9 TCP en Enlaces de Baja Velocidad: Compresión de Cabeceras

9.1 INTRODUCCIÓN

En enlaces de baja velocidad y en enlaces móviles con ancho de banda reducido, es importante optimizar la relación entre datos de usuario y datos de cabecera, para obtener buenos rendimientos en las comunicaciones. Ya hemos comentado en capítulos anteriores que una forma de paliar los efectos de la baja velocidad de los enlaces y el reducido ancho de banda es reducir las cabeceras de los protocolos de red. En este caso particular, hemos evaluado el algoritmo de *Van Jacobson* [RFC1144] de reducción de cabeceras TCP/IP. En capítulos anteriores ya se han comentado las ventajas de este mecanismo. En este capítulo se evaluará el algoritmo de compresión de cabeceras de Van Jacobson [RFC1144], que es el que se incorpora en la mayoría de implementaciones, evaluando los problemas que éste tiene en caso de que se utilice en enlaces móviles con errores. Se estudiarán las diferentes alternativas que aparecen en la bibliografía y se propondrán mejoras.

9.2 ENTORNO DE EVALUACIÓN

Para todos los casos presentados en este capítulo se ha utilizado el mismo entorno presentado en el capítulo de *Comportamiento de TCP en Entornos Móviles: Efecto de los Errores*. Las pruebas realizadas, por tanto, son mediante una implementación real de los protocolos. En los casos de propuesta de algoritmos, éstos se han implementado en el entorno Linux.

9.3 PROBLEMÁTICA DE LA COMPRESIÓN DE CABECERAS EN ENTORNOS MÓVILES

Está claro que la aplicación de algoritmos de compresión de cabeceras como el de Van Jacobson, da buenos resultados en cuanto a caudal ya que se reduce la cantidad de información no útil respecto la útil, basándose en la reconstrucción de la cabecera original a partir de una resumida. Este algoritmo tiene buenas prestaciones en caso de que no existan factores externos que perturben su funcionamiento. En la Figura 9.1, se muestra que la compresión de cabeceras de *Van Jacobson* resulta en una ganancia de la eficiencia de la comunicación de alrededor del 20%, ya que se obtiene un caudal de 902 bytes/s con compresión y de 746 bytes/s sin ella. En este caso la MTU es de 296 bytes y la velocidad del enlace de 9600bps. Además, cuanto menor sea el tamaño del paquete, el hecho de contar con mecanismos de compresión de cabeceras dará mejores prestaciones.

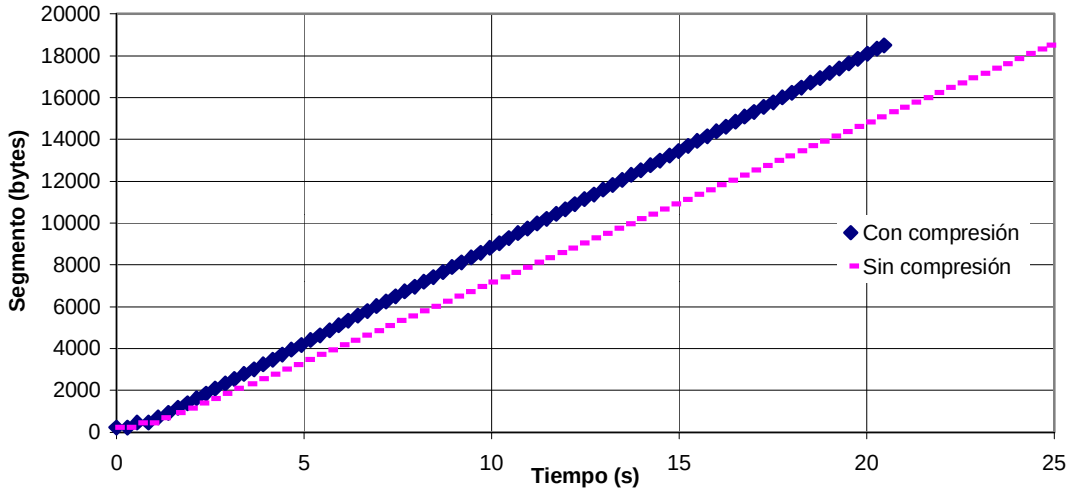


Figura 9.1 Evolución de una transferencia ftp sin errores.

En enlaces fiables con escaso ancho de banda, el método de compresión de cabeceras de Van Jacobson es extremadamente útil. Lo que nos proponemos determinar a continuación es si las ventajas ofrecidas por este método son igualmente extensibles a enlaces en los que los errores de transmisión son frecuentes y aparecen incluso en forma de ráfagas.

Mientras este método reduce la probabilidad de error de un paquete (ya que se reduce su tamaño), el método en sí crea una dependencia entre cada segmento y el anterior. Veremos pues como esta dependencia supone un incremento de la probabilidad de error percibida, ya que la pérdida de un segmento produce la pérdida de toda la ventana de transmisión de TCP. La Figura 9.2 muestra un ejemplo en el que se ponen de manifiesto estos efectos. En ella se presentan dos transferencias *ftp* de un fichero de 20 Kbytes, a una velocidad de 19200 bps y una MTU de 296 bytes.

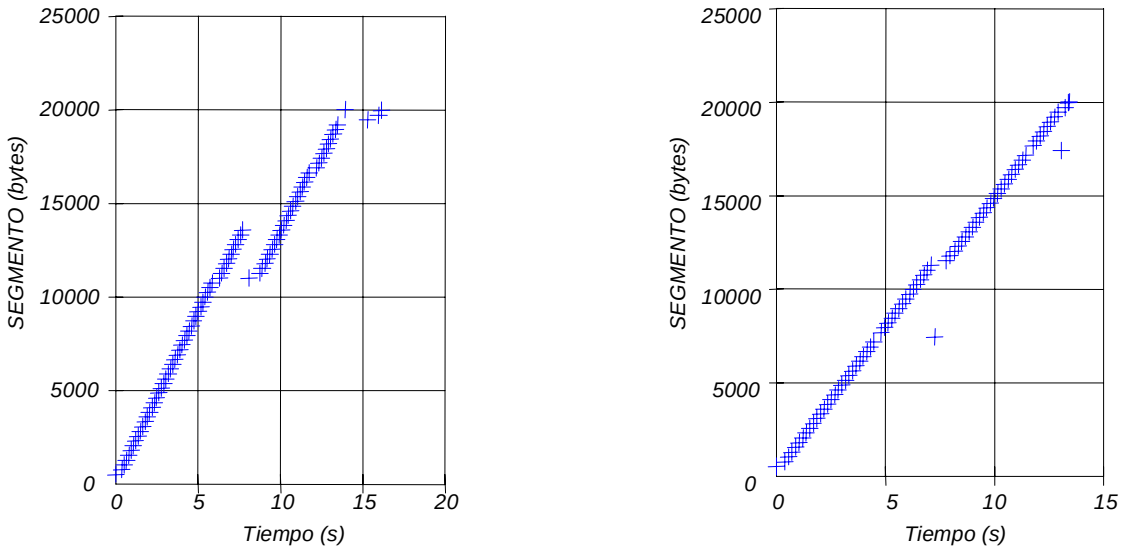


Figura 9.2 Evolución de una transferencia con compresión (izquierda) y sin compresión (derecha)

Cuando se produce un error, en caso de compresión, el descompresor intenta reconstruir el siguiente segmento recibido después del erróneo. Pero en este caso, la cabecera base para la reconstrucción por incrementos es diferente en el compresor y en el descompresor, con lo cuál, esta pérdida de consistencia provoca la descompresión errónea de todos los segmentos siguientes al erróneo. Por lo tanto, además de perder todos los segmentos de la ventana de transmisión, el receptor TCP deja de recibir segmentos (ya que el descompresor los descarta por falta de consistencia), y consecuentemente el algoritmo de Retransmisión Rápida no puede activarse ya que éstos segmentos no generarán reconocimientos duplicados. En este caso, siempre deberá expirar el temporizador de retransmisión para recuperar el segmento perdido. Finalmente, se retransmitirán el resto de segmentos de la ventana tras la expiración de los temporizadores correspondientes.

Por otra parte, en el caso de no tener compresión de cabeceras, el segmento erróneo se descarta , y se retransmite o por el cumplimiento del temporizador de retransmisión o tras la actuación del algoritmo de retransmisión rápida. Puede verse en la Figura 9.2 (derecha), que el resto de segmentos de la ventana de transmisión después del erróneo no son retransmitidos. Comparando ambas figuras puede observarse que, si bien la pendiente en caso de compresión es mayor, el efecto de los errores penaliza el total de la transmisión, obteniendo mayor caudal en el caso de no comprimir las cabeceras. En los apartados siguientes se mostrarán estos efectos de forma más clara.

Podemos afirmar que cuando un segmento resulta afectado por un error, el módulo TCP en emisión tarda un cierto tiempo en recuperarse y continuar enviando nuevos datos, siendo este tiempo aproximadamente:

$$Tiempo\ de\ Recuperación \cong RTO + v \times T \qquad (9.1)$$

donde **RTO** es el valor de la variable tiempo de retransmisión en ese momento, **v** es el tamaño de la ventana de transmisión útil en el momento de producirse el error, y **T** el tiempo de transmisión de un segmento.

El primer término de la ecuación (9.1) corresponde al tiempo de detección del error. Puesto que el receptor no genera reconocimientos ni para el segmento erróneo ni para los siguientes (ya que la estrategia de compresión provoca que éstos sean también desechados en el descompresor), el emisor deberá esperar a la expiración del temporizador de retransmisión para darse cuenta del problema.

El segundo término, por otro lado, corresponde al periodo de tiempo que el emisor debe invertir en retransmitir todos y cada uno de los segmentos pertenecientes a la ventana del erróneo.

En cambio, en el caso de no tener compresión y si no se activa el algoritmo de Retransmisión Rápida (sería el peor de los casos), la pérdida no se recuperará hasta transcurridos RTO segundos, no obstante, si las ventanas de transmisión y congestión lo permiten, no se interrumpirá el flujo de datos, siendo, el tiempo de recuperación aproximadamente:

$$RTO - [(v - 1) \times T] + T \qquad (9.2)$$

Hemos visto como en el caso de errores, la compresión de cabeceras provoca la retransmisión de toda la ventana de transmisión, su tamaño es un aspecto determinante en el caudal del TCP cuando éstos ocurren. A su vez, no obstante, veremos que el tamaño de la ventana de transmisión cuando los errores se producen depende además de la BER del canal. Por lo tanto, existirá un compromiso entre la BER y el tamaño de la ventana. De forma que si la BER crece:

- el caudal disminuye
- el tamaño de la ventana es pequeño, y por lo tanto los efectos negativos de la compresión de cabeceras de Van Jacobson son menos relevantes.

9.3.1 ERRORES EN RECONOCIMIENTOS

Otro aspecto relevante es el efecto que tienen los errores que afectan a segmentos de reconocimiento (*ACK*) sobre el rendimiento de TCP en su servicio de transmisión masiva de datos, cuando se utiliza el algoritmo Van Jacobson de compresión.

Debemos ver si el carácter acumulativo de los reconocimientos en TCP permite amortiguar, en cierta medida, la repercusión que tiene la pérdida, debida a la estrategia de compresión, de todos los reconocimientos asociados a los segmentos pertenecientes a la ventana de aquel cuyo ACK fue afectado por el error.

En la Figura 9.3 se muestra la evolución de los números de secuencia de los segmentos enviados por el TCP emisor junto con los de los reconocimientos que éste recibe. Lo primero que se observa es que, pese a haber introducido un único error, ninguno de los

reconocimientos correspondientes a los segmentos 3713:3945 a 4641:4873 es recibido. La explicación la encontramos en la interacción entre los errores y el algoritmo de compresión: puesto que los reconocimientos posteriores al erróneo dependen de éste para su correcta descompresión, el emisor los elimina en espera de un ACK autocontenido.

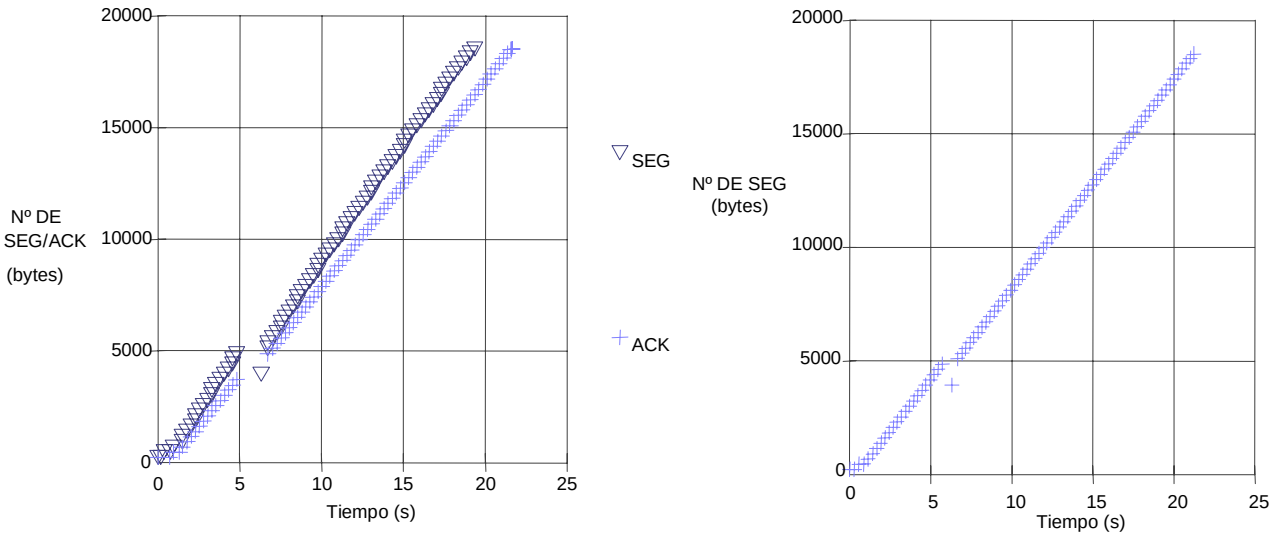


Figura 9.3 Emisión (izquierda) y recepción (derecha)

La falta de reconocimientos provoca, finalmente, la expiración del temporizador de retransmisión, con lo que el emisor acaba por retransmitir el primer segmento de la cola: 3713:3945. Al ser éste recibido, se genera un ACK idéntico al último enviado por el receptor y que, debido a esta circunstancia, se envía sin comprimir. Puesto que este ACK reconoce todos y cada uno de los segmentos de la cola de retransmisión, TCP sale del modo de retransmisión y continúa enviando normalmente.

Sin utilizar compresión, un error simple en un ACK, puesto que TCP incorpora ACK acumulativos, no obliga a TCP a retransmitir ningún segmento y, en consecuencia, el caudal conseguido es igual al obtenido en ausencia de errores. Aunque, en el caso que nos ocupa, la incorporación del algoritmo de Van Jacobson obliga al emisor a retransmitir el segmento asociado al reconocimiento perdido, la mayor eficiencia, en términos de volumen de información de control transmitida, que proporciona este algoritmo compensa sobradamente este hecho y permite obtener un caudal superior al conseguido en el caso sin compresión.

No obstante, en el caso de errores múltiples, no siempre el TCP puede aprovechar el carácter acumulativo de sus reconocimientos para evitar la retransmisión de más segmentos aparte del correspondiente al ACK perdido. Por ejemplo, es posible provocar errores en los reconocimientos de tal manera que el emisor se vea forzado a activar el mecanismo de backoff exponencial. Este caso es el que muestra la Figura 9.4.

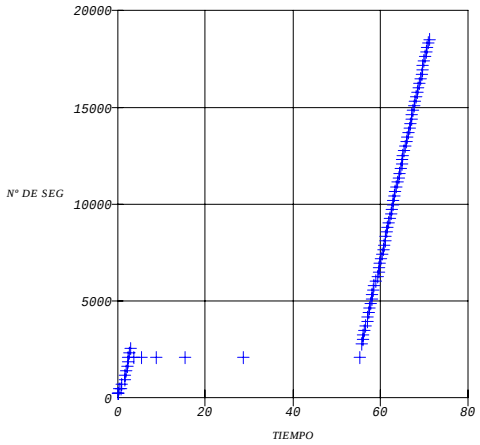


Figura 9.4 Evolución del número de secuencia en emisión

Tras detectar el error a través de la expiración del temporizador (ya que todos los reconocimientos recibidos con posterioridad al erróneo también son descartados), el emisor retransmite el primer segmento de la cola de retransmisión y dobla RTO. Sin embargo, el reconocimiento acumulativo no comprimido que el receptor genera tras recibir este segmento resulta también afectado por un error, de forma que el emisor queda, de nuevo, a la espera de un reconocimiento que no llegará nunca. Cuando el temporizador de retransmisión expira, el emisor vuelve a doblar RTO y retransmite el mismo segmento. Una vez más, el reconocimiento acumulativo resulta corrompido por un error y el emisor vuelve a depender del temporizador para detectar la situación que, como muestra la Figura 9.4 se repite en tres ocasiones más.

Hemos podido comprobar, una vez más, cómo el método de compresión de cabeceras de Van Jacobson, pese a estar diseñado, en principio, para optimizar el ancho de banda del enlace de comunicación, puede tener efectos contraproducentes cuando este enlace se caracteriza por la introducción de errores frecuentes.

9.3.2 EFECTO DE LA BER DEL CANAL

En los apartados anteriores se ha mostrado el efecto de errores simples en las transferencias, mostrando los efectos nocivos que éstos provocan en conjunción con los algoritmos de compresión de cabeceras. Vamos a ver ahora como afecta la distribución de los errores, comparando sus efectos entre errores simples y errores a ráfagas con la distribución de *Fritchman* detallada en el Anexo B. Para todas las transferencias realizadas se ha utilizado un fichero de prueba de 100Kbytes, y una MTU de 296bytes.

En la Figura 9.5 se reflejan las medidas de caudal obtenidas para el modelo de errores deterministas y a ráfagas en función tanto de la tasa de error en bit del canal como del uso o no de compresión.

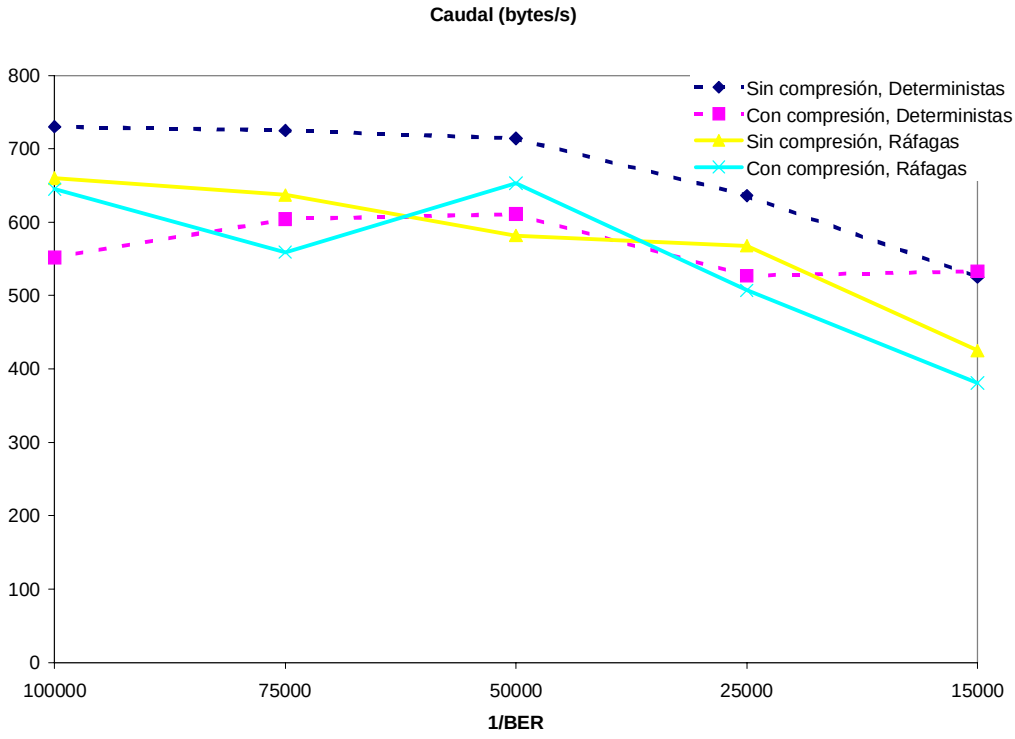


Figura 9.5 Evolución del caudal TCP. Errores Deterministas y Ráfagas

Comparando los resultados de caudal entre errores deterministas y errores a ráfagas, podemos ver que el caudal medido no siempre desciende conforme aumenta la BER del canal. Para entender esto debemos tener en cuenta la relación existente entre la BER del canal y el tamaño de la ventana del emisor en el momento de producirse un error.

Tomemos, por ejemplo, un enlace con una tasa de error en bit dada, BER, suficientemente baja como para que el caudal en recepción pueda calcularse con suficiente aproximación como:

$$Caudal = \frac{IU}{IU + INU} \times BW_{enlace} = \frac{IU}{IU + \left\lceil \frac{IU}{MSS} \right\rceil \times TC + NR \times MTU} BW_{enlace} \qquad (9.3)$$

donde INU es la información no útil, siendo TC el tamaño de la cabecera.

En ausencia de compresión, NR (Número de segmentos Retransmitidos) es aproximadamente igual a:

$$NR_{sin\ compresion} \approx IU\ bytes \times BER\ errores/bit \times 8\ bits/byte = 8 \times IU \times BER \qquad (9.4)$$

Ahora bien, cuando se utiliza compresión, cada error se traduce en toda una ventana de segmentos erróneos y, por tanto:

$$\mathrm{NR}_{\text{con compresion}} \approx \mathrm{NR}_{\text{sin compresion}} \times \mathrm{v} \qquad (9.5)$$

siendo **v** el tamaño medio de la ventana calculado a partir de los valores que adopta este parámetro en los momentos en que se producen los diversos errores.

El caudal puede, entonces, expresarse como:

$$Caudal = \frac{IU}{IU + \left\lceil \frac{IU}{MSS} \right\rceil \times TC + 8 \times IU \times BER \times v \times MTU} BW \qquad (9.6)$$

Sin embargo, **v** también depende de la BER del canal ya que cuanto mayor es la BER, tanto más próximos están los errores y, en consecuencia, tanto más pequeña se mantiene la ventana del emisor.

Se establece, por tanto, un compromiso entre la BER del canal y el tamaño de la ventana del emisor de manera que:

- Cuanto mayor es la BER del canal, mayor es el número de errores de transmisión que se generan durante la transferencia, es decir, mayor es el factor $8 \times \mathrm{IU} \times \mathrm{BER}$ de la ecuación (9.6). En este sentido, la mayor tasa de error tiene un impacto negativo en el caudal del enlace.
- Cuanto mayor es la BER del canal, menor es el tamaño que puede alcanzar la ventana de congestión del emisor entre error y error y, en consecuencia, menor es el factor **v** de la ecuación (9.6). En este sentido, la mayor tasa de error *ayuda a amortiguar* el efecto secundario negativo que supone la introducción del método de compresión.

Para comprobar cómo es este compromiso el responsable de la evolución mostrada por el caudal en la Figura 9.5 podemos fijarnos en la Figura 9.6 en la que se indica el número de bytes retransmitidos en cada uno de los casos, para errores deterministas.

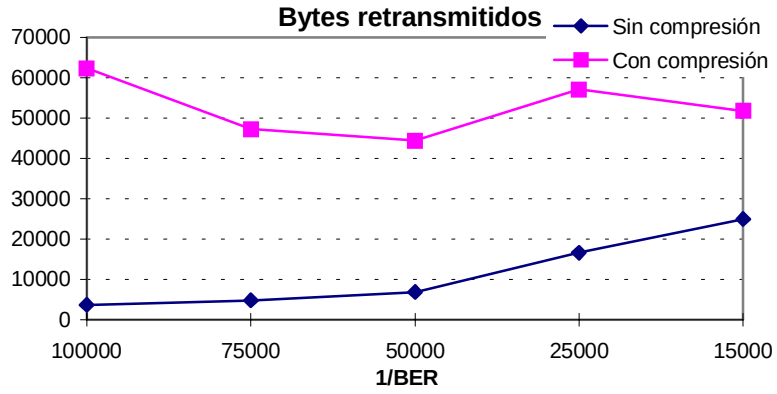


Figura 9.6 Número de bytes retransmitidos para el caso de un canal que introduce errores deterministas

Como puede apreciarse, en ausencia de compresión el número de bytes retransmitidos aumenta conforme lo hace la BER del canal. Con compresión, en cambio, aunque para BER=1/100000 el número de errores introducidos en el canal es inferior al correspondiente a los demás casos, el número total de bytes retransmitidos es claramente superior. Esto es debido a que el tamaño de la ventana en el momento de producirse el error oscila, en este

caso, entre los 9 y los 10 segmentos lo que supone que cada error obliga a retransmitir 10 segmentos. Para BER=1/75000 y BER=1/50000 también es el tamaño de la ventana el factor que determina el caudal alcanzado. No obstante, para la tasa de error en bit BER=1/25000, la Figura 9.6 permite constatar que el factor $8 \times \mathrm{IU} \times \mathrm{BER}$ de la ecuación (9.6), asociado al número de errores de transmisión que realmente introdujo el canal, tuvo mayor peso que **v**, dando lugar a un incremento en el número de bytes retransmitidos respecto a los dos casos anteriores. Finalmente, para el último caso mostrado, BER= 1/15000 el reducido tamaño de la ventana cuando ocurren los errores prima sobre la mayor tasa de error del canal y, en consecuencia, el caudal medido es superior al de BER=1/25000.

Siguiendo con la Figura 9.5, podemos ver que a medida que la tasa de error del canal aumenta se observa una clara tendencia a la aproximación entre el caudal final que se consigue tanto con como sin compresión de cabeceras. Pensemos que en el caso límite en que la BER fuese tan elevada que obligara a que la ventana se mantuviese, a lo largo de toda la transferencia, al valor de 1 segmento, tendrían aplicación todas las ventajas de la estrategia propuesta por Van Jacobson y ninguno de sus inconvenientes.

Como acabamos de ver, los errores deterministas afectan mucho más cuando se utiliza compresión que en ausencia de ella. En cambio, a tenor de los resultados de la Figura 9.5, no pasa lo mismo cuando los errores se generan en ráfaga. En este último caso, los resultados no son tan contundentes.

En la Figura 9.7 aparece el número medio de bytes retransmitidos para cada caso de los que se evalúan en la Figura 9.5 con errores en ráfaga.

En todos los casos que hemos analizado, el número de bytes retransmitidos es superior cuando se utiliza compresión que cuando ésta está inhabilitada. Aun así, como muestra la Figura 9.5, el caudal medido no es siempre inferior con compresión. En ocasiones, la retransmisión de más segmentos de los estrictamente necesarios que provoca la incorporación del algoritmo de compresión permite, aun incrementando la redundancia de la transferencia, evitar posteriores intervalos de espera relacionados con la falta de reconocimientos que permitan la detección de posibles nuevos errores.

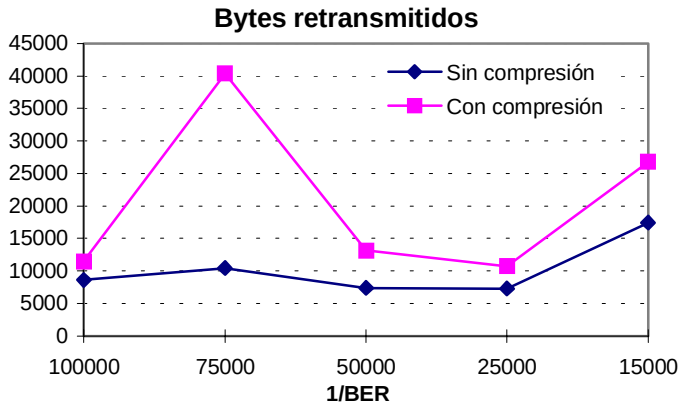


Figura 9.7 Bytes retransmitidos en caso de ráfagas

9.3.3 RESUMEN

A partir de los resultados que hemos obtenido, podemos extraer varias conclusiones sobre la conveniencia o no de utilizar el método de compresión propuesto por Van Jacobson en los enlaces radio móviles de baja velocidad:

1. La compresión es extremadamente beneficiosa en ausencia de errores, especialmente cuando se utiliza en enlaces cuyas características aconsejan el uso de un tamaño de paquete reducido.

   Podría pensarse que los enlaces móviles, al caracterizarse por el uso de valores de MTU muy reducidos, serían candidatos idóneos para la incorporación del algoritmo de compresión de cabeceras TCP/IP; sin embargo, las altas tasas de error que presentan estos enlaces y que los diferencian de aquellos para los que, inicialmente, fue pensado el algoritmo, obligan a pensar en adecuar estos algoritmos a este tipo de entornos.

2. Cuando el canal introduce errores, el algoritmo de Van Jacobson da lugar a dos efectos contrapuestos:

   - Por un lado, el método de Van Jacobson, en tanto que es un método de reducción del tamaño de los paquetes, permite también reducir, de forma proporcional, la probabilidad de error en éstos. Esto es así porque la probabilidad de error en paquete puede expresarse a partir de la tasa de error en bit del canal (BER) y del tamaño de paquete utilizado (T bytes) como:

     $$P(\text{paquete erróneo}) = 1 - (1 - BER)^{8T} \approx BER \times 8 \times T$$

     $BER \downarrow\downarrow$

   - Por otro lado, sin embargo, el método diferencial en el que se basa este algoritmo, que conlleva la dependencia de cada paquete con el anterior para su correcta descompresión, supone un incremento artificial de la tasa de error en paquete ya que cada pérdida se traduce, inevitablemente, en el descarte de todos los segmentos posteriores al realmente afectado por el error y pertenecientes a su ventana.

3. El tamaño de la ventana de emisión en el momento de producirse un error es un factor que repercute de forma decisiva en el rendimiento del protocolo cuando se utiliza este método de compresión.

   Este parámetro tiene una incidencia doble:

   - Por un lado, sabemos que el emisor sólo puede detectar la pérdida de un paquete a partir de la expiración del temporizador de retransmisión. Puesto que el valor de RTO aumenta a medida que lo hace la ventana del emisor (mientras es la ventana de

congestión y no la de control de flujo del receptor la que limita la emisión de segmentos), la detección se producirá tanto más tarde cuanto mayor sea ésta.

- En segundo lugar, el emisor debe, tras detectar el error, retransmitir toda la ventana afectada. En consecuencia, cuantos más segmentos abarque ésta, más tiempo requerirá su transmisión por el enlace.

4. Existe un compromiso entre la tasa de error en el canal y el tamaño de la ventana del emisor, del que depende el rendimiento final del protocolo de transporte.

   Cuanto mayor es la BER del canal, mayor es el número de errores de transmisión que se generan durante la transferencia, es decir, mayor es el factor $8 \times \mathrm{IU} \times \mathrm{BER}$ de la ecuación (9.6). En este sentido, la mayor tasa de error tiene un impacto negativo en el caudal del enlace.

   Cuanto mayor es la BER del canal, menor es el tamaño que puede alcanzar la ventana del emisor entre error y error y, en consecuencia, menor es el factor **v** de la ecuación (9.6). En este sentido, la mayor tasa de error ayuda a amortiguar el efecto secundario negativo que supone la introducción del método de compresión.

5. El efecto de las ráfagas de error respecto errores aislados no es tan contundente cuando se utiliza compresión ya que, de alguna manera, el algoritmo de Van Jacobson ya está "vacunado" contra las ráfagas de error al ser él mismo el que las genera tras detectar un segmento erróneo. Por construcción, este algoritmo descarta todos los paquetes de la ventana del afectado y, en consecuencia, obliga al emisor a retransmitirlos uno a uno; no distingue, por tanto, entre un mero error simple y un grupo de ellos que afecta a sucesivos segmentos de una misma ventana. Es decir, al tener compresión, el efecto de ráfagas de errores reduce la tasa de error efectiva que percibe el protocolo.

Vemos pues, que cabe pensar en un mecanismo de compresión de cabeceras que sea adecuado también en el caso de tener un canal que introduzca errores. En el siguiente apartado presentamos las propuestas de mejora del algoritmo de compresión de Van Jacobson [RFC1144] presentes en la bibliografía.

9.4 PROPUESTAS

En primer lugar, Degermark y otros presentaron en [DEN96, Deg98] una modificación del algoritmo descrito en [RFC1144], con el objetivo de recuperar más rápidamente la pérdida de consistencia entre el compresor y el descompresor en caso de errores. En este caso el algoritmo opera de la misma forma que en [RFC1144], pero con la particularidad de que, si se produce una pérdida, **basándose en la similitud entre dos segmentos consecutivos**, se aplica dos veces el incremento respecto la cabecera de base. Este algoritmo también es

conocido como el "*twice algorithm*". Por lo tanto, siempre que solamente se pierda un segmento, y que ambos tengan el mismo incremento respecto la base, se recuperará la consistencia, y solamente será necesario retransmitir el segmento afectado por el error. Si la transmisión está libre de errores el algoritmo es exactamente igual al [RFC1144], pero en el caso en que se produzca la pérdida, al perderse la consistencia se aplica el nuevo algoritmo. En caso de que no se recupere la consistencia con este algoritmo, sucederá entonces igual que en [RFC1144], y deberá retransmitirse toda la ventana de transmisión.

En [DEN96, Deg98], Degermark y otros muestran las tasas de recuperación de la consistencia. Cabe destacar, que con este algoritmo, el éxito en recuperar la consistencia en el caso de que los errores afecten a reconocimientos es muy baja. La razón principal es el mecanismo de reconocimientos retardados que se aplica muy frecuentemente en TCP. De esta forma, reconocimientos consecutivos pueden incluir diferentes números de segmento y en consecuencia el "*twice algorithm*" no puede recuperar la pérdida de consistencia. La solución que se plantea también en [DEN96, Deg98] para los reconocimientos es el algoritmo conocido como "*Header Request Algorithm*" en el cual el descompresor manda un mensaje al compresor para que envíe un segmento descomprimido para poder recuperar la pérdida de consistencia. El inconveniente de esta solución es la carga del enlace con estos mensajes de control. Cabe destacar, que por tanto este algoritmo no sería adecuado para ser aplicado en **enlaces asimétricos**, en los que un reconocimiento reconoce a varios segmentos (tal y como se ha descrito en el capítulo de asimetría: *TCP en Enlaces Asimétricos*).

No obstante, esta mejora del algoritmo de Van Jacobson [RFC1144], ya consta como estándar propuesto de Internet, publicado en [RFC2507]. Esta proposición de estándar ha sido posterior al desarrollo que hemos realizado en este ámbito. En ella se contempla la posibilidad de aplicar el "*twice algorithm*" de forma consecutiva hasta un máximo de 16 veces. De esta forma podrían recuperase errores múltiples.

Una segunda alternativa se presenta en [PeM97]. En este caso también se propone una modificación del [RFC1144]. La solución propuesta no tiene como objetivo recuperarse rápidamente de la pérdida de consistencia como en [RFC2507], sino evitarla en sí misma. En [PeM97], Perkins y otros proponen mantener una cabecera base fija, respecto a la cuál todos los segmentos están referidos, a diferencia de las soluciones anteriores en las que cada segmento se basa en el anterior, y por tanto la cabecera base cambia para cada segmento enviado. Esta cabecera base fija se va actualizando en función de un umbral que es parámetro de diseño del algoritmo.

En [RFC1144] y en [RFC2507], cuando se produce un error, la base en el compresor y en el descompresor son diferentes, perdiéndose por tanto la consistencia. No obstante, manteniendo la base fija esta pérdida de consistencia raramente ocurre, ya que cada segmento comprimido es independiente de los anteriores. Las mejoras incorporadas por este algoritmo son especialmente significativas en entornos erróneos como el canal móvil, ya que en caso de

tener un canal libre de errores, los segmentos comprimidos son mayores que en los casos anteriores [RFC1144, RFC2507], en los que normalmente el incremento referente al número de segmento es de un octeto frente a los normalmente tres octetos de [PeM97], produciéndose una sobrecarga mayor y obteniéndose así caudales menores.

Cabe destacar que el algoritmo propuesto en [PeM97] no está completamente libre de la pérdida de consistencia, ya que en caso de que los segmentos que forman la base, que son mandados sin comprimir, se vean afectados por errores, se producirá también la pérdida de consistencia. No obstante, en cualquier caso las prestaciones obtenidas con [PeM97] nunca serán perores que en [RFC1144], en el caso de errores.

Tipo entorno	**[RFC1144]**	**[RFC2507]**	**[PeM97]**
Poco corruptos	sobrecarga baja caudal alto	sobrecarga baja caudal alto	sobrecarga alta caudal bajo
Corruptos	sobrecarga baja caudal muy bajo	sobrecarga baja caudal bastante alto	sobrecarga menor caudal óptimo
Muy corruptos	sobrecarga baja caudal muy bajo	sobrecarga baja caudal cae	sobrecarga baja caudal alto

Tabla 9.1 Comparativa entre las diferentes propuestas

La Tabla 9.1 muestra de forma cualitativa el comportamiento de cada solución en función del tipo de entorno. Evidentemente la solución óptima sería la que tuviese un comportamiento como la proposición de [RFC2507] en entornos poco corruptos y un comportamiento como la proposición de [PeM97] en entornos con muchas pérdidas o con segmentos consecutivos muy dispares.

A partir de estas dos propuestas proponemos una mejora de [RFC1144], que tiene como objetivos el no perder la consistencia pero que además siga comportándose como [RFC1144] y [RFC2507] en caso de tener un canal libre de errores.

9.5 MEJORAS DEL ALGORITMO DE COMPRESIÓN DE CABECERAS

La principal desventaja de [RFC1144] es la necesidad de retransmitir toda la ventana de transmisión una vez se produce un error, la de [RFC2507] la vulnerabilidad a los errores en reconocimientos y la capacidad de recuperar la consistencia solamente en los casos en que se produzcan errores aislados y que exista similitud entre segmentos consecutivos. Y finalmente, el inconveniente de [PeM97] es su peor comportamiento en canales libres de errores.

La mejora que proponemos se basa en la proposición de [PeM97], por lo que a partir de ahora vamos a comparar la mejora que proponemos con el algoritmo en [RFC1144] y en [PeM97]. La opción de [RFC2507] la hemos desechado por las restricciones que ella conlleva. Además, el escenario adecuado para realizar pruebas implica la transferencia de ficheros mediante *ftp*, quitando por tanto una de las restricciones que tiene el algoritmo. Por estas razones se ha

descartado su implementación y consecuentemente su comparación con el algoritmo propuesto.

Para ver de forma más detallada el comportamiento de los diferentes algoritmos nos vamos a ayudar de un ejemplo en el que se muestra la evolución de una transferencia *ftp* de 100Kbytes. La MTU es de 128 octetos. En todos los casos se asume que el caso especial de datos unidireccionales [RFC1144] no está habilitado.

En este caso, el incremento en el número de secuencia que forma parte de cada segmento comprimido es de 88 octetos con el algoritmo de Van Jacobson, ya que la cabecera base para la reconstrucción del segmento es siempre el segmento anterior. No obstante, en el caso de [PeM97], el valor del incremento irá creciendo de 88 en 88 octetos, pasando por tanto de ocupar 1 octeto a ocupar 3 en la cabecera comprimida, tal y como indica [RFC1144]. Además, este incremento en el número de octetos necesarios para codificar el incremento también afecta igualmente al campo de Reconocimiento. Suponemos que la transferencia es unidireccional, con lo que simultáneamente no tendremos activos los campos de Reconocimiento y Número de Secuencia. Respecto al campo de Identificador de Datagrama, en [RFC1144], solamente se manda su valor si es diferente de 1, que es lo que sucede normalmente, en cambio, en [PeM97] este incremento va siempre referido al segmento base, con lo que su valor irá incrementándose. Con un octeto podemos representar un incremento máximo de 255, con lo que si mandamos 255 segmentos consecutivos, con un tamaño de MSS de 88 octetos, tenemos un valor inferior al umbral máximo del número de secuencia y de reconocimiento, con lo que este valor también pasará a codificarse con tres octetos.

En la Tabla 9.2 se muestra, para cada campo del segmento comprimido, la longitud en octetos que éste puede tomar. Los campos sombreados indican que en ambos algoritmos su valor será el mismo. En el resto de campos el valor en negrita es el más común, y en el caso del Identificador de Datagrama, su valor pasará a ocupar 3 octetos después del envío de 255 segmentos sin comprimir.

	[RFC1144]	**[PeM97]**
Máscara	1	1
Número de Conexión	1	1
Checksum	2	2
Puntero Urgente	1 ó 3	1 ó 3
∆Ventana	1 ó 3	1 ó 3
∆ack	1	1 ó **3**
∆Secuencia	1	1 ó **3**
Identificador IP	1 ó **0**	1 ó 3

Tabla 9.2 Tamaño, en octetos, de los campos del segmento comprimido

A partir de esta tabla podemos ver que existe un incremento importante en el número de octetos transmitidos en un caso respecto el otro. Para valores de MTU pequeñas, como es el caso en entornos móviles, el campo de Identificador del datagrama tendrá más importancia que

en le caso de MTU mayores, en los que rara vez alcanzará a ocupar 3 octetos. Dependerá también del tamaño del fichero transmitido.

Estas consideraciones nos llevan a proponer unas modificaciones en el algoritmo [PeM97], para mejorar las prestaciones del algoritmo en caso de que el canal está libre de errores independientemente de la similitud entre segmentos, y a su vez que mantenga o mejore la protección contra la pérdida de consistencia del algoritmo de [PeM97].

En el algoritmo de [PeM97] se cambia la base fija cuando alguno de los campos del segmento comprimido alcanza su valor máximo con el tamaño de 3 octetos. Cabe recordar que los valores se codifican con 1 octeto o con 3, pero en este caso el primer octeto siempre vale cero, codificándose por tanto el valor con 2 octetos. De esta forma el valor máximo es de 2^{16}. Para cambiar la base fija en [PeM97] se envía un segmento descomprimido actualizando así las bases en el compresor y en el descompresor. Nuestra propuesta es la siguiente:

- **Inclusión de un umbral de cambio de base fija.**

- **Cambio de base fija sin necesidad de enviar un segmento descomprimido, en caso de alcanzar el umbral.**

El algoritmo opera de la siguiente forma. Cada nuevo segmento que llega al compresor se comprime respecto a una cabecera base fija. Se calculan las diferencias respecto los campos del nuevo paquete y la base fija. Esta base, obviamente es la misma del descompresor. Una vez se alcanza el valor del umbral en cualquiera de los campos del segmento comprimido, debe cambiarse la base fija. Para hacerlo, el mismo segmento comprimido actualizará la base, evitando por tanto enviar el segmento descomprimido, y reduciendo el valor de los incrementos.

Cuando el compresor detecta que la cabecera comprimida resultante del segmento a enviar excede el umbral, vuelve a comprimirla sobre la base de la cabecera del último segmento enviado. La nueva base será esta cabecera. El compresor en emisión se ocupará también de avisar al descompresor en recepción que se ha realizado un cambio de base. El descompresor, a su vez, al detectar este aviso, aplicará los incrementos que llegan por el canal (cabecera comprimida) sobre la cabecera del último segmento recibido. Finalmente, para mantener la misma base en el compresor que en el descompresor, este último establecerá como base fija la cabecera utilizada en esta operación. Vemos pues, que este algoritmo requiere:

- Almacenamiento en el compresor y descompresor tanto de la cabecera base fija como del la cabecera del último segmento enviado/recibido. En los algoritmos anteriores es suficiente con el almacenamiento de una cabecera.

- Aviso al descompresor del cambio de base. Para que el compresor pueda informar al descompresor sobre un cambio de base proponemos la activación del bit reservado del

octeto de la Máscara de Cambios de la cabecera comprimida. Si el receptor recibe un octeto con dicho bit activo procederá a realizar el cambio de base. La Figura 9.8 muestra el octeto del la Máscara de Cambio. El primer bit reservado se utiliza para avisar del Cambio de Base (CB).

CB	C	I	P	S	A	W	U

Figura 9.8 Máscara de Cambio de la cabecera comprimida

En caso de producirse error, el segmento se retransmitirá descomprimido y se realizará también el cambio de base.

9.5.1 UMBRAL DE CAMBIO DE BASE

El algoritmo propuesto propone la inclusión de un umbral de cambio de base. Este umbral debe asegurar que el comportamiento del algoritmo sea óptimo tanto en caso de tener un canal libre de errores como en el caso de tener un canal con una BER importante.

- Si la BER del canal es pequeña o cero, como menor sea el valor del umbral, mejor rendimiento tendrá el protocolo, ya que los campos de la cabecera comprimida ocuparán su valor mínimo de un octeto. Cada vez que se llegue a este umbral se cambiará la base sin necesidad de mandar ningún segmento descomprimido. El caso óptimo para BER cero es el de realizar el cambio de base en cada segmento. Este sería el caso del [RFC1144]. No obstante, al tener BER diferente de cero, cada cambio de base aumenta la vulnerabilidad del algoritmo a la pérdida de consistencia, tal y como veremos en el siguiente apartado, por lo tanto tendremos un compromiso.

- Si la BER del canal es elevada, por una parte cuanto mayor sea el umbral mejor rendimiento tendrá el protocolo, dado que los cambios de base hacen más vulnerable el algoritmo a la pérdida de consistencia. Por otra parte, cuanto mayor sea el tamaño del segmento más vulnerable será a los errores, con lo que interesaría un umbral pequeño. En cualquier caso, dado que cada segmento erróneo será retransmitido, produciendo un cambio de base, el valor de este umbral pasa a ser irrelevante, ya que el mismo funcionamiento del algoritmo regulará el valor del umbral.

Consecuentemente, aunque exista un compromiso entre la sobrecarga introducida por el algoritmo dependiendo del umbral escogido, y la probabilidad de la pérdida de consistencia, hemos limitado la sobrecarga del Campo de Identificador de Datagrama a un solo octeto, permitiendo así el envío de un máximo de 255 segmentos consecutivos sin cambio de base. El resto de campos pueden ocupar 3 octetos.

9.5.2 PÉRDIDA DE CONSISTENCIA

Vamos a describir ahora en qué situaciones podemos perder la consistencia, viendo así la vulnerabilidad del algoritmo a la pérdida de la misma en relación a las otras propuestas.

Mientras la base fija sea la misma en el compresor y descompresor, aunque se pierda un segmento los posteriores podrán descomprimirse sin perder la consistencia, siempre y cuando el segmento perdido no sea ni el que provoca un cambio de base ni el inmediatamente anterior a este cambio de base. Vamos a analizar con más detalle las dos situaciones:

- En caso de perder el segmento que provoca el cambio de base (tanto sea un segmento descomprimido o uno comprimido con aviso de cambio de base), el compresor trabajará con la base fija nueva y el descompresor con la antigua. Consecuentemente se perderá la consistencia. Es por esta razón que no interesará actualizar muy a menudo la base fija.

- En caso de que se notifique cambio de base mediante un segmento comprimido, la base fija pasa a ser el último segmento enviado/recibido por el compresor/descompresor. Si ese segmento resultó erróneo, la nueva base fija será diferente en el compresor y en el descompresor, provocando así también pérdida de consistencia.

Es por esta razón que existe el compromiso entre la frecuencia en la que se actualiza la base fija y la probabilidad de perder la consistencia.

En cualquier caso, dado que se reduce el tamaño de la cabecera, la protección contra errores será mayor. Además, la actualización de la base se hace mediante un segmento comprimido, a diferencia de [PeM97]. De esta forma la probabilidad de pérdida se reduce.

9.5.3 BENEFICIOS EN UN CANAL LIBRE DE ERRORES

Los beneficios del algoritmo propuesto en un canal libre de errores pueden cuantificarse, teniendo en cuenta que realizamos una transferencia *ftp* de un fichero suficientemente grande como para que la fase de apertura de la ventana de congestión no afecte al total de la transferencia. Suponemos también que la transferencia es unidireccional, con lo que un segmento no puede tener los campos de Reconocimiento y Número de Secuencia simultáneamente. Además, nos centraremos en el flujo de datos, ya que la compresión de cabeceras en el sentido de los reconocimientos solamente sería un aspecto crítico en enlaces asimétricos como el que se ha estudiado en capítulos anteriores. Suponemos también que los campos sombreados en la Tabla 9.2 son iguales para cualquiera de las propuestas.

Con estas hipótesis para poder comparar los diferentes mecanismos, nos bastará con calcular el número de segmentos en los que el campo de Número de secuencia está codificado con 1 o 3 octetos, y el campo de Identificador de Datagrama con 0, 1 o 3 octetos.

Definimos

los estados $\Delta S_i / ID_j$

siendo *i*=1,3 y *j*=0,1,3

de forma que el número de segmentos comprimidos con *i* octetos en el campo de Número de Secuencia y *j* en el campo de Identificador de Datagrama, enviados en cada estado será el valor a determinar en cada uno de los algoritmos propuestos, definido como

$$Longitud\ estado\ \Delta S_i / ID_j = L_{\Delta S_i / ID_j}$$

El paso de un estado a otro vendrá determinando por el umbral correspondiente de cada mecanismo.

Definimos los dos umbrales posibles como

$$\alpha = 2^8 - 1$$
$$\beta = 2^{16} - 1$$

Una vez obtenidos los segmentos enviados en cada unos de los estados $\Delta S_i / ID_j$ posibles para cada mecanismo, deberemos calcular el número de veces que pasamos por cada uno de los estados, para poder enviar la totalidad del **fichero te tamaño F octetos**.

Como veremos más adelante, en un mismo mecanismo de compresión, los estados se suceden consecutivamente, de forma que definimos la **Longitud Total de un Ciclo** como la suma de los segmentos enviados por cada uno de los estados desde el inicio del ciclo (primer segmento enviado), hasta una reinicialización de ciclo (se ha llegado al umbral),

$$Longitud\ Total\ de\ Ciclo = \sum_{i=1,3} \sum_{j=0,1,3} L_{\Delta S_i / ID_j}$$

Definimos M como la cantidad total de segmentos de tamaño MSS que debemos enviar para poder transmitir el fichero de tamaño F,

$$M = [F/MSS]$$

Dado que M puede no ser un múltiplo entero de la Longitud Total de Ciclo, restringiremos el estudio analítico para el caso de que sí lo sea, de forma que el número total de ciclos completos que deberán sucederse para poder enviar el fichero completo es,

$$K = M \big/ Longitud\ Total\ de\ Ciclo$$

Finalmente, definimos la sobrecarga como el número de octetos enviados en los campos de número de secuencia e identificador de datagrama, y el número de segmentos sin comprimir. En general será,

$$Sobrecarga = K * \left(\sum_{i=1,3} \sum_{j=0,1,3} (i+j) * L_{\Delta S_i / ID_j} \right) + SSC * TC$$

siendo SCC el número se segmentos sin comprimir enviados para la actualización de la base y TC el tamaño de la cabecera sin compresión. El valor de SCC será igual a K dadas las restricciones del estudio.

A continuación vamos a calcular las longitudes de cada uno de los estado para cada uno de los mecanismos estudiados.

Algoritmo [RFC1144]

En este caso tenemos que el campo de Identificador de Datagrama no se envía nunca en la cabecera comprimida, ya que siempre vale 1. De forma que

$j = 0$
$i = 1 \text{ y } 3 \text{ si } MSS \leq \alpha$
$i = 3 \text{ si } MSS > \alpha$

a) Si $MSS \leq \alpha$ entonces el ciclo se inicia en $\mathbf{\Delta S_1/ID_0}$, en el que tenemos la sucesión siguiente del campo ΔS

$$(MSS, 2*MSS, 3*MSS, \ldots, (n-1)*MSS) \qquad \text{con límite } \alpha$$

con lo que el número de segmentos de este estado es

$$n-1 = [\alpha / MSS]$$

Una vez superado el umbral pasamos al estado $\mathbf{\Delta S_3/ID_0}$, en el que tenemos la siguiente sucesión del campo ΔS,

$$\left(\left[\alpha/MSS\right]+1\right)*MSS, \left(\left[\alpha/MSS\right]+2\right)*MSS, \ldots, \left(\left[\alpha/MSS\right]+m-1\right)*MSS) \quad \text{con límite } \beta$$

con lo que el número de segmentos de este estado es

$$m-1 = \left[\beta/MSS\right] - \left[\alpha/MSS\right]$$

Una vez superado el umbral reiniciamos pasando al estado inicial $\mathbf{\Delta S_1/ID_0}$ enviando un segmento sin comprimir.

b) Si $MSS > \alpha$ entonces el ciclo se inicia en **$\Delta S_3/ID_0$**, en el que tenemos la sucesión siguiente del campo ΔS

$$(MSS, 2*MSS, 3*MSS, ..., (I-1)*MSS) \qquad \text{con límite } \beta$$

con lo que el número de segmentos de este estado es

$$I - 1 = \left[\beta / MSS \right]$$

Una vez superado el umbral reiniciamos pasando al estado inicial **$\Delta S_3/ID_0$** enviando un segmento sin comprimir.

Algoritmo [PeM97]

En este caso tenemos que el campo de Identificador de Datagrama se envía siempre, de forma que en principio vamos a considerar todos los estados posibles excepto aquellos con *j=0*.

a) Si $MSS \leq \alpha$ entonces el ciclo se inicia en **$\Delta S_1/ID_1$**, en el que tenemos las sucesiones siguientes de los campos ΔS y ID respectivamente

$$(MSS, 2*MSS, 3*MSS, ..., (n-1)*MSS) \qquad \text{con límite } \alpha$$

$$(1, \quad 2, \quad 3 ..., \quad (n-1)) \qquad \text{con límite } \alpha$$

La transición de estado siempre vendrá dada por la llegada al umbral del campo ΔS, ya que MSS>1. Por lo tanto el número de segmentos de este estado es

$$n - 1 = [\alpha / MSS]$$

Una vez superado el umbral pasamos al estado **$\Delta S_3/ID_1$**, en el que tenemos la siguiente sucesión del campo ΔS,

$$\left([\alpha/MSS]+1\right)*MSS, \left([\alpha/MSS]+2\right)*MSS, ..., ([\alpha/MSS]+m-1)*MSS) \quad \text{con límite } \beta$$
$$\left([\alpha/MSS]+1\right), \quad \left([\alpha/MSS]+2\right), ..., \quad ([\alpha/MSS]+m-1)) \quad \text{con límite } \alpha$$

Al tener diferentes límites cada una de las sucesiones, debemos estudiar cual de ellas alcanzará en primer lugar el límite establecido.

El límite de cada una se obtiene para los siguientes valores de m:

$$m_{\Delta S} = \left[\beta / MSS \right] - \left[\alpha / MSS \right] + 1$$

$$m_{ID} = \alpha - [\alpha / MSS] + 1$$

Para obtener cuál de los dos valores se produce antes definimos la función

$$f(MSS) = m_{\Delta S} - m_{ID} = \left[\beta / MSS\right] - \alpha$$

Podemos afirmar que f(MSS) es estrictamente decreciente siempre que MSS>1 con lo que obteniendo los ceros de la función podremos determinar qué umbral se produce antes.

Los ceros de la función son tales que

$$\left[\beta / MSS\right] = \alpha$$

Con lo que

$$MSS = \alpha + 1$$

y

$$MSS = \alpha + 2$$

Estos valores están fuera del intervalo en estudio en este subapartado. Para

$$MSS \leq \alpha$$

se cumple entonces que f(MSS)>0, con lo que el umbral que se alcanza primero es el de ID. Por lo tanto, el número de segmentos enviados en este estado es

$$m_{ID} - 1 = \alpha - [\alpha / MSS]$$

Una vez superado el umbral pasamos al estado $\mathbf{\Delta S_3/ID_3}$, en el que tenemos las siguientes sucesiones de los campos ΔS e ID sucesivamente

$$((\alpha+1)*MSS, (\alpha+2)*MSS, \ldots, (\alpha+l-1)*MSS) \quad \text{con límite } \beta$$

$$((\alpha+1), \quad (\alpha+2), \ldots, \quad (\alpha+l-1)) \quad \text{con límite } \beta$$

En este caso el límite de las sucesiones es el mismo, con lo que limitará siempre la de ΔS, obteniendo un número de segmentos enviados de

$$l - 1 = \left[\beta / MSS\right] - \alpha$$

Una vez superado el umbral reiniciamos pasando al estado inicial **$\Delta S_1/ID_1$**, enviando un segmento sin comprimir.

b) Si $MSS > \alpha$ entonces el ciclo se inicia en **$\Delta S_3/ID_1$**, en el que tenemos las sucesiones siguientes de los campos ΔS e ID

$(MSS, 2*MSS, 3*MSS, ..., (n-1)*MSS)$ *con límite* β

$(1, \quad 2, \quad 3, ..., \quad (n-1))$ *con límite* α

Al igual que ha sucedido en el caso anterior, al tener límites diferentes debemos evaluar qué umbral se supera con anterioridad. En este caso

$$I_{\Delta S} = \left[\beta / MSS \right] - 1$$

$$I_{ID} = \alpha + 1$$

Definimos la función

$$g(MSS) = I_{\Delta S} - I_{ID} = \left[\beta / MSS \right] - \alpha = f(MSS)$$

Dado que las funciones coinciden, aplican, los mismos resultados, pero ahora aplicados al intervalo de estudio $MSS > \alpha$, por lo tanto

- Para $MSS > \alpha + 2$, limita el umbral de ΔS, con lo que el número de segmentos enviados en este estado es de

  $$I_{\Delta S} - 1 = \left[\beta / MSS \right]$$

  Una vez alcanzado el umbral reiniciamos pasando al estado inicial **$\Delta S_3/ID_1$**, y enviamos un segmento sin comprimir.

- Para $MSS = \alpha + 1$ *y* $MSS = \alpha + 2$, ambos llegan al límite con lo que el número de segmentos enviados en este estado es de

  $$I_{ID} - 1 = \alpha$$

  Una vez alcanzado el umbral reiniciamos pasando al estado inicial **$\Delta S_3/ID_1$**, y enviamos un segmento sin comprimir.

Algoritmo Propuesto

En este caso tenemos que el campo de Identificador de Datagrama se envía siempre, pero su máximo está en un octeto, por lo que *j=1* siempre.

a) Si $MSS \leq \alpha$ entonces el ciclo se inicia en **$\Delta S_1/ID_1$**, en el que tenemos las sucesiones

$$(MSS, 2*MSS, 3*MSS, ..., (n-1)*MSS) \quad \text{con límite } \alpha$$

$$(1, \quad 2, \quad 3 ..., \quad (n-1)) \quad \text{con límite } \alpha$$

La transición de estado siempre vendrá dada por la llegada al umbral del campo ΔS, ya que MSS>1. Por lo tanto el número de segmentos de este estado es

$$n-1 = [\alpha / MSS]$$

Una vez superado el umbral pasamos al estado **$\Delta S_3/ID_1$**, en el que tenemos la siguiente sucesión del campo ΔS,

$$((\lceil \alpha / MSS \rceil + 1) * MSS, (\lceil \alpha / MSS \rceil + 2) * MSS, ..., (\lceil \alpha / MSS \rceil + m-1) * MSS) \quad \text{con límite } \beta$$
$$((\lceil \alpha / MSS \rceil + 1), \quad (\lceil \alpha / MSS \rceil + 2), ..., \quad (\lceil \alpha / MSS \rceil + m-1)) \quad \text{con límite } \alpha$$

Al tener diferentes límites cada una de las sucesiones, debemos estudiar cual de ellas alcanzará en primer lugar el límite establecido. Al igual que en los casos anteriores obtenemos que para el rango de MSS considerado quien provoca la transición es ID, con lo que, el número de segmentos enviados en el estado **$\Delta S_3/ID_1$** es de

$$m_{ID} - 1 = \alpha - [\alpha / MSS]$$

Una vez alcanzado el umbral reiniciamos pasando al estado inicial **$\Delta S_1/ID_1$**. En este caso re realiza el cambio de base sin necesidad de enviar ningún segmento sin comprimir.

b) Si $MSS > \alpha$ entonces el ciclo se inicia en **$\Delta S_3/ID_1$**. De la misma forma que se ha desarrollado en la propuesta de [PeM97] se obtiene

- Para $MSS > \alpha + 2$, limita el umbral de ΔS, con lo que el número de segmentos enviados en este estado es de

$$I_{\Delta S} - 1 = \left[\beta / MSS \right]$$

Una vez alcanzado el umbral reiniciamos pasando al estado inicial **$\Delta S_3/ID_1$**, sin necesidad de enviar ningún segmento sin comprimir.

- Para $MSS = \alpha + 1$ *y* $MSS = \alpha + 2$, ambos llegan al límite con lo que el número de segmentos enviados en este estado es de

$$I_{ID} - 1 = \alpha$$

Una vez alcanzado el umbral reiniciamos pasando al estado inicial **$\Delta S_3/ID_1$**, sin necesidad de enviar ningún segmento sin comprimir.

A continuación presentamos en la Tabla 9.3 el resumen de los resultados obtenidos.

RFC1144

	$MSS \le \alpha$	$MSS > \alpha$
Longitud del Ciclo	$\left\lceil \beta / MSS \right\rceil$	
K	$\lceil F/MSS \rceil / \left\lceil \beta / MSS \right\rceil$	
Sobrecarga	$K^*\lceil \alpha / MSS \rceil +$ $K^*3^*\left(\left\lceil \beta / MSS \right\rceil - \lceil \alpha / MSS \rceil\right) +$ k^*TC	$K^*3^*\left\lceil \beta / MSS \right\rceil$

PeM97

	$MSS \le \alpha$	$MSS = \alpha + 1, \alpha + 2$	$MSS > \alpha$
Longitud del Ciclo	$\left\lceil \beta / MSS \right\rceil$	α	$\left\lceil \beta / MSS \right\rceil$
K	$\lceil F/MSS \rceil / \left\lceil \beta / MSS \right\rceil$	$\lceil F/MSS \rceil / \alpha$	$\lceil F/MSS \rceil / \left\lceil \beta / MSS \right\rceil$
Sobrecarga	$K^*2^*\lceil \alpha / MSS \rceil +$ $K^*4^*(\alpha - \lceil \alpha / MSS \rceil) +$ $K^*6^*\left(\left\lceil \beta / MSS \right\rceil - \alpha\right) +$ k^*TC	$K^*4^* \alpha +$ K^*TC	$K^*4^*\left\lceil \beta / MSS \right\rceil +$ K^*TC

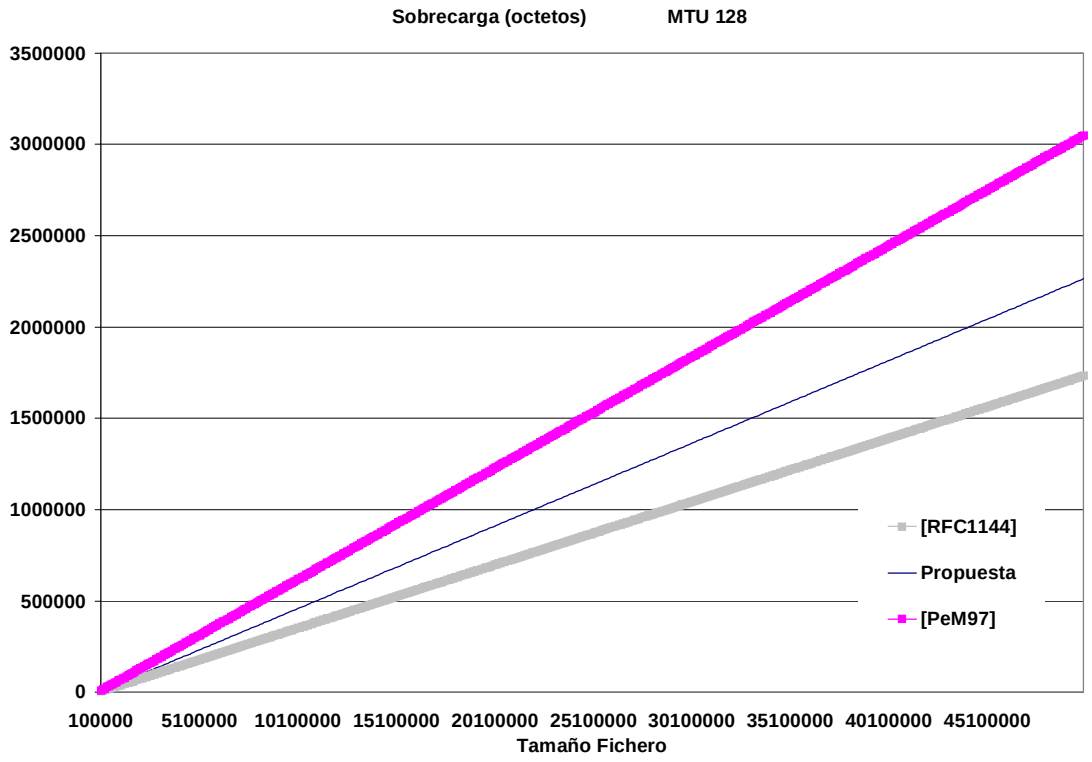
Propuesta

	$MSS \leq \alpha$	$MSS = \alpha + 1, \alpha + 2$	$MSS > \alpha$
Longitud del Ciclo	α	α	$\left[\beta / MSS\right]$
K	$\lceil F/MSS \rceil / \alpha$	$\lceil F/MSS \rceil / \alpha$	$\lceil F/MSS \rceil / \left[\beta / MSS\right]$
Sobrecarga	$K*2*[\alpha / MSS] +$ $K*4*(\alpha - [\alpha / MSS])$	$K*4*\alpha$	$K*4*\left[\beta / MSS\right]$

Tabla 9.3 Resumen de las ecuaciones obtenidas para cada algoritmo

Como conclusiones podemos obtener que, en cualquier caso, en un canal libre de errores nuestra propuesta es mejor al caso de [PeM97], y es ligeramente peor a la [RFC1144] debido al octeto u octetos de Identificador de Datagrama que se envían en cada segmento comprimido. No obstante, el hecho de no enviar ningún segmento sin comprimir en nuestra propuesta, a medida que aumenta el valor del tamaño de fichero transferido la diferencia entre la sobrecarga enviada por [RFC1144] y nuestra propuesta se reduce y se aumentan las diferencias con la [PeM97].

En la Figura 9.9 se muestra la evolución del valor de sobrecarga, para dos valores de MTU.



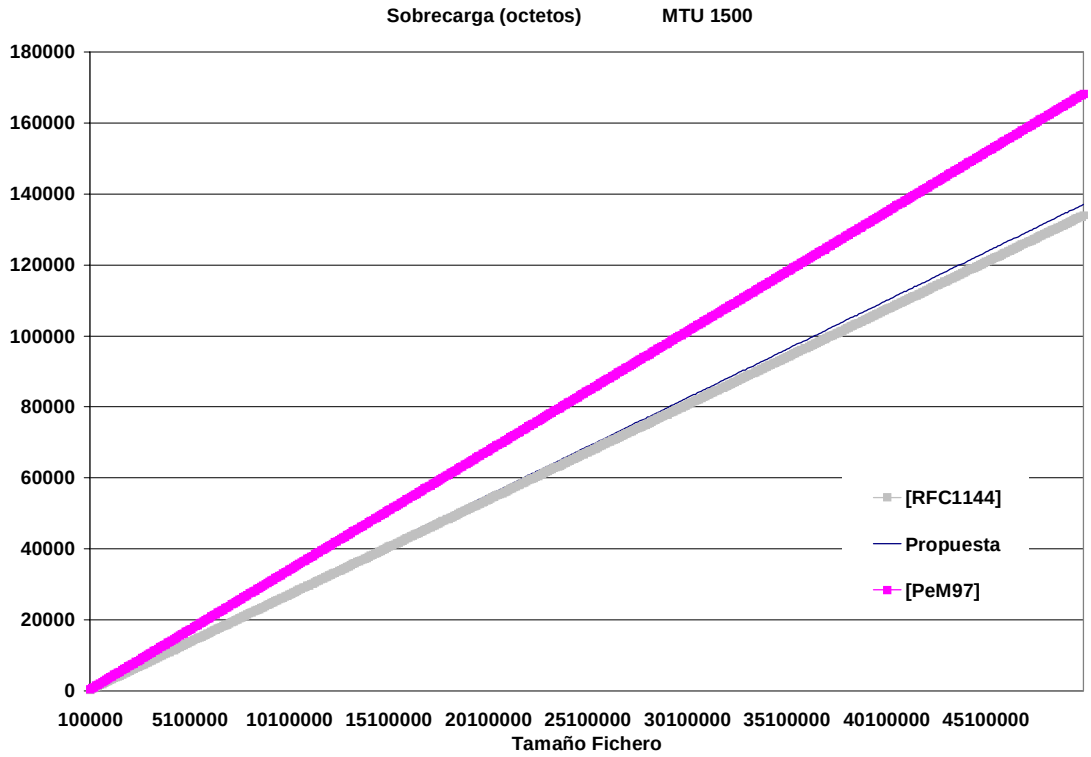


Figura 9.9 Evolución de la sobrecarga en función del tamaño del fichero. De arriba a abajo las líneas representan la propuesta [PeM97], nuestra propuesta y [RFC1144].

9.6 RESULTADOS

En este apartado vamos a comentar los resultados obtenidos al comparar los diferentes algoritmos de compresión de cabeceras. El estudio se ha realizado variando la BER del canal y el tamaño de la MTU del enlace.

En primer lugar presentamos en la Figura 9.10, la evolución del número de secuencia de una transferencia realizada con el algoritmo propuesto. El tamaño de fichero es de 20Kbytes, la velocidad de 19200bps y la MTU de 296 octetos. Comparándola con la Figura 9.2 podemos ver que no se requiere la retransmisión de todos los segmentos de la ventana, y se mejora en caudal, obteniendo una pendiente superior al caso de no tener compresión.

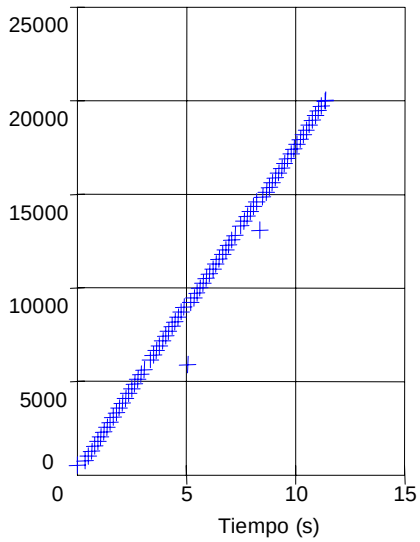


Figura 9.10 Evolución del número de secuencia.

En los siguientes apartados se presentan los resultados obtenidos para cada uno de los algoritmos de compresión evaluados, en función de la BER del canal, a partir de un patrón de errores deterministas y de la MTU. El tamaño del fichero transferido es de 100Kbytes y la velocidad del enlace de 9600bps.

9.6.1 EFECTO DE LA BER DEL CANAL

La propuesta del algoritmo está pensada para mejorar el comportamiento en cuanto a caudal en caso de que el canal esté libre de errores, pero manteniendo la protección en caso de errores. Es por esta razón que se presentan los resultados obtenidos en función de la BER del canal, introduciendo errores de forma aislada.

En primer lugar, en la Figura 9.11 se compara para cada una de las MTU analizadas los efectos de utilizar compresión de cabeceras o no.

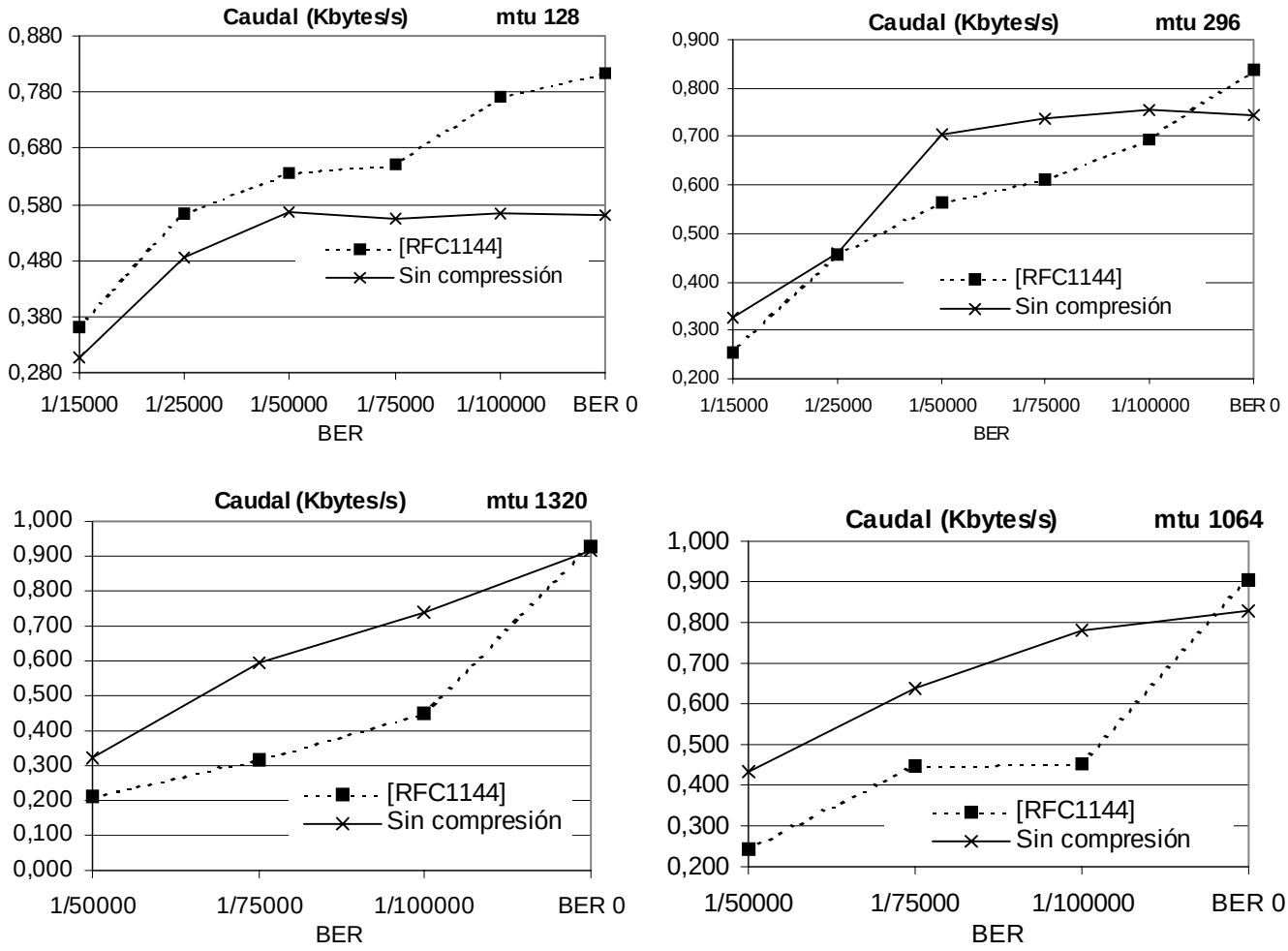


Figura 9.11 Caudal obtenido con y sin compresión

Podemos ver como para la MTU de 128 octetos, es mejor comprimir los datos por la gran cantidad de información de cabeceras que viaja en cada paquete. No obstante a partir de MTU de 296 octetos puede verse como este factor ya no es relevante, obteniendo mejor caudal en el caso de no comprimir.

En la Figura 9.12 se compara el caudal obtenido para cada una de las propuestas en función de la BER y para una MTU de 128 octetos.

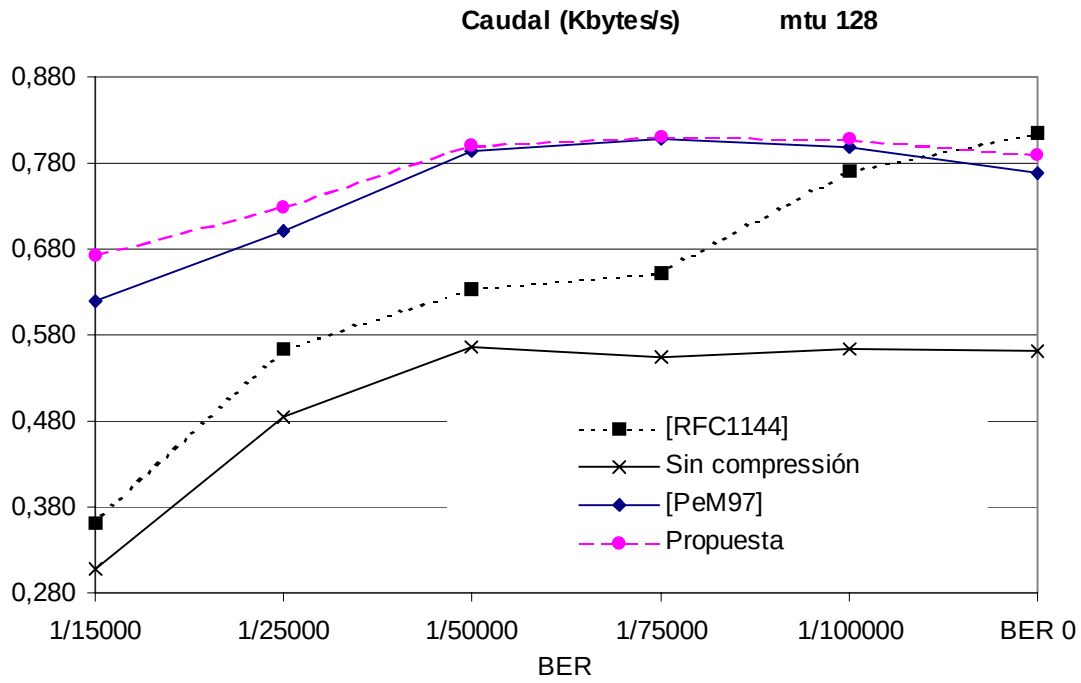


Figura 9.12 Caudal en función de la BER

Podemos observar que en cualquier caso con la mejora propuesta se obtiene un caudal ligeramente superior al algoritmo [PeM97], es decir además la nueva propuesta está más protegida contra los errores. En caso de tener el enlace libre de errores, nuestra propuesta se acerca más a [RFC1144].

No obstante, vemos como se produce un comportamiento no esperado, ya que en cualquiera de las propuestas a medida que se reduce la BER del canal, si bien el caudal debería incrementarse hasta llegar al máximo valor para BER cero, se produce una ligera caída. Esto es debido a un mal funcionamiento de la implementación TCP/IP/PPP utilizada. Tras analizar los posibles efectos y estar en contacto con implementadores de la plataforma Linux no se ha podido solucionar este comportamiento anómalo. Los módulos TCP e IP envían los segmentos de forma normal, pero no son enviados por el medio de transmisión físico (por lo tanto el comportamiento anómalo se sitúa por debajo de IP). Dado que TCP no recibe dichos paquetes, se activan los mecanismos de retransmisión provocando la caída en el caudal. Este funcionamiento se produce independientemente del algoritmo implementado.

Con este problema de implementación, los resultados con el canal libre de errores no son tan positivos como eran de esperar tal y como se ha justificado en el apartado anterior. Si bien existe una mejora, ésta es poca significativa. El hecho de producirse estas retransmisiones en el canal teóricamente libre de errores implica que no estemos en el caso de BER cero. Podría cuestionarse en este caso la aplicación de este nuevo algoritmo. Los esfuerzos están en solucionar este posible error de implementación en la plataforma utilizada y corroborar los resultados experimentalmente.

Además, tal y como se ha justificado en el apartado anterior, la mejora de la propuesta debería aumentar a medida que lo hace el tamaño del fichero. No obstante, las pruebas realizadas muestran que este comportamiento anómalo no permite corroborarlo tampoco con tamaños de fichero elevados.

En la Figura 9.13 se presentan los resultados obtenidos para un fichero de 1Mbyte. Puede observarse que los resultados son muy similares al caso presentado en la Figura 9.12.

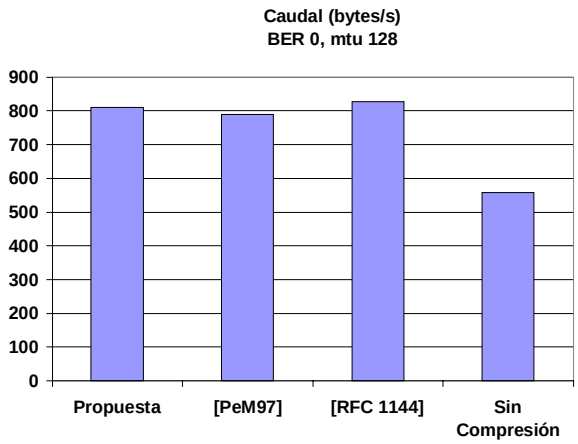


Figura 9.13. Transferencias realizadas con un fichero de 1Mbyte

A continuación en la Figura 9.14 se presentan los resultados obtenidos para el resto de MTU. De estas gráficas pueden extraerse las mismas conclusiones.

El mecanismo de compresión en [RFC1144] no es capaz de evitar la pérdida de consistencia en presencia de errores en el enlace. A mayor numero de errores más veces la pierde provocando la reducción del caudal. Indudablemente, tanto la nueva propuesta como la proposición descrita en [PeM97] se basan en el mecanismo de una base fija que evita la pérdida de consistencia dando uniformidad en el trazado del caudal.

Si aumentamos el tamaño de segmento el decrecimiento de todos los mecanismos será más importante aunque aun se mantiene la uniformidad de los mecanismos de base fija.

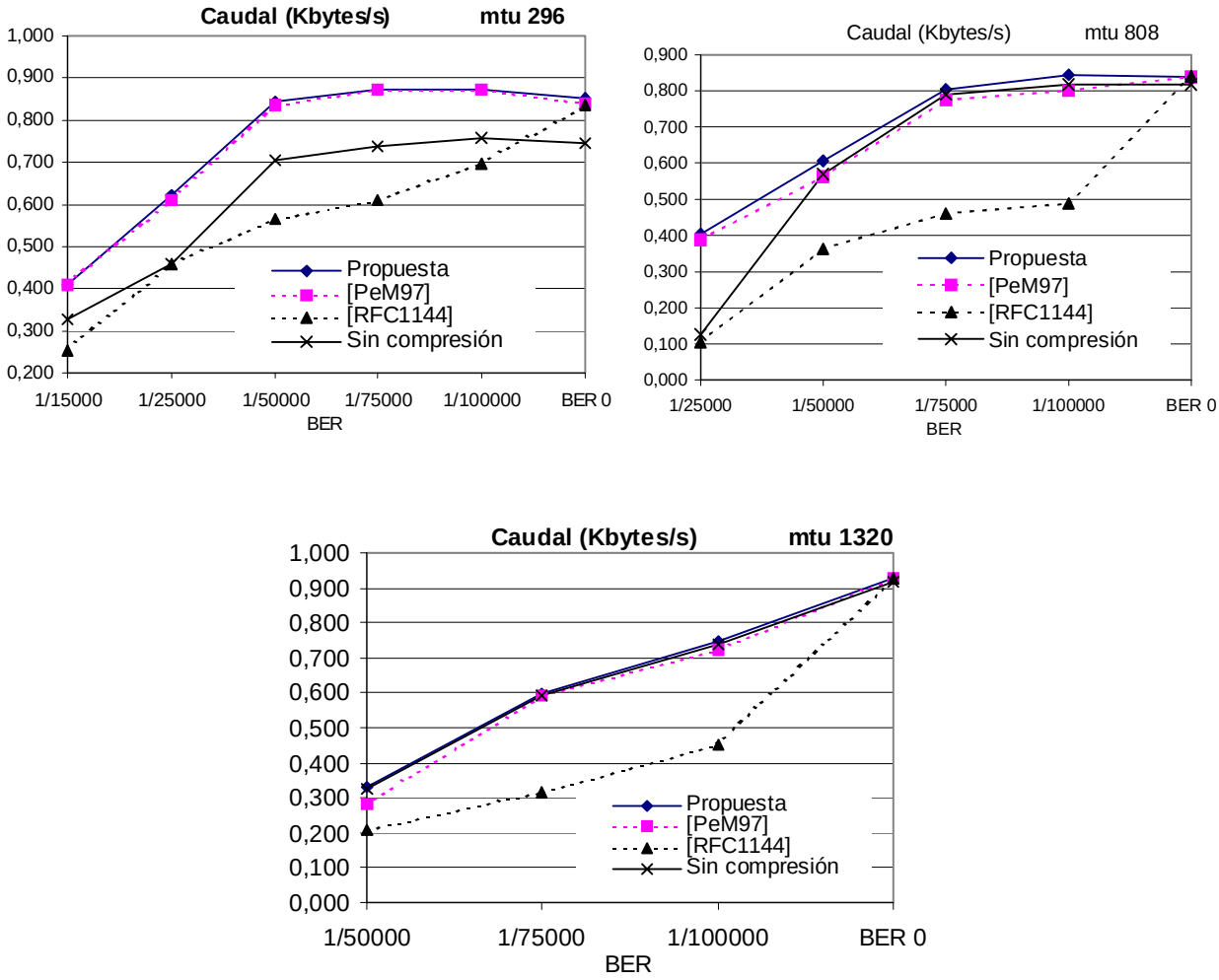


Figura 9.14 Evolución del caudal en función de la BER para diferentes MTU.

Para valores mayores de MTU se reducen aún más el número de cabeceras completas de 40 bytes necesaria para enviar y el mecanismo de compresión que proponemos se asemeja más al resultado obtenido sin compresión. Esto indica que la ganancia en comprimir cabeceras es mucho menor para tamaños grandes de segmentos. Sin embargo, el mecanismo descrito en [RFC1144] sigue muy por debajo del resultado obtenido sin compresión para toda tasa de error. Únicamente para un enlace sin errores el throughput alcanzado iguala a los otros resultados.

9.6.2 EFECTO DE LA MTU

En este apartado se muestran los resultados anteriores representando el caudal en función de la MTU para cada valor de BER. De esta forma se puede observar como se comporta la propuesta independientemente de la MTU utilizada.

La Figura 9.15 muestra para cada tasa de error la evolución del caudal en función de la MTU del enlace.

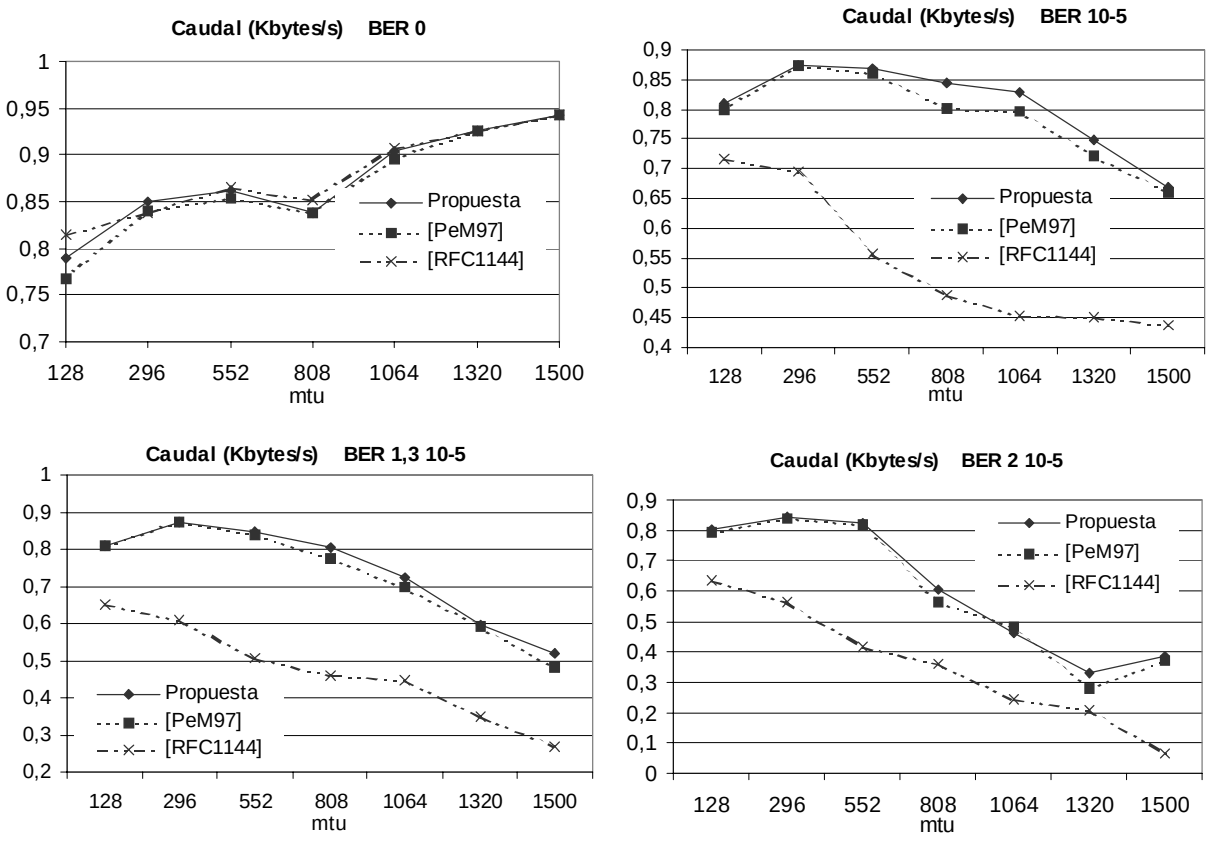


Figura 9.15 Evolución del caudal para cada tasa de error en función de la MTU del enlace

Para un tamaño de segmento pequeño el efecto de la limitación de la sobrecarga evita el envío de datos innecesarios y se observa el incremento de caudal. A medida que el tamaño de segmento aumenta se produce la lógica igualación de todas las trazas debido al menor numero de cabeceras a enviar.

En la evolución del throughput en función del tamaño de segmento también queda reflejado el efecto de la pérdida de la consistencia en la proposición descrita en [RFC1144].

De las figuras anteriores se puede observar otra vez la constancia que caracterizan los mecanismos de base fija. Para una tasa de error de 10^{-5}, el throughput no decae de forma decisiva hasta tamaños de segmento superiores a 1320 bytes, y para una tasa de error de $1{,}3\ 10^{-5}$ se mantiene a un nivel muy alto hasta un tamaño de segmento de 808 bytes.

Una segunda observación que se obtiene de las gráficas de la Figura 9.14 es la posición del máximo en la traza para los mecanismos de base fija. Para tamaños de segmento superiores a 296 bytes el caudal decrece a medida que aumenta el tamaño de segmento. La razón reside en el aumento del coste en tiempo que supone el retransmitir cada segmento corrupto. Cada error supone la retransmisión de (40+MSS) bytes y a mayor tamaño de segmento mayor es dicho coste. Sin embargo, cuánto mayor es el tamaño de segmento menor es el número de cabeceras que se deben utilizar y para tamaños inferiores a 296 bytes, el coste de las retransmisiones pierde peso con respecto al ahorro en número de cabeceras. Para tasas de

error mayores el número de retransmisiones aumenta y el ahorro en número de cabeceras es menos significativo. Para tasas de error muy grandes el caudal empieza la caída ya desde el tamaño de segmento mínimo.

9.7 CONCLUSIONES

En este capítulo hemos analizado la problemática de los algoritmos de compresión de cabeceras en entornos móviles de baja velocidad, concretamente se ha estudiado el estándar vigente en comunicaciones TCP/IP/PPP, [RFC1144]. Las conclusiones obtenidas al respecto pueden resumirse en:

- El algoritmo de compresión de cabeceras [RFC1144] es adecuado para cualquier MTU del enlace en caso de tener un canal libre de errores. Lógicamente a mayor MTU el efecto de la compresión es menos relevante. A medida que la tasa de error del canal aumenta, el algoritmo [RFC1144] deja de ser eficiente debido a la pérdida de consistencia entre el compresor y descompresor. En este ámbito se han estudiado con detalle los diferentes aspectos que afectan a este método de compresión.

- Se ha visto la necesidad de pensar en la mejora del algoritmo de compresión para que también opere en el caso de enlaces con tasas de error en bit apreciable.

Para mejorar el comportamiento de [RFC1144] en enlaces radio se partió del análisis de dos propuestas en la bibliografía [PeM97] y [RFC2507]. De éstas se concluye lo siguiente:

- En [RFC2507] se propone una mejora de [RFC1144] para solucionar la pérdida de consistencia, pero tiene dos restricciones importantes. Por una parte solamente se recupera la consistencia en caso de que únicamente resulte corrompido un segmento de forma consecutiva. Y por otra parte, dado que se basa en la similitud entre segmentos consecutivos, solamente se recupera la consistencia en caso de que los incrementos codificados en la cabecera comprimida sean iguales de un segmento al siguiente.

- En [PeM97] se plantea el evitar la pérdida de consistencia, eliminando la dependencia de un segmento con el inmediatamente anterior, y haciendo esta dependencia con uno definido como base. Esta solución tiene el inconveniente de la sobrecarga en caso de tener un canal libre de errores.

A partir de lo anterior, se pasa a proponer una mejora de los algoritmos evaluados. El propósito de la nueva propuesta es obtener un buen comportamiento en cuanto a caudal en el caso de canal libre de errores, y mantenerlo o mejorarlo en el caso de tener errores en el canal. Las diferentes propuestas se han implementado y evaluado en una plataforma real. De los resultados obtenido podemos concluir:

- Hemos observado que en cualquier caso con la mejora propuesta se obtiene un caudal ligeramente superior al algoritmo [PeM97], es decir además **la nueva propuesta está más protegida contra los errores**. En caso de tener el **enlace libre de errores**, nuestra propuesta **se acerca más a la óptima** [RFC1144].

- Se ha detectado un comportamiento no esperado, ya que en cualquiera de las propuestas a medida que se reduce la BER del canal, si bien el caudal debería incrementarse hasta llegar al máximo valor para BER cero, se produce una ligera caída. Esto es debido a un mal funcionamiento de la implementación TCP/IP/PPP utilizada. Tras analizar los posibles efectos y estar en contacto con implementadores de la plataforma Linux no se ha podido solucionar este comportamiento anómalo.

  Con este problema de implementación, los resultados con el canal libre de errores no son tan positivos como eran de esperar tal y como se ha justificado analíticamente. Si bien existe una mejora, ésta es poca significativa. El hecho de producirse estas retransmisiones en el canal teóricamente libre de errores implica que no estemos en el caso de BER cero. Los esfuerzos están en solucionar este posible error de implementación en la plataforma utilizada y corroborar los resultados experimentalmente.