Estudio y simulación de TCP en redes de conmutación óptica de ráfagas (OBS)

Óscar González de Dios[1,2], Ignacio de Miguel[2], Víctor López Álvarez[1],
Ramón J. Durán[2], Noemí Merayo[2], Jesús Felipe Lobo Poyo[1]
[1] Telefónica I+D, Emilio Vargas 6, 28043 Madrid
e-mail: {ogondio, vla, jflp}@tid.es
[2] Dpto. de Teoría de la Señal y Comunicaciones e Ingeniería Telemática, Universidad de Valladolid
Campus "Miguel Delibes", Camino del Cementerio s/n, 47011 Valladolid
e-mail {ignacio.miguel, ramdur, noemer}@tel.uva.es

Resumen

La conmutación óptica de ráfagas (OBS) ha surgido como alternativa prometedora para la implantación de redes ópticas de área amplia. Asimismo, TCP seguirá siendo el protocolo de transporte dominante. En el presente artículo se realiza un estudio de la problemática que surge cuando TCP se emplea en redes OBS, apoyándose en simulación. Se presentan dos modelos de simulación de TCP sobre OBS que amplían el modelo empleado típicamente en la literatura. Se tienen en cuenta aspectos de TCP que condicionan en gran medida los resultados, y que no han sido considerados por trabajos anteriores, y se analiza un entorno más realista que en los trabajos previos, al combinar tráfico sintético con tráfico TCP. Como resultado se muestra que para el estudio de TCP sobre OBS no basta con considerar un único cliente y servidor, y que TCP y OBS son tecnologías compatibles, siempre que la red OBS se dimensione correctamente.

1. Introducción

El crecimiento espectacular del tráfico en Internet ha acelerado la investigación en tecnologías y redes de comunicaciones ópticas debido al gran ancho de banda que ofrece la fibra óptica. Una primera alternativa en el diseño de redes de comunicaciones ópticas son las redes con encaminamiento por longitud de onda [1]. Se trata de una tecnología basada en conmutación óptica de circuitos, siendo una alternativa viable y relativamente madura, pero que desaprovecha el ancho de banda. Una alternativa que soluciona este inconveniente es la conmutación óptica de paquetes [2]. Sin embargo, esta tecnología tiene aún una notable falta de madurez, lo que no permite pensar en ella como una alternativa viable a corto o medio plazo. Como solución intermedia ha surgido una tecnología prometedora, la conmutación óptica de ráfagas (OBS, *Optical Burst Switching*) [3]. OBS combina las ventajas de la conmutación de circuitos y de la conmutación de paquetes, lo que permite aprovechar el ancho de banda de una manera eficiente, siendo además una alternativa viable a medio plazo. Estas razones han hecho que OBS esté recibiendo gran atención por parte de la comunidad científica.

Por otro lado, TCP [4] se ha convertido en el estándar de facto de los protocolos de transporte, y se emplea por la mayor parte de las aplicaciones, como la navegación web, la transferencia de ficheros por FTP, el correo electrónico, etc. Todas las previsiones apuntan a que TCP seguirá siendo, en el medio y largo plazo, el protocolo de transporte dominante. Aquí es donde surge la necesidad de estudiar TCP en redes OBS, puesto que su combinación puede ser una alternativa viable para la próxima generación de Internet óptica de alta velocidad.

En primer lugar, en las Secciones 2 y 3, explicamos el funcionamiento de las redes ópticas de conmutación de ráfagas (OBS) y las características básicas del protocolo de transporte TCP. Una vez conocidos los fundamentos, analizamos los problemas con los que TCP se encuentra en las redes OBS. TCP está diseñado pensando en las redes actuales, en las que los nodos intermedios tienen capacidad de almacenamiento para absorber un exceso puntual de tráfico, por lo que una pérdida de información se debe a una congestión por exceso de tráfico durante un largo periodo de tiempo. Sin embargo, en OBS, la capacidad de almacenamiento es limitada o nula, y una perdida se debe a una congestión puntual, que no tiene por qué implicar un volumen de tráfico excesivo. En la Sección 4 se explica el comportamiento de TCP en el caso de la pérdida de ráfagas, que es la unidad básica de transmisión en OBS. Finalmente, mediante la herramienta de simulación OPNET Modeler [5] se han desarrollado y estudiado dos modelos para evaluar el comportamiento de TCP en OBS, y que amplían el modelo empleado típicamente en la literatura. Estos modelos, presentados en la Sección 5, tienen en cuenta aspectos de TCP, no considerados en modelos anteriores, que condicionan en gran medida los resultados, además de combinar tráfico TCP con tráfico sintético.

2. Introducción a la conmutación óptica de ráfagas (OBS)

Las redes de conmutación óptica de ráfagas se encuentran a medio camino entre las redes de conmutación óptica de circuitos y las de paquetes. La arquitectura de red OBS (Figura 1) consta de dos

tipos de nodos: los nodos de ingreso y los nodos troncales. Los paquetes, habitualmente datagramas IP, llegan a los nodos de ingreso, donde se clasifican según su destino y clase de servicio, y se agrupan electrónicamente formando ráfagas. Estas ráfagas se transmiten de manera totalmente óptica por el núcleo de la red hasta llegar al destino (otro nodo de ingreso), donde se desensambla la ráfaga y se recuperan los paquetes IP.

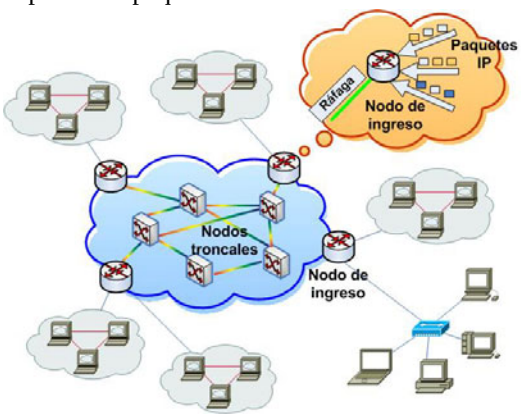


Figura 1: Arquitectura de una red OBS

Esta transmisión totalmente óptica se puede producir gracias a que los nodos troncales son configurados con anterioridad a la llegada de la ráfaga. Para ello, antes de la transmisión de la ráfaga, el nodo de ingreso envía un paquete de control (*Burst Control Packet*, BCP) a los nodos troncales, indicándoles que se va a proceder a transmisión de una ráfaga, así como el momento en el que se va a producir y la longitud esperada de la misma. Un cierto tiempo después (denominado tiempo de *offset*), se envía la ráfaga. Durante ese tiempo de *offset*, los nodos intermedios tienen tiempo para reservar recursos para el transporte de la ráfaga que se va a transmitir. Este protocolo se denomina JET (*Just-Enough Time*) [3] y se ha convertido en la alternativa más popular en OBS, superando a otras opciones. Cuando un nodo intermedio recibe el BCP, elige un enlace de salida, y a continuación, mediante un algoritmo de planificación, escoge una longitud de onda de salida para la ráfaga asociada. Puede darse el caso de que dos o más ráfagas necesiten el mismo recurso en el mismo instante de tiempo. En este caso se produce una contienda. En las redes clásicas, este problema se solventaba mediante el almacenamiento electrónico en una memoria, habitualmente denominada *buffer*, hasta que el recurso quedara libre. Sin embargo, cuando la transmisión se realiza por medios ópticos, el almacenamiento se complica. Actualmente la tecnología de almacenamiento óptico está muy inmadura, ya que no existe el equivalente de una memoria RAM óptica. Una solución a la contienda es el empleo de bucles de fibra óptica (FDL, *fiber delay lines*) que retrasan la ráfaga un tiempo fijo determinado. Otras soluciones consisten en cambiar la longitud de onda de una de las ráfagas (de forma que puedan ir simultáneamente por la misma fibra de salida), encaminar una de las ráfagas por otro puerto de salida distinto del que le correspondería (encaminamiento por deflexión), o descartar la ráfaga.

2.1. Ensamblado de ráfagas

Uno de los aspectos clave de OBS es el proceso de agregación de paquetes en los nodos de ingreso para formar ráfagas. Existen diversos métodos, denominados algoritmos de ensamblado [3,6-8]. El más común, y el que se emplea y simula en el presente estudio, es el ensamblado basado en temporizador. Cuando llega un paquete, y no se ha empezado a formar una ráfaga, se crea una nueva ráfaga y se activa un temporizador. Los paquetes que llegan posteriormente se van añadiendo a la ráfaga hasta que vence el temporizador.

También existen algoritmos de ensamblado por longitud de ráfaga. En estos algoritmos se añaden paquetes hasta que se alcanza a un tamaño de ráfaga fijado.

Ambos algoritmos tienen sus ventajas y desventajas. En los algoritmos con temporizador puede ocurrir que el valor de temporizador sea muy bajo, lo cual incrementa la carga de los paquetes de control. Si por el contrario se emplea un temporizador muy alto, el retardo introducido por el proceso de formación de la ráfaga puede ser intolerable. Por otro lado, los algoritmos basados en longitud de ráfaga no controlan el tiempo de finalización de la ráfaga, con lo que el retardo de los paquetes no está acotado, de modo que no es adecuado para muchas aplicaciones.

Por estos motivos han aparecido versiones mixtas de ambos algoritmos, en los que finaliza una ráfaga bien por temporizador, bien por tamaño, buscando un equilibrio entre ambos. Por otro lado, existen algoritmos adaptativos, en los que los valores del temporizador y/o del tamaño de las ráfagas se adaptan a las características del tráfico existente en cada momento [8].

3. Introducción a TCP

TCP se ha convertido en el estándar de facto de los protocolos de transporte y es empleado en la actualidad por la mayor parte de aplicaciones [4]. Este protocolo se encarga de la comunicación extremo a extremo, para lo cual emplea las facilidades que le proporciona la capa de red, habitualmente IP. TCP envía la información en partes, denominadas segmentos, que son asentidos por el receptor. Cada segmento está numerado, para posibilitar su ordenamiento en el destino y detectar posibles pérdidas.

3.1. Control de flujo y congestión

Para realizar la tarea de control de flujo, TCP emplea un mecanismo de ventana deslizante. El tamaño de la ventana de transmisión determina cuántos datos pueden estar en tránsito, es decir, cuántos datos pueden haber sido enviados por el transmisor sin que aún se haya recibido su asentimiento. Cuando se tienen en tránsito tantos

datos como indica la ventana de transmisión, el emisor deja de enviar nuevos segmentos. Cada vez que se recibe un asentimiento, y si el valor de la ventana de transmisión lo permite, TCP transmite nuevos datos. El valor de la ventana de transmisión se determina por el mínimo de dos límites, uno que está impuesto por el receptor, e indica el tamaño del *buffer* de recepción disponible, para evitar saturar el receptor, y otro impuesto por el propio emisor, denominado ventana de congestión, que tiene el objetivo de evitar el envío de más datos que los que la red soporta. TCP tiene varias fases en la transmisión, en las que el valor de la ventana de congestión varía de manera diferente. TCP empieza la comunicación probando la red, enviando un solo segmento[1], y fija el tamaño de la ventana de congestión (*cwnd*) a un segmento. En cuanto recibe el asentimiento la ventana aumenta *cwnd* a dos, con lo que se envían dos segmentos, y así sucesivamente, resultando en un aumento exponencial de la ventana de congestión, como se puede observar en la Figura 2.a. Este comportamiento se denomina *slow start*, y finaliza cuando la ventana de congestión alcanza un umbral denominado *ssthresh* (*show start threshold*). A continuación, la ventana de congestión aumenta de forma lineal en la fase denominada *congestion avoidance*, con el fin de no saturar la red. En la Figura 2.b se observa la evolución de la ventana de transmisión, que, como se ha mencionado, es el mínimo entre la ventana de congestión y el tamaño del buffer de recepción.

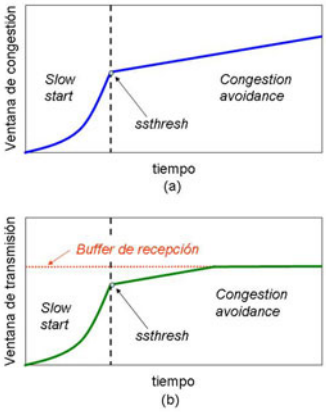


Figura 2: (a) Evolución de la ventana de congestión en TCP. (b) Evolución de la ventana de transmisión en TCP.

Por lo tanto, el valor de la ventana de transmisión en cada momento da una indicación de la tasa de transmisión de TCP, ya que, en ausencia de errores se va a transmitir durante un RTT (*Round Trip Time)* como máximo tantos segmentos como indique la ventana de transmisión. (El RTT se define como el tiempo que transcurre desde que se envía un segmento hasta que llega su asentimiento). Así pues, la relación entre la tasa de transmisión, $X(t)$, expresada en segmentos por segundo, y la ventana de transmisión, $W(t)$, expresada en segmentos, es

$$X(t) = \frac{W(t)}{RTT}. \quad (1)$$

De esta forma, el rendimiento máximo de TCP viene dado por

$$X_{\max} = \frac{W_{\max}}{RTT}, \quad (2)$$

siendo $W_{\max}$ el tamaño máximo que puede alcanzar la ventana de transmisión [10].

TCP es un protocolo de transporte que garantiza la fiabilidad de la transmisión. Con tal fin dispone de una serie de mecanismos para actuar en caso de que se pierdan los segmentos. A lo largo de la historia de TCP, se han ido incorporando nuevos métodos de recuperación ante la pérdida de segmentos, que se explican en los apartados del 3.2 al 3.5. Como TCP está pensado para las redes actuales, una pérdida de segmentos es un indicativo de congestión en la red, y el objetivo de la recuperación es no sólo retransmitir los segmentos perdidos, sino evitar que vuelva a haber congestión en la red.

3.2. Temporizador de retransmisión

TCP, como hemos comentado, numera los segmentos para poder reordenarlos si llegan en desorden y detectar pérdidas de segmentos. Un asentimiento confirma la recepción no sólo de un segmento, sino de todos los bytes transmitidos hasta el número indicado en el asentimiento. Cuando el emisor envía un segmento, pone en marcha un temporizador denominado "temporizador de retransmisión". Si el asentimiento de dicho segmento llega antes del vencimiento del temporizador, se entiende que el segmento ha llegado correctamente. Sin embargo, si el temporizador vence antes de la llegada del asentimiento de dicho paquete, se entiende que el segmento se ha perdido. Tras el vencimiento del temporizador, el emisor ha de retransmitir el segmento y volver a iniciar el proceso de *slow start*. Esta drástica reducción de la tasa de transmisión obedece a que TCP asume que si ha habido una pérdida de segmento es porque se ha producido una congestión en algún punto de la red, y para evitar saturarla, reduce la tasa de transmisión.

3.3. Fast Retransmit

La técnica de *fast retransmit* [9] permite que un emisor conozca que se ha perdido un segmento incluso antes de que venza el temporizador de retransmisión. Cuando a un receptor TCP le llega un segmento cuyo número de secuencia no es el que espera, sino uno posterior y por lo tanto en desorden, debe enviar un ACK (asentimiento) duplicado inmediatamente, es decir, vuelve a asentir los mismos datos que ya asintió anteriormente. La técnica *fast retransmit* consiste en que en cuanto llegan al emisor tres asentimientos duplicados (cuatro ACKs idénticos), éste retransmite el

[1] Realmente, la RFC 2581 [9] permite empezar la comunicación enviando más de un segmento, pero este comportamiento es poco habitual y queda fuera de nuestra discusión.

segmento sin esperar a que venza el temporizador de retransmisión.

3.4. Fast Recovery

Con el fin de no reducir de manera abrupta la tasa de transmisión cuando se detecta la pérdida de un segmento, se introdujo el algoritmo *fast recovery* [9]. Este algoritmo funciona de la siguiente forma

1. Cuando llega el tercer ACK duplicado, se actualiza el valor de *ssthresh* a la mitad de *flightsize* (número de segmentos en tránsito, es decir que han sido enviados pero aún no han sido asentidos), se retransmite el segmento perdido, y se pone *cwnd* a *ssthresh* más 3 veces el tamaño máximo de segmento (el valor de este tamaño es un parámetro configurable).
2. Cada vez que llegue un nuevo ACK duplicado, se incrementa *cwnd* en el valor del tamaño máximo de segmento. En caso de que el número de segmentos en tránsito sea menor que el valor de la ventana de transmisión, se transmite un nuevo segmento.
3. Cuando llegue el ACK que asienta otro segmento, es decir, un ACK no duplicado, se actualiza *cwnd* con el valor de *ssthresh*. A continuación, se continua en la fase *congestion avoidance*.

Los mecanismos *fast retransmit* y *fast recovery* son eficaces cuando se pierde un único segmento. Sin embargo, cuando se pierden varios segmentos consecutivos, el algoritmo no se recupera con facilidad. Esto se debe a que después de la recuperación de la pérdida de un segmento, la ventana de congestión generalmente queda reducida a la mitad del valor que tenía la ventana de transmisión, con lo que en caso de que el asentimiento que llegue sea parcial, es decir, que no asienta a todos los segmentos que se habían enviado antes de entrar en *fast recovery*, la ventana es tan pequeña que no se pueden enviar nuevos segmentos, por lo que se ha de esperar al vencimiento del temporizador para continuar con la retransmisión. La versión de TCP que emplea *fast retransmit* y *fast recovery* tal y como se han descrito, se denomina TCP Reno. Existe una versión de TCP que modifica el comportamiento de *fast recovery*, denominada NewReno [11], cuyo estudio queda fuera del objetivo de este artículo.

3.5. Asentimiento selectivo

Con el objetivo de mejorar el comportamiento ante pérdidas múltiples, aparece el asentimiento selectivo (SACK, *Selective Acknowledgement*) [12] [13]. Éste consiste en que en el asentimiento se incluye un campo adicional, donde, cuando llegan segmentos en desorden, el receptor informa del número de segmento que ha llegado fuera de orden, así como cuántos segmentos contiguos a partir de él han sido recibidos. De esta manera, el emisor se hace una idea de cuántos segmentos se han perdido y cuántos han llegado al receptor. Con esta información, el emisor retransmite los segmentos que se han perdido sin esperar al temporizador de retransmisión.

TCP SACK usa los mismos mecanismos para el control de la congestión que Reno (o NewReno). Como en el caso de Reno, cuando el emisor recibe el tercer ACK duplicado, entra en la fase *fast recovery*, y reduce a la mitad la ventana de congestión. En este caso, además, el emisor mantiene una nueva variable de estado, *pipe*, que indica el número de segmentos que se han enviado, pero no han recibido asentimiento. Se transmite (o retransmite) únicamente cuando *pipe* es menor que la ventana de congestión. La variable *pipe* se incrementa en uno cada vez que se envía un segmento, y se reduce cada vez que se reciben asentimientos con ciertas características. Si se recibe un asentimiento duplicado en el que su información de asentimiento selectivo indica que han llegado nuevos segmentos al receptor, entonces la variable *pipe* se reduce en una unidad. Si se recibe un asentimiento parcial, es decir, un asentimiento no duplicado pero que no confirma todos los segmentos enviados antes de entrar en *fast recovery*, se reduce en dos. La razón de reducirse en dos, es que se supone que se han enviado dos segmentos, el segmento original, y el segmento retransmitido. El objetivo de este mecanismo es no parar la transmisión, y continuar enviando nuevos segmentos además de retransmitir los segmentos perdidos. Una vez que ha llegado el asentimiento que confirma la recepción de todos los datos perdidos, incluyendo todos los datos que se habían enviado hasta el momento en el que se produjo la detección de la pérdida, se continúa con *congestion avoidance*.

3.6. Versiones de TCP

El comportamiento de TCP está especificado en las RFC (*Request for Comments*). Sin embargo, no hay un TCP "estándar", sino que existen versiones de TCP que emplean los distintos algoritmos para el control de la congestión. En la Tabla 1 se muestra un resumen de las versiones de TCP más conocidas.

Tabla 1: Versiones de TCP

	Tahoe	Reno	NewReno	SACK
Slow Start, congestion avoidance	Sí	Sí	Sí	Sí
Fast Retransmit	Sí	Sí	Sí	Sí
Fast Recovery	NO	Sí (Reno)	Sí (New Reno)	Sí (Reno o New Reno)
ACK selectivo	NO	NO	NO	Sí

4. Impacto de la pérdida de ráfagas en TCP

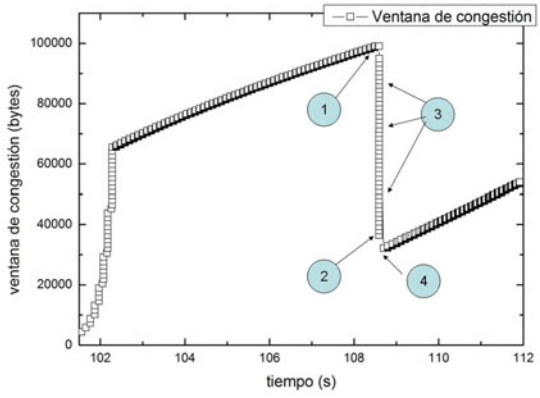
En esta sección se analiza el comportamiento básico de TCP en redes OBS desde un punto de vista cualitativo, pero apoyado en simulaciones. El

análisis se centrará en las versiones de TCP Reno y SACK.

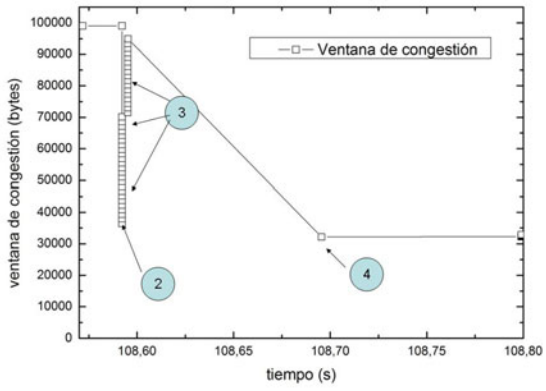
Los datagramas IP, los cuales contienen segmentos TCP, se agrupan en ráfagas en los nodos de ingreso de la red OBS. Este ensamblado de los paquetes introduce un retardo adicional, debido al tiempo de espera hasta que se completa la formación de la ráfaga, que reduce el rendimiento de TCP. El efecto es similar al de incrementar el RTT, que impacta en la tasa de transmisión como indica la ecuación (1) del apartado 3.1. Sin embargo, el aspecto que más influye en el rendimiento de TCP es la pérdida de ráfagas debido a la contienda en los nodos intermedios. La pérdida de una ráfaga implica generalmente la pérdida de varios segmentos consecutivos pertenecientes a un mismo flujo. Tal y como se comentó en el apartado anterior, TCP tiene dos maneras de detectar la pérdida de un segmento, bien mediante la recepción de asentimientos duplicados, bien mediante el vencimiento de un temporizador. En OBS, las ráfagas contienen varios segmentos TCP, con lo que dependiendo de la cantidad de segmentos que se transmitan con relación al tamaño de la ventana de transmisión, TCP tiene una reacción u otra. Si en una ráfaga se transmite una ventana de TCP completa, vencerá el temporizador de retransmisión y comenzará la fase de *slow start*. En cambio, si no se transmite una ventana completa, llegarán al receptor segmentos fuera de orden, con lo que enviará asentimientos duplicados, que pondrán en marcha los mecanismos de recuperación de TCP explicados en la sección anterior. Por lo tanto, en el caso de que se pierda una ráfaga, la influencia en el rendimiento de TCP viene dada, no por el número total de segmentos que viajan en dicha ráfaga, sino por el número total de segmentos de un mismo flujo. Se pueden tener perdidas en una ráfaga desde 1 hasta *n* segmentos de un mismo flujo, donde *n* es el número máximo de segmentos que caben en la ventana de transmisión. Principalmente son tres los casos interesantes, que abordamos en los apartados siguientes.

4.1. Pérdida de ráfagas conteniendo un solo segmento de un mismo flujo TCP

En el caso de que la ráfaga que se pierde contenga un único segmento de un mismo flujo TCP, y la ventana de transmisión sea mayor que un segmento, al receptor van a llegar segmentos fuera de orden, que provocarán el envío de asentimientos duplicados, uno por cada segmento fuera de orden. El valor de la ventana de transmisión cuando se recibe el primer duplicado determina cuántos segmentos pueden estar en tránsito. Si este número es mayor o igual que cuatro, entonces llegarán al receptor tres segmentos fuera de orden, generando tres asentimientos duplicados, que al llegar al emisor hacen que éste inicie sus mecanismos de protección ante pérdidas. En este caso, los asentimientos duplicados van a llegar siempre antes de que venza el temporizador, y TCP se va a poder recuperar en poco tiempo y sin reducir tan drásticamente la tasa de transmisión como lo haría si venciera éste. Es de interés destacar que en los primeros instantes de un flujo TCP, la ventana tiene menos de cuatro segmentos, con lo que una pérdida de una ráfaga al comienzo de la transmisión provocaría el vencimiento del temporizador con la consiguiente penalización en el rendimiento.



(a)



(b)

Figura 3: Pérdida de una ráfaga conteniendo un solo segmento de un mismo flujo con TCP Reno. (a) Transmisión completa (b) Detalle de la zona de recuperación.

Para estudiar este escenario en Reno, vamos a emplear la Figura 3, donde se muestra la evolución de la ventana de congestión en el tiempo para la transmisión de ráfagas conteniendo un único segmento de un flujo en cada ráfaga, y en la que se produce la pérdida de una ráfaga.

En la Figura 3.a se muestra la transmisión completa, y la Figura 3.b el detalle de la recuperación. En ambas aparecen numerados los puntos de interés. En el punto ① se produce la pérdida de la ráfaga. Cuando llegan los tres asentimientos duplicados, entran en acción los mecanismos *fast retransmit* y *fast recovery* explicados en la sección anterior, con lo que primero se reduce la ventana a *flightsize*/2+3 (punto ② de la Figura 3), se retransmite el segmento, se incrementa artificialmente la ventana cada vez que llega un asentimiento (punto ③), y finalmente se reduce la ventana a la mitad del número de segmentos en tránsito (punto ④).

En el caso de SACK (Figura 4), tras la pérdida de paquete (punto ①), cuando llegan los tres asentimientos duplicados (punto ②), se reduce la ventana a la mitad del número de segmentos en tránsito y se retransmite. A continuación van llegando más asentimientos duplicados (punto ③). Un RTT más tarde del envío del segmento retransmitido, llega el asentimiento que confirma su recepción, con lo que se continúa creciendo con *congestion avoidance* (punto ④).

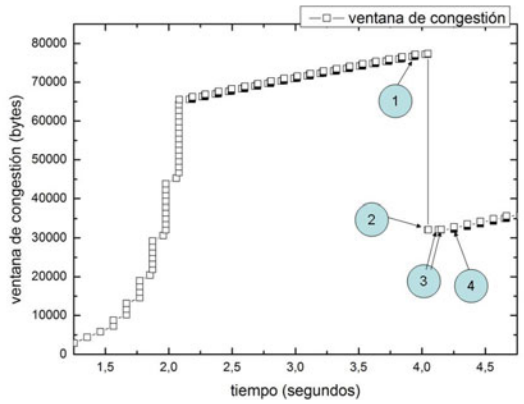


Figura 4: Pérdida de una ráfaga conteniendo un solo segmento de un mismo flujo con TCP SACK.

Tanto Reno como SACK consiguen recuperarse de la pérdida de un segmento en poco tiempo, un RTT desde la detección de los ACKs duplicados. Después de la recuperación en ambas versiones el resultado es que la ventana de congestión se reduce a la mitad del valor que tenía la ventana de transmisión antes de producirse las pérdidas. Por lo tanto, en este caso, el asentimiento selectivo no aporta ventajas significativas en el rendimiento.

4.2. Pérdida de ráfagas conteniendo dos o más segmentos de un mismo flujo TCP

Cuando se pierde una ráfaga que lleva dos o más segmentos de un mismo flujo, pueden darse dos escenarios posibles para la detección de la pérdida. Si la ventana de transmisión tiene un tamaño mayor o igual que el número de segmentos perdidos más tres, entonces el emisor acabará recibiendo tres asentimientos duplicados, detectando así la pérdida. Si la ventana de transmisión no alcanza dicho valor, entonces vencerá el temporizador de retransmisión. El caso de mayor interés, en el que Reno y SACK muestran diferente comportamiento, es el de la recepción de tres ACK duplicados, que procedemos a explicar mediante un ejemplo. Supondremos que se pierde una ráfaga que contiene dos segmentos de un flujo TCP y que la siguiente ráfaga transmitida (con nuevos segmentos) llega correctamente. La Figura 5 muestra el comportamiento de Reno y la Figura 6 el de SACK.

En el caso de TCP Reno, tras producirse la pérdida (punto ①), el emisor va a retransmitir el segmento, y va a reducir la ventana de congestión a la mitad más tres segmentos (punto ②). A medida que van llegando nuevos asentimientos duplicados, la ventana de congestión se va incrementando en un segmento (punto ③). Sin embargo, no puede enviar nuevos segmentos, ya que la ventana de transmisión es superior a la ventana de congestión, ya que tiene almacenados todos los segmentos que se han enviado después de la pérdida. A continuación, llega el asentimiento del primer segmento perdido, que ha sido generado tras la recepción del segmento retransmitido, y la ventana de congestión se reduce (punto ④). La ventana de transmisión en este punto, con el número de segmentos que hay en tránsito, no permite el envío de ningún segmento, por lo tanto el emisor permanece inactivo hasta que vence el temporizador de retransmisor asociado al segundo segmento perdido (punto ⑤) , momento en el que se retransmite el segmento y se vuelve a *slow start*.

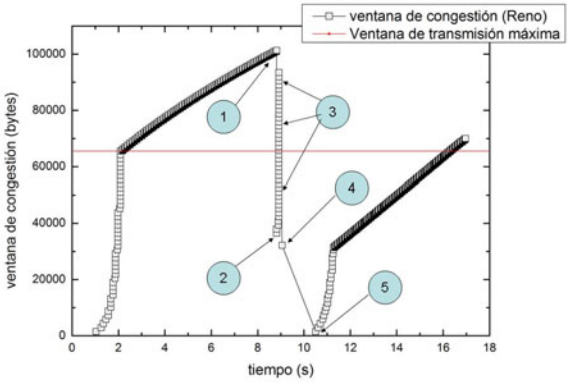


Figura 5: Pérdida de una ráfaga conteniendo dos segmentos de un mismo flujo con TCP Reno.

En el caso de SACK, tras la pérdida de la ráfaga que lleva dos segmentos del flujo considerado (punto ①), cuando ocurre la detección de los tres asentimientos duplicados, se reduce la ventana a la mitad del número de segmentos en tránsito (punto ②). A medida que llegan los asentimientos duplicados con información de asentimientos selectivos, la variable *pipe* se va actualizando, como se ha explicado en el apartado 3.5, pero la ventana de congestión permanece constante (punto ③). Con la información de los asentimientos selectivos, re retransmiten los segmentos perdidos. Cuando llega el asentimiento que confirma la recepción de todos los seguidos, se continúa con la fase de *congestion avoidance* (punto ④).

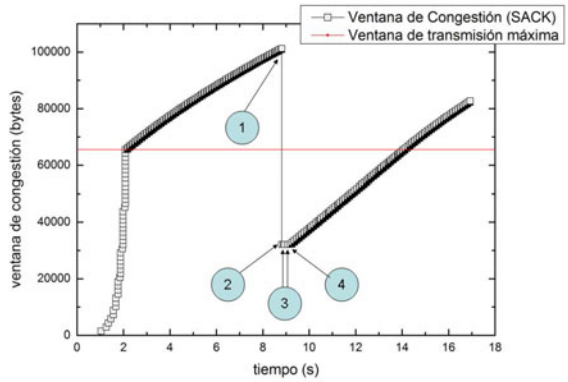


Figura 6: Pérdida de una ráfaga conteniendo dos segmentos de un mismo flujo con TCP SACK.

El caso que hemos explicado corresponde a la pérdida de una ráfaga que lleve dos segmentos pertenecientes a un mismo flujo TCP, pero si la ráfaga lleva más de dos segmentos (siempre que lleguen tres ACK duplicados) el comportamiento es similar.

En este escenario, la primera versión de TCP analizada, Reno, se recupera mediante el vencimiento del temporizador y continúa la transmisión con *slow start*, reduciendo drásticamente la tasa de transmisión. Sin embargo SACK se recupera en aproximadamente un RTT y continúa la transmisión con la ventana reducida a la mitad de lo que tenia antes en vez de reducirla a un segmento como en Reno. Por lo tanto, SACK ofrece una mejora significativa de rendimiento, por lo que su uso es altamente recomendable al transmitir en redes OBS.

4.3. Pérdida de ráfagas conteniendo una ventana completa de un mismo flujo TCP

En el caso de que en una ráfaga contenga todos los segmentos enviados en una ventana de transmisión TCP, la pérdida de la misma implica el vencimiento del temporizador. Esto se debe a que el emisor TCP, tanto Reno como SACK, no recibe asentimientos duplicados que le informen de dicha pérdida (Figura 7).

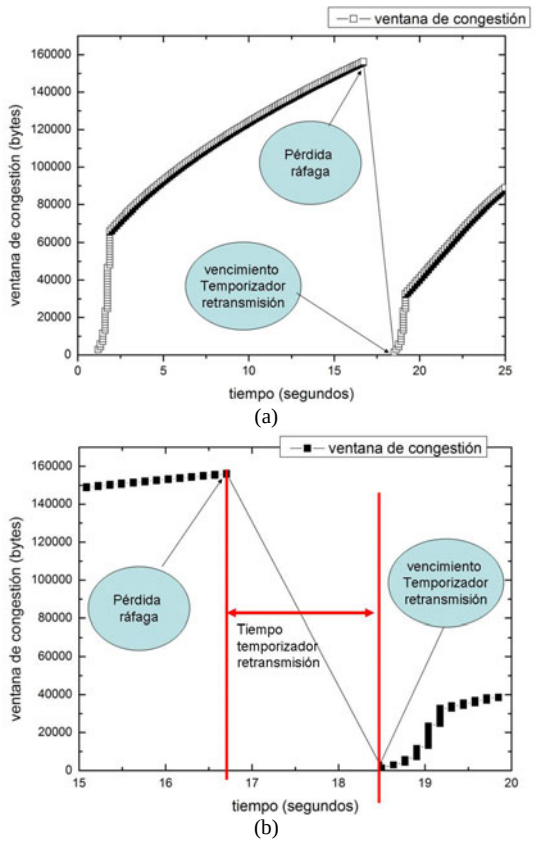


Figura 7: Pérdida de una ráfaga conteniendo una ventana TCP completa. (a) Transmisión completa (b) Detalle de la zona de recuperación.

4.4. Clasificación de fuentes de tráfico

Teniendo en cuenta los análisis que acabamos de mostrar, surge la necesidad de conocer cuál es el número de segmentos de un mismo flujo que transporta cada ráfaga. Si el número de segmentos de un mismo flujo en cada ráfaga es elevado, cuando hay una pérdida de ráfaga, el tiempo de recuperación es alto, lo que conlleva una bajada del rendimiento. Ahora bien, si el número de segmentos de un mismo flujo en cada ráfaga es uno, el comportamiento de TCP es similar al habitual en las redes actuales. Y si estamos en un caso intermedio, siempre que se use SACK el comportamiento también se acercará al de las redes actuales, ya que no se ve afectado exceso por perdidas de varios segmentos.

La literatura [14] [15] se ha centrado en analizar el ensamblado en ráfagas basado en temporizador, y distingue entre fuentes lentas, medias y rápidas. Yu et al. [16] establecen los criterios para que todo el contenido de una ventana viaje en una ráfaga o en varias. Según la velocidad de acceso (λ , medida en segmentos por segundo) y el periodo de ensamblado (T_b, medido en segundos) distinguen tres tipos de flujo:

- Flujo rápido: Si se verifica $\lambda \cdot T_b \geq W_{\max} - 1$, entonces todo el contenido de la ventana de transmisión se transmite en una ráfaga.
- Flujo medio: Si se verifica $1 \leq \lambda \cdot T_b < W_{\max}$, entonces el contenido de la ventana se reparte en varias ráfagas.
- Flujo lento: Si se verifica $\lambda \cdot T_b < 1$, entonces una ráfaga contiene un único segmento.

Sin embargo, estos estudios se basan en un modelo con un único cliente y servidor, por lo que hay que tener cuidado. Como se mostrará mediante simulación en el apartado 5.5, cuando se tiene en cuenta tráfico adicional, el número de segmentos por ráfaga disminuye, y por lo tanto, no se puede hablar estrictamente de flujos medios, lentos o rápidos.

5. Simulaciones de TCP sobre OBS

Con el objetivo de evaluar el rendimiento, ahora sí, desde un punto de vista cuantitativo, se ha desarrollado un simulador de TCP sobre OBS utilizando la herramienta OPNET Modeler 11.0 [5]. En primer lugar, se ha replicado el modelo de simulación empleado en los estudios de Detti *et al.* [14] y Yu *et al.* [15, 16], y a los que denominaremos ‘modelos clásicos’. A continuación hemos desarrollado dos modelos de simulación más completos, un primer modelo que denominaremos ‘modelo básico’, en el que se añade el algoritmo *delayed* ACK (o ACK retardado) de TCP [17] y un segundo modelo que denominaremos ‘modelo con varias fuentes de tráfico’, en el que no sólo se tiene un cliente y un servidor TCP, sino que se considera la existencia de tráfico adicional en la red. En este apartado mostraremos las características y los resultados obtenidos con estos nuevos modelos.

5.1. Modelo básico

El primer modelo que se ha desarrollado está basado en los modelos clásicos, incorporando características adicionales. Los modelos clásicos sólo modelaban un sentido de la transmisión, y los asentimientos emitidos por el receptor eran enviados por un camino ideal sin pérdidas. En nuestro modelo básico (Figura 8), los asentimientos se transmiten igualmente por la red OBS, pasando por el proceso de ensamblado de ráfagas, con la correspondiente probabilidad de pérdida de ráfaga. Los modelos clásicos además también obvian el algoritmo *delayed* ACK de TCP. Éste es un algoritmo por el cual los asentimientos de TCP no se envían nada más recibir un segmento, sino que se espera o bien a la llegada de otro segmento o bien al vencimiento de un temporizador. Ahora bien, se ha demostrado que el uso de este algoritmo en las redes OBS tiene un impacto significativo en el rendimiento [18], por lo que en nuestro modelo esta característica también está incluida.

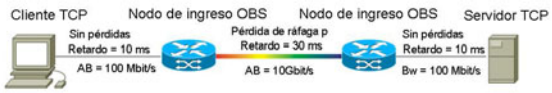


Figura 8: Modelo básico de simulación de TCP sobre OBS.

5.2. Modelo con varias fuentes de tráfico

El modelo básico, y por extensión los modelos empleados en la literatura hasta este momento, sólo consideran un flujo TCP entre un cliente y un servidor. Sin embargo, en un escenario real habrá miles de clientes y servidores intercambiando flujos TCP. Por lo tanto, es necesario considerar este escenario más realista. Para ello, parece lógico introducir en el modelo en vez de un solo cliente, un número elevado (1000 ó 10000). Sin embargo, un modelo con tal número de flujos TCP hace que el tiempo necesario para realizar una simulación sea demasiado elevado. Así pues, se ha desarrollado un modelo alternativo, cuyo esquema se puede ver en la Figura 9, en el que el nodo OBS está conectado a un cliente TCP y a un generador de tráfico que simula el tráfico de una LAN. Se ha empleado un generador de tráfico fractal. Este es un modelo de tráfico autosimilar que está reconocido en la literatura como generador con exactitud del tráfico de una LAN [19] De esta manera, el generador de ráfagas se alimenta de una manera más real que al considerar un único cliente y servidor TCP.

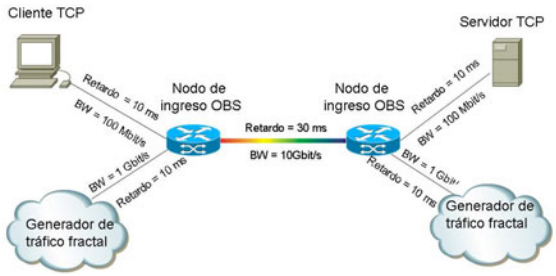


Figura 9: Modelo con varias fuentes de tráfico para la simulación de TCP sobre OBS.

5.3. Parámetros de las simulaciones

El objetivo de este estudio es determinar la viabilidad de TCP sobre OBS. Por lo tanto, los valores de los parámetros que necesitamos en las simulaciones han de ser lo más cercanos a la realidad. Teniendo en cuenta que OBS es una tecnología para el medio-largo plazo, se ha analizado la evolución de los parámetros a lo largo del tiempo, y se han escogido valores coherentes con dicha progresión.

5.3.1. Ancho de banda de la red de acceso

En la actualidad (septiembre 2005) el ancho de banda de acceso residencial con ADSL o cable oscila entre los cientos de kbit/segundo y unos pocos Mbit/segundo en el sentido descendente. Sin embargo, tecnologías venideras como ADSL2, y finalmente VDSL, posibilitarán un ancho de banda mayor en el futuro. En concreto, el estándar VDSL promete 100 Mbit/s simétricos. Por lo tanto escogeremos esta cifra para nuestro estudio.

5.3.2. Retardo de la red de acceso

Actualmente el retardo de acceso es alto, del orden de varias decenas de milisegundos, debido por ejemplo en el caso del ADSL, al *interleaving*. Por lo tanto, se espera que con la mejora de la tecnología en el medio plazo este valor descienda. Consideraremos un peor caso como 10 ms.

5.3.3. Retardo del núcleo de la red OBS y ancho de banda

Con respecto al retardo de la red, este es debido al retardo de propagación entre origen y destino. Por lo tanto habría que considerar diferentes escenarios, desde unos 5 ms hasta 30 ms (los cuales se corresponden con distancias entre los nodos de ingreso origen y destino comprendidas entre 1000 y 6000 km, al estar conectados los nodos mediante enlaces de fibra óptica). En el presente estudio consideramos el peor escenario, de 30 ms, si bien en estudios futuros habría que analizar todos los casos. Como ancho de banda del canal se escoge un valor habitual en los sistemas de fibra óptica, 10 Gbit/s.

5.3.4. Tamaño de los flujos TCP

Vamos a estudiar no sólo el rendimiento en el estado estacionario de TCP para un flujo de duración larga, sino también el de flujos de duración corta. Por lo tanto, simularemos la transferencia de ficheros grandes, de 20 Mbytes, y la transferencia de ficheros de 100 kbytes.

5.3.5. Parámetros de TCP

TCP ha ido evolucionando a lo largo de sus años de vida. Para elegir los parámetros de TCP se ha empleado como referencia un estudio sobre la evolución de los protocolos de transporte en Internet [20]. Los valores elegidos se muestran en la tabla 2. Mención especial merece el buffer de recepción, que limita el máximo de la ventana de transmisión. La ventana de transmisión máxima tenía en el año 2000 una media de 18 kbytes, y en el 2004 de 44 kbytes.

Es de esperar que siga subiendo, como mínimo a 64 kbytes, que es el máximo sin emplear escalado de ventana (una opción de TCP para incrementar el tamaño máximo de este valor [21]). En este estudio se ha empleado el búfer de recepción máximo sin escalado de ventana. Sin embargo, en estudios posteriores se considerará escalado de ventana con escalados pequeños. Con respecto a la versión de TCP, se ha elegido SACK por ser la versión que está implantándose cada vez más, así como Reno, ya que es la versión de referencia en la literatura.

Tabla 2: Parámetros de TCP

Parámetro	Valor
Tamaño máximo segmento	1460 bytes
Buffer de recepción típico (máximo de la ventana de transmisión)	65535 bytes
Escalado de ventana	NO
Versión TCP	SACK y Reno
Delayed ACK	SI (temporizador de este algoritmo configurado a 200 ms)
Temporizador de retransmisión mínimo	1 segundo

5.4. Simulaciones con el modelo básico

En primer lugar, se ha realizado una simulación con el modelo básico, con el objetivo de buscar el valor óptimo del temporizador que controla el proceso de construcción de ráfagas, utilizando TCP SACK. La simulación ha consistido en la transferencia de 100 ficheros de 20 Mbytes bajo distintas condiciones de probabilidad de pérdida de ráfagas. Se ha medido el tiempo empleado en realizar la transferencia de cada fichero, y el rendimiento se ha calculado dividiendo el tamaño del fichero entre el tiempo necesario para realizar la transferencia.

El resultado, que podemos ver en la Figura 10 (y en la que se muestran los intervalos de confianza del 95%), muestra un rendimiento óptimo cuando el temporizador de ensamblado es aproximadamente un milisegundo. Con todas las probabilidades de pérdida de ráfagas consideradas, se observa que cuando el temporizador de ensamblado toma un valor suficientemente pequeño, no hay variaciones en el rendimiento por el hecho de disminuir más el temporizador, ya que a partir de ese punto todas las ráfagas contienen un único segmento TCP. La variación del rendimiento con el valor del temporizador es más significativa en la zona intermedia, donde el número de segmentos de un flujo por ráfaga varía, encontrándose el óptimo en la región del milisegundo. Estos resultados son coherentes con los obtenidos en la literatura [16], a pesar de emplear un modelo ligeramente modificado.

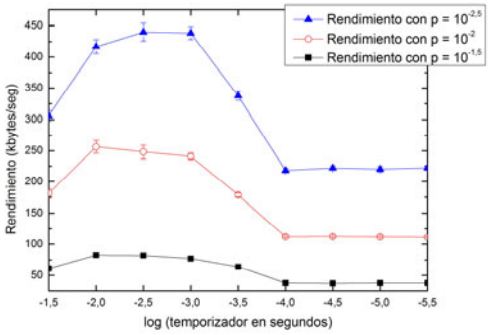


Figura 10: Rendimiento frente al temporizador de ensamblado para distintas probabilidades de pérdida de ráfagas (*p*).

5.5. Simulación del número de segmentos por ráfaga

Cada ráfaga en OBS contendrá segmentos de muchos flujos TCP. El objetivo de este conjunto de simulaciones es determinar cuál es el número de segmentos de un mismo flujo TCP por ráfaga, ya que esta es una cuestión crítica, como pusimos de manifiesto en la Sección 4. Para ello, se ha simulado el envío de ficheros de 20 Mbytes por una red OBS sin pérdidas. Se ha simulado la transferencia de 500 ficheros, y se ha calculado el histograma normalizado del número de segmentos de un mismo flujo TCP por ráfaga. Se realiza la misma simulación en los distintos modelos para observar las diferencias de comportamiento.

En primer lugar se emplea el modelo clásico. El resultado, que se puede ver en la Figura 11, es muy determinista, y muestra que prácticamente todas las ráfagas tienen ocho o nueve segmentos.

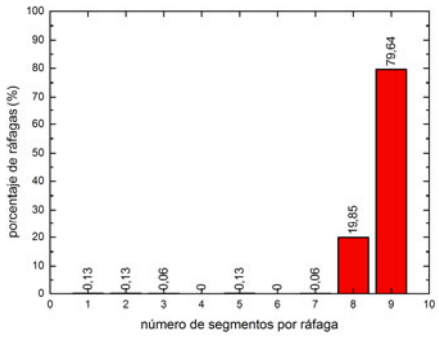


Figura 11: Número de segmentos de un flujo TCP por ráfaga en el 'modelo clásico'.

Cuando ampliamos el modelo al modelo básico del presente artículo, se observa en la Figura 12 cómo el número de segmentos por ráfaga se reparte un poco más entre valores más bajos.

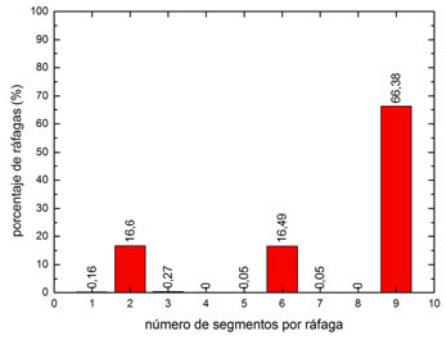


Figura 12: Número de segmentos de un flujo TCP por ráfaga en el 'modelo básico'.

Finalmente, cuando empleamos el modelo con varias fuentes de tráfico, se observa un comportamiento radicalmente distinto. Tal y como se aprecia en la Figura 13, la gran mayoría de las ráfagas contiene dos segmentos, en vez de ocho o nueve como se observaba en el primer caso. Cuando consideramos que el generador de ráfagas es alimentado no sólo por un flujo TCP sino también por tráfico adicional, es frecuente que se produzca el siguiente escenario. Cuando llega un segmento del flujo TCP al nodo de ingreso OBS, no tiene por qué iniciar una el proceso de formación de una ráfaga nueva, pues es posible que un segmento asociado a las otras fuentes de tráfico ya haya puesto en marcha el mecanismo de formación de la ráfaga. De esta forma, puede que reste poco tiempo para la finalización de la misma, y que por ello no puedan introducirse en ella tantos segmentos del flujo TCP como en el caso anterior. El resto de los segmentos se agregarán en la siguiente ráfaga. La razón de que tienda a dos segmentos es que se está empleando asentimiento retardado, que generalmente da lugar a que se asientan dos segmentos a la vez en lugar de uno solo. De esta forma, cuando el emisor recibe un asentimiento, transmite dos nuevos segmentos, que en la mayoría de los casos viajan juntos en la misma ráfaga, aunque cuando lleguen al ensamblador de ráfagas, el tiempo restante para la finalización del ensamblado sea pequeño.

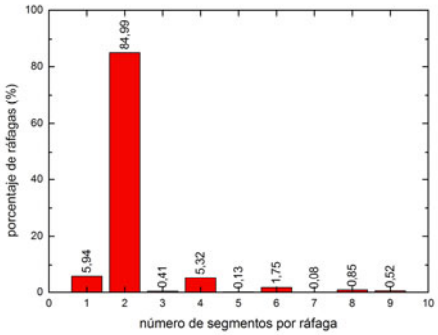


Figura 13: Número de segmentos de un flujo TCP por ráfaga en el 'modelo con varias fuentes de tráfico'

Este resultado tiene una gran importancia, ya que muestra que cuando se tiene en cuenta que un nodo OBS es alimentado por múltiples clientes en vez de uno sólo, el comportamiento es distinto del que se consideraba hasta ahora, siendo más cercano al que se produce en una red de paquetes habitual.

5.6. Simulación del rendimiento frente al temporizador de ensamblado

El objetivo de esta simulación es triple, por un lado comparar el rendimiento estacionario de TCP considerando o no tráfico adicional, obtener el valor óptimo del temporizador de ensamblado y comparar el rendimiento de dos versiones de TCP, Reno y SACK. Para ello, se va a obtener el rendimiento para distintos valores del temporizador de formación de ráfagas empleando el modelo básico y el modelo con varias fuentes de tráfico.

La simulación ha consistido en la transferencia de 100 ficheros de 20 Megabytes y la probabilidad de pérdida de ráfagas se ha fijado a 10^{-3}. Probabilidades mayores pueden ser poco realistas, ya que un operador no instalaría una red con pérdidas elevadas.

Además, se han escogido valores del temporizador lo más realista posibles, en el rango de los estudios previos de la literatura.

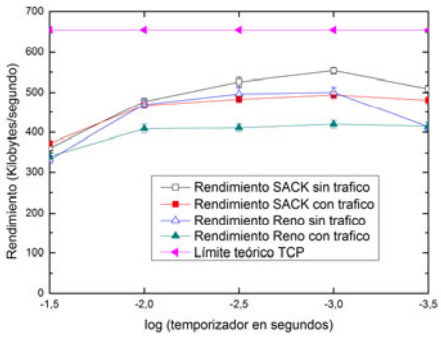


Figura 14: Rendimiento de un flujo TCP de 20 Megabytes en el 'modelo básico' y en el 'modelo con varias fuentes de tráfico'

En la **Figura 14** se muestra el resultado de la simulación, en la que se observa el rendimiento obtenido para distintas probabilidades de pérdida de ráfagas y distintos tiempos de temporizador de creación de ráfagas. Cuando se considera una única fuente de tráfico, el rendimiento máximo se obtiene para valores del temporizador cercanos a 1 ms. Cuando consideramos un escenario más realista con varias fuentes de tráfico adicionales, la diferencia de rendimiento entre varios valores del temporizador es insignificante. Esto se debe al efecto que se ha observado en la simulación anterior, el número de segmentos por ráfaga no es el máximo que permite teóricamente el temporizador. Incluso, el rendimiento obtenido cuando tenemos en cuenta tráfico adicional es inferior en la zona de temporizador óptimo al obtenido cuando no consideramos tráfico adicional. Esto se debe a que, si bien cada vez que hay una pérdida, la recuperación es mejor, al ser el número de segmentos de un mismo flujo por ráfaga menor, el mayor número de ráfagas perdidas contrarresta este beneficio, y en global se obtiene peor rendimiento.

Por otro lado, se observa que SACK siempre obtiene mejor rendimiento que Reno debido a su mejor comportamiento en pérdidas de múltiples segmentos.

5.7. Simulación del rendimiento frente al tiempo del temporizador de ensamblado con ficheros de tamaño grande y pequeño

El objetivo de esta simulación es comparar el rendimiento de TCP en estado estacionario y en una conexión corta. Para ello, se ha calculado el rendimiento variando el valor del temporizador de ensamblado para un fichero grande, de 20 Mbytes, y uno pequeño, de 100 kbytes. La simulación se ha realizado con TCP SACK, tras haber mostrado mejor comportamiento que Reno, y con una probabilidad de pérdida de ráfagas de 10^{-3}.

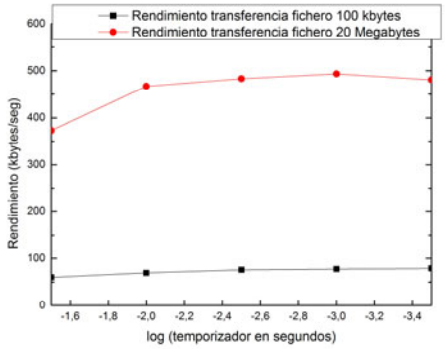


Figura 15: Rendimiento de un flujo TCP de 100 kbytes y otro de 20 Megabytes

En el caso de una conexión corta, el rendimiento, que se puede ver en la Figura 15 es menor, del orden de 8 veces más pequeño que con el flujo grande. Este hecho se explica por la evolución de la ventana de transmisión. En una conexión corta, no se puede aprovechar el máximo de la ventana de transmisión ya que casi toda la transmisión pertenece a la fase de *slow start*. Sin embargo, cuando consideramos un flujo TCP grande, medimos el comportamiento estacionario de TCP, y por lo tanto, la ventana de transmisión tendrá en media un valor mayor. Por otro lado, una pérdida de ráfaga en una conexión corta, tiene un gran impacto en el rendimiento, en cuyo caso el tiempo necesario para la transferencia se puede duplicar o triplicar. Este hecho puede ser especialmente molesto en el caso de la navegación web, donde se espera siempre una respuesta inmediata.

6. Conclusiones

En este artículo se han revisado los mecanismos de TCP para hacer frente a las pérdidas de segmentos, y se ha explicado cómo influyen en el caso concreto de las redes de conmutación óptica de ráfagas (OBS). En las redes OBS se pierden ráfagas, que perjudican en gran medida el rendimiento de TCP, aunque dependiendo de la versión de TCP y del número de segmentos de un flujo por ráfaga, el impacto es mayor o menor. Se ha demostrado que SACK es la versión de TCP que mejor se adapta a redes OBS.

También se ha confirmado que es posible encontrar un valor de temporizador óptimo cuando se estudia un flujo TCP estacionario de manera aislada, pero se ha demostrado que cuando se estudia un flujo TCP junto con tráfico adicional, la distinción entre fuentes lentas, medias y rápidas no es tal, ya que el comportamiento es muy parecido en todos los casos, obteniéndose un rendimiento similar en un rango de valores de temporizador de ensamblado.

Cuando estudiamos TCP considerando múltiples fuentes de tráfico en una red OBS, nos encontramos con que una pérdida de ráfaga no es siempre tan negativa como se pensaba en estudios previos, al contener la mayoría de ráfagas pocos segmentos.

Por otro lado, el rendimiento en términos de velocidad de transferencia de un flujo estacionario de TCP en OBS es mucho mayor que el obtenido en la transferencia de un fichero pequeño, como es el caso de una navegación web.

En el caso de una implementación por parte de un operador de una red OBS, se ha de garantizar una que la probabilidad de pérdida de ráfaga sea lo suficientemente baja como para permitir un buen rendimiento de TCP.

Por lo tanto, OBS es una tecnología adecuada para transmitir tráfico TCP. Sin embargo, no todas las versiones de TCP obtienen el mismo rendimiento, y es altamente recomendable emplear SACK.

Como trabajo futuro, se va a estudiar el comportamiento con mayores ventanas de transmisión y con diferentes valores de los retardos de acceso y transmisión, y varios valores de carga de la red. Asimismo, se estudiarán nuevas versiones de TCP, para evaluar si mejoran el rendimiento y es necesario promoverlas.

Agradecimientos

Este trabajo ha sido financiado parcialmente por el proyecto integrado NOBEL, por el Ministerio de Ciencia y Tecnología (proyecto TIC2002-03859) y por la Junta de Castilla y León (proyecto VA037/04). Los autores agradecen a Javier Aracil sus sugerencias y comentarios sobre este trabajo.

Referencias

[1] R. Ramaswami y K. N. Sivarajan, *Optical Networks: A Practical Perspective*, Morgan Kauffman Publishers, Inc., 1998.

[2] T. S. El-Bawab y J.-D. Shin, “Optical Packet Switching in Core Networks: Between Vision and Reality”, *IEEE Communications Magazine*, vol. 40, no. 9, Septiembre 2002, pp. 60-65.

[3] Y. Chen, C. Qiao y X. Yu, “Optical Burst Switching: A New Area in Optical Networking Research”, *IEEE Network*, vol. 18, no. 3, pp. 16-23 Mayo/Junio 2004.

[4] R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, 1994

[5] OPNET Modeler. http://www.opnet.com
[6] Y. Xiong, M. Vandenhoute y H. Cankaya, "Control Architecture in Optical Bursa-Switched WDM Networks", *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, Octubre 2000, pp. 1838-1851.
[7] X. Yu, Y. Chen y C. Qiao, "Study of Traffic Statistics of Assembled Burst Traffic in Optical Burst Switched Networks", *Proc. Opticomm 2002*, pp. 149-159.
[8] M. Düser y P. Bayvel, "Burst aggregation control and scalability of wavelength-routed optical-burst-switched (WR-OBS) networks", *Proc. ECOC 2002*, paper 2.4.7.
[9] M. Allman, V. Paxson y W. Stevens, "TCP Congestion Control", *RFC 2581*, Abril 1999.
[10] M. Hassan y R. Jain, *High Performance TCP/IP Networking: Concepts, Issues, and Solutions*, Prentice-Hall, 2003.
[11] S. Floyd, "The NewReno Modification to TCP's Fast Recovery Algorithm", *RFC 2582*, Abril 1999.
[12] M. Mathis, J. Mahdavi, S. Floyd y A. Romanow, "TCP Selective Acknowledgment Options", *RFC 2018*, Octubre 1996.
[13] K. Fall y S. Floyd "Simulation-based Comparisons of Tahoe, Reno and SACK TCP", *ACM SIGCOMM Computer Communication Review*, vol. 26, Julio 1996, pp. 5-21
[14] A. Detti y M. Listanti, "Impact of Segments Aggregation on TCP Reno Flows in Optical Burst Switching Networks", *Proc. of INFOCOM 2002*, Vol. 3, pp. 1803-1812.
[15] X. Yu, C. Qiao, Y. Liu y D. Towsley, "Performance Evaluation of TCP Implementations in OBS Networks", *Technical Report 2003-13*, CSE Dept., SUNY, Buffalo, 2003
[16] X. Yu, C. Qiao y Y. Liu, "TCP Implementations and False Timeout Detection in OBS Networks", *Proc. of INFOCOM 2004*, pp. 774-784.
[17] M. Allman, "On the Generation and Use of TCP Acknowledgments", *ACM SIGCOMM Computer Communication Review*, Vol. 28, Octubre 1998, pp. 4-21.
[18] O. González, I. de Miguel, N. Merayo, P. Fernández, R. M. Lorenzo y E. J. Abril, "The Impact of Delayed ACK in TCP Flows in OBS Networks", *Proceedings of NOC 2005*, pp. 367-374, 2005
[19] B. Ryu y S. Lowen, "Fractal Traffic Models for Internet Simulation", *IEEE Int'l Symposium on Computer Communications*, Julio 2000, pp. 200-206.
[20] A. Medina, M. Allman y S. Floyd, "Measuring the evolution of Transport protocols in the Internet", *ACM SIGCOMM Computer Communication Review*, Vol. 35, Abril 2005, pp. 37-52
[21] V. Jacobson, R. Braden y D. Borma, "TCP Extensions for High Performance", *RFC 1323*, Mayo 1992.