Stefan Savageand Thomas Anderson, "Understanding the performance of TCP Pacing," *Proceedings IEEE INFOCOM'2000*, pp. 1157-1165, (2000).

[10] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott, "The Macroscopic behavior of the TCP Congestion Avoidance Algorithm," *Computer Communications Review of ACM SIGCOMM*, vol 27, n. 3, (1997)

[11] Sally Floyd, "Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic," *Computer Communications Review*, vol 21, n. 5 (1995).

[12] Hongqing Li, Kai-Yeung Siu, Hong-Yi Tzeng, Ikeda, C., and Suzuki, H., "A simulation study of TCP performance in ATM networks with ABR and UBR services," *Proceedings IEEE INFOCOM'96*, pp. 1269-1276, (1996).

[13] Shunsaku Nagata, Naotaka Morita, Hiromi Noguchi, and Kou Miyake, "An analysis of the impact of suspending cell discarding in TCP-over-ATM," *Proccedings IEEE INFOCOM'2000*, pp. 1147-1156, (2000).